

Manual Version 1.6  
Software Version 2.5

# Doosan Robot

M0609 | M0617 | M1013 | M1509  
A0509 | A0509s | A0912 | A0912s

## Programming Guide



# Table of Contents

|  |           |
|--|-----------|
| <b>Preface</b> .....                               | <b>29</b> |
| Copyright .....                                    | 29        |
| <b>1. DRL Basic Syntax</b> .....                   | <b>30</b> |
| 1.1 Indent .....                                   | 30        |
| 1.2 Comment .....                                  | 31        |
| 1.3 Variable name .....                            | 32        |
| 1.4 Numeric value .....                            | 33        |
| 1.5 String .....                                   | 34        |
| 1.5.1 String .....                                 | 34        |
| 1.5.2 +, * .....                                   | 35        |
| 1.5.3 Indexing & slicing .....                     | 35        |
| 1.6 list .....                                     | 36        |
| 1.7 tuple .....                                    | 37        |
| 1.8 dictionary .....                               | 37        |
| 1.9 Function .....                                 | 38        |
| 1.10 Scoping rule .....                            | 39        |
| 1.11 Parameter mode .....                          | 40        |
| 1.12 pass .....                                    | 41        |
| 1.13 if .....                                      | 42        |
| 1.14 while .....                                   | 43        |
| 1.15 for .....                                     | 44        |
| 1.16 break .....                                   | 44        |
| 1.17 continue .....                                | 45        |
| 1.18 Else in a loop .....                          | 45        |
| <b>2. Motion-related Commands</b> .....            | <b>46</b> |
| 2.1 posj(q1=0, q2=0, q3=0, q4=0, q5=0, q6=0) ..... | 46        |
| 2.2 posx(x=0, y=0, z=0, w=0, p=0, r=0) .....       | 47        |

---

|      |  |     |
|------|--|-----|
| 2.3  | trans(pos, delta, ref, ref_out) .....            | 48  |
| 2.4  | posb(seg_type, posx1, posx2=None, radius=0)..... | 50  |
| 2.5  | fkin(pos, ref).....                              | 51  |
| 2.6  | ikin(pos, sol_space, ref) .....                  | 53  |
| 2.7  | addto(pos, add_val=None) .....                   | 55  |
| 2.8  | set_velj(vel).....                               | 56  |
| 2.9  | set_accj(acc).....                               | 57  |
| 2.10 | set_velx(vel1, vel2) .....                       | 58  |
| 2.11 | set_velx(vel).....                               | 59  |
| 2.12 | set_accx(acc1, acc2).....                        | 60  |
| 2.13 | set_accx(acc) .....                              | 61  |
| 2.14 | set_tcp(name).....                               | 62  |
| 2.15 | set_ref_coord(coord).....                        | 63  |
| 2.16 | movej .....                                      | 65  |
| 2.17 | movevel .....                                    | 68  |
| 2.18 | movejx .....                                     | 71  |
| 2.19 | movevec .....                                    | 74  |
| 2.20 | movesj.....                                      | 78  |
| 2.21 | movesx .....                                     | 81  |
| 2.22 | moveb.....                                       | 85  |
| 2.23 | move_spiral .....                                | 89  |
| 2.24 | move_periodic .....                              | 92  |
| 2.25 | move_home.....                                   | 95  |
| 2.26 | amovej.....                                      | 97  |
| 2.27 | amovevel .....                                   | 100 |
| 2.28 | amovejx .....                                    | 103 |
| 2.29 | amovevec .....                                   | 106 |
| 2.30 | amovesj.....                                     | 109 |
| 2.31 | amovesx.....                                     | 112 |
| 2.32 | amoveb.....                                      | 115 |

---

|  |     |
|--|-----|
| 2.33 amove_spiral.....   | 118 |
| 2.34 amove_periodic .....  | 121 |
| 2.35 mwait(time=0).....  | 124 |
| 2.36 begin_blend(radius=0).....                                      | 126 |
| 2.37 end_blend().....  | 128 |
| 2.38 check_motion() .....  | 129 |
| 2.39 stop(st_mode).....  | 130 |
| 2.40 change_operation_speed(speed) .....                             | 132 |
| 2.41 wait_manual_guide().....  | 134 |
| 2.42 wait_nudge().....   | 136 |
| 2.43 enable_alter_motion(n,mode,ref,limit_dPOS,limit_dPOS_per) ..... | 138 |
| 2.44 alter_motion([x,y,z,rx,ry,rz]) .....                            | 141 |
| 2.45 disable_alter_motion().....                                     | 143 |

### **3. Auxiliary Control Commands ..... 145**

|  |     |
|--|-----|
| 3.1 get_control_mode().....                | 145 |
| 3.2 get_control_space() .....              | 146 |
| 3.3 get_current_posj() .....               | 147 |
| 3.4 get_current_velj().....                | 148 |
| 3.5 get_desired_posj().....                | 149 |
| 3.6 get_desired_velj() .....               | 150 |
| 3.7 get_current_posx(ref).....             | 151 |
| 3.8 get_current_tool_flange_posx(ref)..... | 153 |
| 3.9 get_current_velx(ref).....             | 154 |
| 3.10 get_desired_posx(ref).....            | 155 |
| 3.11 get_desired_velx(ref) .....           | 156 |
| 3.12 get_current_solution_space().....     | 157 |
| 3.13 get_current_rotm(ref).....            | 158 |
| 3.14 get_joint_torque() .....              | 159 |
| 3.15 get_external_torque().....            | 160 |

---

|           |   |            |
|-----------|---|------------|
| 3.16      | get_tool_force(ref).....  | 161        |
| 3.17      | get_solution_space(pos) .....   | 162        |
| 3.18      | get_orientation_error(xd, xc, axis) .....                                 | 163        |
| <b>4.</b> | <b>Other Settings and Safety-related Commands .....</b>                   | <b>164</b> |
| 4.1       | get_workpiece_weight().....   | 164        |
| 4.2       | reset_workpiece_weight().....   | 165        |
| 4.3       | set_tool(name) .....  | 166        |
| 4.4       | set_tool_shape(name).....   | 167        |
| 4.5       | set_singularity_handling(mode).....                                       | 168        |
| <b>5.</b> | <b>Force/Stiffness Control and Other User-Friendly<br/>Features .....</b> | <b>170</b> |
| 5.1       | parallel_axis(x1, x2, x3, axis, ref) .....                                | 170        |
| 5.2       | parallel_axis(vect, axis, ref).....                                       | 172        |
| 5.3       | align_axis(x1, x2, x3, pos, axis, ref).....                               | 174        |
| 5.4       | align_axis(vect, pos, axis, ref).....                                     | 176        |
| 5.5       | is_done_bolt_tightening(m=0, timeout=0, axis=None) .....                  | 177        |
| 5.6       | release_compliance_ctrl().....  | 179        |
| 5.7       | task_compliance_ctrl(stx, time).....                                      | 180        |
| 5.8       | set_stiffnessx(stx, time) .....   | 182        |
| 5.9       | calc_coord(x1, x2, x3, x4, ref, mod).....                                 | 183        |
| 5.10      | set_user_cart_coord(pos, ref).....  | 185        |
| 5.11      | set_user_cart_coord(x1, x2, x3, pos, ref).....                            | 187        |
| 5.12      | set_user_cart_coord(u1, v1, pos, ref).....                                | 189        |
| 5.13      | overwrite_user_cart_coord(id, pos, ref).....                              | 191        |
| 5.14      | get_user_cart_coord(id).....  | 192        |
| 5.15      | set_desired_force(Fd, dir, time, mod).....                                | 193        |
| 5.16      | release_force(time=0) .....   | 195        |
| 5.17      | check_position_condition(axis, min, max, ref, mod, pos).....              | 196        |

---

|      |   |     |
|------|---|-----|
| 5.18 | check_force_condition(axis, min, max, ref).....                 | 198 |
| 5.19 | check_orientation_condition(axis, min, max, ref, mod).....      | 200 |
| 5.20 | check_orientation_condition(axis, min, max, ref, mod, pos)..... | 202 |
| 5.21 | coord_transform(pose_in, ref_in, ref_out).....                  | 204 |

## **6. System Commands..... 206**

|            |  |            |
|------------|--|------------|
| <b>6.1</b> | <b>IO Related.....</b>                                 | <b>206</b> |
| 6.1.1      | set_digital_output(index, val=None).....               | 206        |
| 6.1.2      | set_digital_outputs(bit_list).....                     | 208        |
| 6.1.3      | set_digital_outputs(bit_start, bit_end, val).....      | 209        |
| 6.1.4      | get_digital_input(index).....                          | 211        |
| 6.1.5      | get_digital_inputs(bit_list).....                      | 212        |
| 6.1.6      | get_digital_inputs(bit_start, bit_end).....            | 213        |
| 6.1.7      | wait_digital_input(index, val, timeout=None).....      | 214        |
| 6.1.8      | set_tool_digital_output(index, val=None).....          | 215        |
| 6.1.9      | set_tool_digital_outputs(bit_list).....                | 216        |
| 6.1.10     | set_tool_digital_outputs(bit_start, bit_end, val)..... | 217        |
| 6.1.11     | get_tool_digital_input(index).....                     | 219        |
| 6.1.12     | get_tool_digital_inputs(bit_list).....                 | 220        |
| 6.1.13     | get_tool_digital_inputs(bit_start, bit_end).....       | 221        |
| 6.1.14     | wait_tool_digital_input(index, val, timeout=None)..... | 222        |
| 6.1.15     | set_mode_analog_output(ch, mod).....                   | 224        |
| 6.1.16     | set_mode_analog_input(ch, mod).....                    | 225        |
| 6.1.17     | set_analog_output(ch, val).....                        | 226        |
| 6.1.18     | get_analog_input(ch).....                              | 227        |
| <b>6.2</b> | <b>TP Interface.....</b>                               | <b>228</b> |
| 6.2.1      | tp_popup(message, pm_type=DR_PM_MESSAGE, type=0).....  | 228        |
| 6.2.2      | tp_log(message).....                                   | 230        |
| 6.2.3      | tp_progress(cur_progress, total_progress).....         | 231        |
| 6.2.4      | tp_get_user_input(message, input_type).....            | 232        |
| <b>6.3</b> | <b>Thread.....</b>                                     | <b>234</b> |

---

|            |                                      |            |
|------------|--------------------------------------|------------|
| 6.3.1      | thread_run(th_func_name, loop=False) | 234        |
| 6.3.2      | thread_stop(th_id)                   | 236        |
| 6.3.3      | thread_pause(th_id)                  | 237        |
| 6.3.4      | thread_resume(th_id)                 | 238        |
| 6.3.5      | thread_state(th_id)                  | 239        |
| 6.3.6      | Integrated example                   | 240        |
| <b>6.4</b> | <b>Others</b>                        | <b>242</b> |
| 6.4.1      | wait(time)                           | 242        |
| 6.4.2      | exit()                               | 243        |
| 6.4.3      | sub_program_run(name)                | 244        |
| 6.4.4      | dri_report_line(option)              | 246        |
| 6.4.5      | set_fm(key, value)                   | 247        |

## **7. Mathematical Function ..... 248**

|      |             |     |
|------|-------------|-----|
| 7.1  | sin(x)      | 248 |
| 7.2  | cos(x)      | 249 |
| 7.3  | tan(x)      | 250 |
| 7.4  | asin(x)     | 251 |
| 7.5  | acos(x)     | 252 |
| 7.6  | atan(x)     | 253 |
| 7.7  | atan2(y, x) | 254 |
| 7.8  | ceil(x)     | 255 |
| 7.9  | floor(x)    | 256 |
| 7.10 | pow(x, y)   | 257 |
| 7.11 | sqrt(x)     | 258 |
| 7.12 | log(x, b)   | 259 |
| 7.13 | d2r(x)      | 260 |
| 7.14 | r2d(x)      | 261 |
| 7.15 | norm(x)     | 262 |
| 7.16 | random()    | 263 |

---

|      |   |     |
|------|---|-----|
| 7.17 | rotx(angle).....                              | 264 |
| 7.18 | roty(angle).....                              | 265 |
| 7.19 | rotz(angle).....                              | 266 |
| 7.20 | rotm2eul(rotm).....                           | 267 |
| 7.21 | rotm2rotvec(rotm).....                        | 268 |
| 7.22 | eul2rotm([alpha,beta,gamma]).....             | 269 |
| 7.23 | eul2rotvec([alpha,beta,gamma]).....           | 270 |
| 7.24 | rotvec2eul([rx,ry,rz]).....                   | 271 |
| 7.25 | rotvec2rotm([rx,ry,rz]).....                  | 272 |
| 7.26 | htrans(posx1,posx2).....                      | 273 |
| 7.27 | get_intermediate_pose(posx1,posx2,alpha)..... | 274 |
| 7.28 | get_distance(posx1, posx2).....               | 275 |
| 7.29 | get_normal(x1, x2, x3).....                   | 276 |
| 7.30 | add_pose(posx1,posx2).....                    | 277 |
| 7.31 | subtract_pose(posx1,posx2).....               | 278 |
| 7.32 | inverse_pose(posx1).....                      | 279 |
| 7.33 | dot_pose(posx1, posx2).....                   | 280 |
| 7.34 | cross_pose(posx1, posx2).....                 | 281 |
| 7.35 | unit_pose(posx1).....                         | 282 |

## 8. External Communication Commands ..... 283

|       |   |     |
|-------|---|-----|
| 8.1   | Serial.....   | 283 |
| 8.1.1 | serial_open(port=None, baudrate=115200, bytesize=DR_EIGHTBITS,<br>parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)..... | 283 |
| 8.1.2 | serial_close(ser).....  | 285 |
| 8.1.3 | serial_state(ser).....  | 286 |
| 8.1.4 | serial_set_inter_byte_timeout(ser, timeout=None).....   | 287 |
| 8.1.5 | serial_write(ser, tx_data).....   | 288 |
| 8.1.6 | serial_read(ser, length=-1, timeout=-1).....  | 289 |
| 8.1.7 | serial_get_count().....   | 291 |
| 8.1.8 | serial_get_info(id).....  | 292 |



---

|            |   |            |
|------------|---|------------|
| 8.1.9      | Combined Example.....   | 293        |
| <b>8.2</b> | <b>Tcp/Client.....</b>  | <b>294</b> |
| 8.2.1      | client_socket_open(ip, port).....   | 294        |
| 8.2.2      | client_socket_close(sock).....  | 295        |
| 8.2.3      | client_socket_state(sock).....  | 296        |
| 8.2.4      | client_socket_write(sock, tx_data).....   | 297        |
| 8.2.5      | client_socket_read(sock, length=1, timeout=1).....  | 298        |
| 8.2.6      | Integrated example.....   | 300        |
| <b>8.3</b> | <b>Tcp/Server.....</b>  | <b>302</b> |
| 8.3.1      | server_socket_open(port).....   | 302        |
| 8.3.2      | server_socket_close(sock).....  | 303        |
| 8.3.3      | server_socket_state(sock).....  | 304        |
| 8.3.4      | server_socket_write(sock, tx_data).....   | 305        |
| 8.3.5      | server_socket_read(sock, length=1, timeout=1).....  | 307        |
| 8.3.6      | Integrated example.....   | 309        |
| <b>8.4</b> | <b>Modbus.....</b>  | <b>311</b> |
| 8.4.1      | add_modbus_signal(ip, port, name, reg_type, index, value=0, slaveid=255).....   | 311        |
| 8.4.2      | add_modbus_rtu_signal(slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name, reg_type, index, value=0)..... | 314        |
| 8.4.3      | del_modbus_signal(name).....  | 316        |
| 8.4.4      | set_modbus_output(jobus, val).....  | 317        |
| 8.4.5      | set_modbus_outputs(jobus_list, val_list).....   | 318        |
| 8.4.6      | get_modbus_input(jobus).....  | 319        |
| 8.4.7      | get_modbus_inputs(jobus_list).....  | 320        |
| 8.4.8      | get_modbus_inputs_list(jobus_list).....   | 321        |
| 8.4.9      | wait_modbus_input(jobus, val, timeout=None).....  | 322        |
| 8.4.10     | set_modbus_slave(address, val).....   | 323        |
| 8.4.11     | get_modbus_slave(address).....  | 324        |
| 8.4.12     | modbus_crc16(data).....   | 325        |
| 8.4.13     | modbus_send_make(send_data).....  | 326        |
| 8.4.14     | modbus_recv_check(recv_data).....   | 327        |

---

|   |            |
|---|------------|
| <b>8.5 Industrial Ethernet (EtherNet/IP,PROFINET) .....</b>           | <b>328</b> |
| 8.5.1 set_output_register_bit(address, val).....                      | 328        |
| 8.5.2 set_output_register_int(address, val).....                      | 329        |
| 8.5.3 set_output_register_float(address, val).....                    | 330        |
| 8.5.4 get_output_register_bit(address).....                           | 331        |
| 8.5.5 get_output_register_int(address).....                           | 332        |
| 8.5.6 get_output_register_float(address).....                         | 333        |
| 8.5.7 get_input_register_bit(address).....                            | 334        |
| 8.5.8 get_input_register_int(address).....                            | 335        |
| 8.5.9 get_input_register_float(address).....                          | 336        |
| <br>  |            |
| <b>9. External Vision Commands .....</b>                              | <b>337</b> |
| Overview .....  | 337        |
| 9.1 vs_set_info(type) .....   | 338        |
| 9.2 vs_connect(ip_addr, port_num=9999) .....                          | 339        |
| 9.3 vs_disconnect() .....   | 340        |
| 9.4 vs_get_job().....   | 341        |
| 9.5 vs_set_job(job_name).....   | 342        |
| 9.6 vs_trigger().....   | 343        |
| 9.7 vs_set_init_pos(vision_posx_init, robot_posx_init, vs_pos=1)..... | 344        |
| 9.8 vs_get_offset_pos(vision_posx_meas, vs_pos=1) .....               | 345        |
| 9.9 vs_request(cmd).....  | 346        |
| 9.10 vs_result() .....  | 347        |
| 9.11 Integrated example 1 (DR_VS_COGNEX, DR_VS_SICK).....             | 348        |
| 9.12 Integrated Example 2.....  | 349        |
| <br>  |            |
| <b>10. Doosan Vision(SVM) Command.....</b>                            | <b>351</b> |
| 10.1 svm_connect (ip= "192.168.137.5", port=20).....                  | 351        |
| 10.2 svm_disconnect ().....   | 352        |
| 10.3 svm_set_job (job_id).....  | 353        |

---

|       |   |     |
|-------|---|-----|
| 10.4  | svm_get_robot_pose (job_id)                           | 354 |
| 10.5  | svm_get_vision_info (job_id)                          | 355 |
| 10.6  | svm_get_variable (tool_id, var_type)                  | 356 |
| 10.7  | svm_get_offset_pos (posx_robot_init, job_id, tool_id) | 358 |
| 10.8  | svm_set_init_pos_data(ld_list, Pos_list)              | 359 |
| 10.9  | svm_set_tp_popup (svm_flag)                           | 360 |
| 10.10 | svm_set_led_brightness(value)                         | 361 |
| 10.11 | svm_get_led_brightness()                              | 362 |
| 10.12 | svm_set_camera_exp_val(value)                         | 363 |
| 10.13 | svm_set_camera_gain_val(value)                        | 364 |
| 10.14 | svm_set_camera_load(job_id)                           | 365 |
| 10.15 | Intergrated example (SVM)                             | 366 |

## 11. Application Commands ..... 367

|        |  |     |
|--------|--|-----|
| 11.1   | External Encoder Setting Commands  | 367 |
| 11.1.1 | set_extenc_polarity(channel, polarity_A, polarity_B, polarity_Z, polarity_S)           | 367 |
| 11.1.2 | set_extenc_mode(channel, mode_AB, pulse_AZ, mode_Z, mode_S, inverse_cnt)               | 368 |
| 11.1.3 | get_extenc_count(channel)  | 370 |
| 11.1.4 | dear_extenc_count(channel)   | 371 |
| 11.2   | Conveyor Tracking  | 372 |
| 11.2.1 | set_conveyor(name)   | 372 |
| 11.2.2 | set_conveyor_ex  | 373 |
| 11.2.3 | get_conveyor_obj(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None) | 377 |
| 11.2.4 | tracking_conveyor(conv_id)   | 381 |
| 11.2.5 | untracking_conveyor(conv_id, time=None)  | 383 |

## 12. A-Series Command ..... 385

|      |   |     |
|------|---|-----|
| 12.1 | get_function_input(index)   | 385 |
| 12.2 | flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE) | 386 |

---

|   |     |
|---|-----|
| 12.3 flange_serial_close().....         | 388 |
| 12.4 flange_serial_write(tx_data) ..... | 389 |
| 12.5 flange_serial_read().....          | 390 |

## • DRL Basic Syntax

| No.  | Function              | Description  |
|------|-----------------------|--|
| 1.1  | <b>Indent</b>         | This function is used to separate each code block.   |
| 1.2  | <b>Comment</b>        | This function is used to provide an additional description of the code. The comments do not affect the source code since they are excluded from code processing. |
| 1.3  | <b>Variable name</b>  | This function is used to express the data value and can consist of letters, numbers, and underscores (_). (The first character cannot be a number.)              |
| 1.4  | <b>Numeric value</b>  | int, float, complex  |
| 1.5  | <b>String</b>         | String   |
| 1.6  | <b>list</b>           | This function lists the result values and recognizes the sequence.   |
| 1.7  | <b>tuple</b>          | This function is similar to a list but is faster at processing since it is read-only.  |
| 1.8  | <b>dictionary</b>     | This function specifies the keys and values and lists the values.  |
| 1.9  | <b>Function</b>       | This function is used to obtain a value by inputting the function name.  |
| 1.10 | Scoping rule          | If there is no value corresponding to the local variable name in a function, the name can be found based on the LGB rule.  |
| 1.11 | <b>Parameter mode</b> | This function uses default parameter values, keyword parameters, and variable parameters.  |
| 1.12 | <b>pass</b>           | This function is used when an operation is not executed.   |
| 1.13 | <b>if</b>             | This function is a conditional function that can use "elif" and "else" according to whether the condition of the "if" syntax is true or false.                   |
| 1.14 | <b>while</b>          | This function is a conditional function that repeats an operation according to whether the condition is true or false.   |
| 1.15 | <b>for</b>            | This function repeats an operation within the specified repeating "range".   |
| 1.16 | <b>break</b>          | This function exits the internal block of a loop.  |
| 1.17 | <b>continue</b>       | This function stops further executing the internal block of a loop and returns to the beginning point of the loop.   |
| 1.18 | <b>Else in a loop</b> | The "else" block is executed when the loop is executed until the end without being terminated by the "break" function in the middle of executing a loop.         |

## • Motion-related Commands

| No.  | Function  | Description   |
|------|---|---|
| 2.1  | <b>posj</b> (q1=0, q2=0, q3=0, q4=0, q5=0, q6=0)    | This function designates the joint space angle in coordinate values.  |
| 2.2  | <b>posx</b> (x=0, y=0, z=0, w=0, p=0, r=0)          | This function designates the task space in coordinate values.   |
| 2.3  | <b>trans</b> (pos, delta, ref, ref_out)             | This function is equivalent to "addto" in the task space and returns the value after homogeneous transformation as much as the delta of an object in the task space.  |
| 2.4  | <b>posb</b> (seg_type, posx1, posx2=None, radius=0) | 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. |
| 2.5  | <b>fkin</b> (pos, ref)                              | This function receives the input data of points or equivalent forms (float[6]) in joint space and returns the coordinates of the TCP (objects in the task space).   |
| 2.6  | <b>ikin</b> (pos, sol_space, ref)                   | This function returns to the joint location corresponding to sol_space, which is equivalent to the robot pose in the task space, among 8 joints.  |
| 2.7  | <b>addto</b> (pos, add_val=None)                    | This function creates a new posj object by adding add_val to each joint value of posj.  |
| 2.8  | <b>set_velj</b> (vel)                               | This function sets the global velocity in joint motion (movej, movejx, amovej, or amovejx) after using this command.  |
| 2.9  | <b>set_accj</b> (acc)                               | This function sets the global velocity in joint motion (movej, movejx, amovej, or amovejx) after using this command.  |
| 2.10 | <b>set_velx</b> (vel1, vel2)                        | This function sets the velocity of the task space motion globally.  |
| 2.11 | <b>set_velx</b> (vel)                               | This function sets the velocity of the task space motion globally.  |
| 2.12 | <b>set_accx</b> (acc1, acc2)                        | This function sets the acceleration of the task space motion globally.  |
| 2.13 | <b>set_accx</b> (acc)                               | This function calls the name of the TCP registered in the Teach Pendant and sets it as the current TCP.   |
| 2.14 | <b>set_tcp</b> (name)                               | This function calls the name of the TCP registered in the Teach Pendant and sets it as the current TCP.   |
| 2.15 | <b>set_ref_coord</b> (coord)                        | This function sets the reference coordinate system.   |
| 2.16 | <b>movej</b>  | The robot moves to the target joint position (pos) from the current joint   |

| No.  | Function             | Description  |
|------|----------------------|--|
|      |                      | position.  |
| 2.17 | <b>moveI</b>         | The robot moves along the straight line to the target position (pos) within the task space.  |
| 2.18 | <b>movejx</b>        | The robot moves to the target position (pos) within the joint space.   |
| 2.19 | <b>movec</b>         | 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.   |
| 2.20 | <b>movesj</b>        | 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.                |
| 2.21 | <b>movesx</b>        | 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.                 |
| 2.22 | <b>moveb</b>         | 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.                           |
| 2.23 | <b>move_spiral</b>   | The radius increases in a radial direction and the robot moves in parallel with the rotating spiral motion in an axial direction.  |
| 2.24 | <b>move_periodic</b> | 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. |
| 2.25 | <b>move_home</b>     | The robot performs homing to move the mechanical or user-defined home position.  |
| 2.26 | <b>amovej</b>        | The asynchronous movej motion operates in the same way as movej except that it does not have the radius parameter for blending.  |
| 2.27 | <b>amovel</b>        | The asynchronous moveI motion operates in the same way as moveI except that it does not have the radius parameter for blending.  |
| 2.28 | <b>amovejx</b>       | The asynchronous movejx motion operates in the same way as movejx except that it does not have the radius parameter for blending.  |
| 2.29 | <b>amovec</b>        | The asynchronous movec motion operates in the same way as movec except that it does not have the radius parameter for blending.  |
| 2.30 | <b>amovesj</b>       | The asynchronous movesj motion operates in the same way as movesj() except for the asynchronous processing.  |
| 2.31 | <b>amovesx</b>       | The asynchronous movesx motion operates in the same way as movesx() except for the asynchronous processing.  |

| No.  | Function   | Description   |
|------|--|---|
| 2.32 | <b>amoveb</b>  | 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.  |
| 2.33 | <b>amove_spiral</b>  | 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.                                  |
| 2.34 | <b>amove_periodic</b>  | 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.                              |
| 2.35 | <b>mwait(time=0)</b>   | This function sets the waiting time between the previous motion command and the motion command in the next line.  |
| 2.36 | <b>begin_blend(radius=0)</b>                                     | This function begins the blending section.  |
| 2.37 | <b>end_blend()</b>   | This function ends the blending section.  |
| 2.38 | <b>check_motion()</b>  | This function checks the status of the currently active motion.   |
| 2.39 | <b>stop(st_mode)</b>   | 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. |
| 2.40 | <b>change_operation_speed(speed)</b>                             | This function adjusts the operation velocity.   |
| 2.41 | <b>wait_manual_guide()</b>                                       | This function enables the user to perform hand during the execution of the program.   |
| 2.42 | <b>wait_nudge()</b>  | This function enables users to resume the execution of the program through the user's nudge input (applying external force to the robot) when the execution of the program is paused.                           |
| 2.43 | <b>enable_alter_motion(n,mode,ref,limit_dPOS,limit_dPOS_per)</b> | This function activates alter motion.   |
| 2.44 | <b>alter_motion([x,y,z,rx,ry,rz])</b>                            | This function excutes alter motion as input pose.   |
| 2.45 | <b>disable_alter_motion()</b>                                    | This function deactivates alter motion.   |

### • Auxiliary Control Commands

| No. | Function                  | Description                                     |
|-----|---------------------------|---|
| 3.1 | <b>get_control_mode()</b> | This function returns the current control mode. |



| No.  | Function                                   | Description  |
|------|--|--|
| 3.2  | <b>get_control_space()</b>                 | This function returns the current control space.   |
| 3.3  | <b>get_current_posj()</b>                  | This function returns the current joint angle.   |
| 3.4  | <b>get_current_velj()</b>                  | This function returns the current joint velocity.  |
| 3.5  | <b>get_desired_posj()</b>                  | This function returns the current target joint angle.  |
| 3.6  | <b>get_desired_velj()</b>                  | This function returns the current target joint velocity.   |
| 3.7  | <b>get_current_posx(ref)</b>               | This function returns the pose and solution space of the current coordinate system.                  |
| 3.8  | <b>get_current_tool_flange_posx(ref)</b>   | This function returns the pose of the current tool flange.   |
| 3.9  | <b>get_current_velx(ref)</b>               | This function returns the current tool velocity.   |
| 3.10 | <b>get_desired_posx(ref)</b>               | This function returns the target pose of the current tool.   |
| 3.11 | <b>get_desired_velx(ref)</b>               | This function returns the target velocity of the current tool.                                       |
| 3.12 | <b>get_current_solution_space()</b>        | This function returns the current solution space value.  |
| 3.13 | <b>get_current_rotm(ref)</b>               | This function returns the direction and matrix of the current tool.                                  |
| 3.14 | <b>get_joint_torque()</b>                  | This function returns the sensor torque value of the current joint.                                  |
| 3.15 | <b>get_external_torque()</b>               | This function returns the torque value generated by the external force on each current joint.        |
| 3.16 | <b>get_tool_force(ref)</b>                 | This function returns the external force applied to the current tool.                                |
| 3.17 | <b>get_solution_space(pos)</b>             | This function obtains the solution space value.  |
| 3.18 | <b>get_orientation_error(xd, xc, axis)</b> | This function returns the orientation error value between the arbitrary poses xd and xc of the axis. |

#### • Other Settings and Safety-related Commands

|     |                                 |  |
|-----|---------------------------------|--|
| 4.1 | <b>get_workpiece_weight()</b>   | This function measures and returns the weight of the workpiece.  |
| 4.2 | <b>reset_workpiece_weight()</b> | This function initializes the weight data of the material to initialize the algorithm before measuring the weight of the material. |

|     |                                    |  |
|-----|------------------------------------|--|
| 4.3 | <b>set_tool(name)</b>              | This function activates the tool of the entered name among the tool information registered in the Teach Pendant.                         |
| 4.4 | <b>set_tool(name)</b>              | This function activates the tool shape information of the entered name among the tool shape information registered in the Teach Pendant. |
| 4.5 | <b>set_singular_handling(mode)</b> | In case of path deviation due to the effect of singularity in task motion, user can select the response policy                           |

### • Force/Stiffness Control and Other User-Friendly Features

| No.  | Function  | Description  |
|------|---|--|
| 5.1  | <b>parallel_axis(x1, x2, x3, axis, ref)</b>               | This function matches the axis received as the argument axis in the normal vector and tool frame obtained by get_normal (x1, x2, x3).                                    |
| 5.2  | <b>parallel_axis(vect, axis, ref)</b>                     | This function matches the axis received as the argument axis in the tool frame in the given vect direction.  |
| 5.3  | <b>align_axis(x1, x2, x3, pos, axis, ref)</b>             | This function matches the axis received as the argument axis in the normal vector and tool frame obtained by get_normal (x1, x2, x3).                                    |
| 5.4  | <b>align_axis(vect, pos, axis, ref)</b>                   | This function matches the axis received as the argument axis in the tool frame in the given vect direction.  |
| 5.5  | <b>is_done_bolt_tightening(m=0, timeout=0, axis=None)</b> | This function monitors the tightening torque of the tool and returns True if the set torque (m) is reached within the given time and False if the given time has passed. |
| 5.6  | <b>release_compliance_ctrl()</b>                          | This function terminates compliance control and begins position control at the current position.   |
| 5.7  | <b>task_compliance_ctrl</b>                               | This function begins task compliance control based on the preset reference coordinate system.  |
| 5.8  | <b>set_stiffnessx</b>                                     | This function sets the stiffness value.  |
| 5.9  | <b>calc_coord</b>   | The user can calculate the new cartesian coordinate system using up to 4 input poses (x1~x4), input mode (mod) and the reference coordinate system (ref).                |
| 5.10 | <b>set_user_cart_coord</b>                                | The user can set the new cartesian coordinate system using position (pos) and reference coordinate system (ref).   |
| 5.11 | <b>set_user_cart_coord</b>                                | The user can set the new cartesian coordinate system using x1,   |

| No.  | Function  | Description  |
|------|---|--|
|      |   | x2, and x3.  |
| 5.12 | <b>set_user_cart_coord</b>                        | The user can set the new cartesian coordinate system using u1 and v1.  |
| 5.13 | <b>overwrite_user_cart_coord</b>                  | This function changes the position (pos) and reference coordinate system (ref) of the preset cartesian coordinate system with the requested ID (id). |
| 5.14 | <b>get_user_cart_coord</b>                        | The user can get the position and reference coordinate system of the cartesian coordinate system with the requested ID (id).                         |
| 5.15 | <b>set_desired_force</b>                          | This function sets the force control target value, force control direction, force target value, and variation time.                                  |
| 5.16 | <b>release_force</b> (time=0)                     | This function reduces the force control target value to 0 through the time value and returns the task space to adaptive control.                     |
| 5.17 | <b>check_position_condition</b>                   | This function checks the status of the given position.   |
| 5.18 | <b>check_force_condition</b>                      | This function checks the status of the given force. It disregards the force direction and only compares the sizes.                                   |
| 5.19 | <b>check_orientation_condition</b>                | This function checks the difference between the current pose and the specified pose of the robot end effector.                                       |
| 5.20 | <b>check_orientation_condition</b>                | This function checks the difference between the current pose and the rotating angle range of the robot end effector.                                 |
| 5.21 | <b>coord_transform</b> (pose_in, ref_in, ref_out) | This function transforms given task position expressed in reference coordinate.  |

### • System Commands

| No.   | Function   | Description  |
|-------|--|--|
| 6.1.1 | <b>set_digital_output</b> (index, val =None)         | This function sends a signal at the digital contact point of the controller.   |
| 6.1.2 | <b>set_digital_outputs</b> (bit_list)                | This function sends a signal at multiple digital output contact points of the controller.  |
| 6.1.3 | <b>set_digital_outputs</b> (bit_start, bit_end, val) | This function sends multiple signals at once from the digital output start contact point (bit_start) to the end contact point (bit_end) of the controller. |
| 6.1.4 | <b>get_digital_input</b> (index)                     | This function reads the signals from digital contact points of the controller and reads the digital input contact value.                                   |

| No.    | Function  | Description  |
|--------|---|--|
| 6.1.5  | <b>get_digital_inputs</b> (bit_list)                      | This function reads the signals from multiple digital contact points of the controller.  |
| 6.1.6  | <b>get_digital_inputs</b> (bit_start, bit_end)            | This function reads multiple signals at once from the digital input start contact point (start_index) to the end contact point (end_index) of the controller.  |
| 6.1.7  | <b>wait_digital_input</b> (index, val, timeout=None)      | This function waits until the signal value of the digital input register of the controller becomes val (ON or OFF).  |
| 6.1.8  | <b>set_tool_digital_output</b> (index, val=None)          | This function sends the signal of the robot tool from the digital contact point.   |
| 6.1.9  | <b>set_tool_digital_outputs</b> (bit_list)                | This function sends the signal of the robot tool from the digital contact point. The digital signals of the contact points defined in bit_list are output at one.  |
| 6.1.10 | <b>set_tool_digital_outputs</b> (bit_start, bit_end, val) | This function sends the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (bit_start) to the last contact point (bit_end) are output at one.    |
| 6.1.11 | <b>get_tool_digital_input</b> (index)                     | This function reads the signal of the robot tool from the digital contact point.   |
| 6.1.12 | <b>get_tool_digital_inputs</b> (bit_list)                 | This function reads the signal of the robot tool from the digital contact point. The digital signals of the contact points defined in bit_list are input at one.   |
| 6.1.13 | <b>get_tool_digital_inputs</b> (bit_start, bit_end)       | This function reads the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (start_index) to the last contact point (end_index) are input at one. |
| 6.1.14 | <b>wait_tool_digital_input</b> (index, val, timeout=None) | This function waits until the digital input signal value of the robot tool becomes val (ON or OFF).  |
| 6.1.15 | <b>set_mode_analog_output</b> (ch, mod)                   | This function sets the channel mode of the controller analog output.   |
| 6.1.16 | <b>set_mode_analog_input</b> (ch, mod)                    | This function sets the channel mode of the controller analog input.  |
| 6.1.17 | <b>set_analog_output</b> (ch, val)                        | This function outputs the channel value corresponding to the controller analog output.   |
| 6.1.18 | <b>get_analog_input</b> (ch)                              | This function reads the channel value corresponding to the controller analog input.  |

| No.   | Function   | Description   |
|-------|--|---|
| 6.2.1 | <b>tp_popup</b> (message, pm_type=DR_PM_MESSAGE, type=0) | This function provides a message to users through the Teach Pendant.  |
| 6.2.2 | <b>tp_log</b> (message)                                  | This function records the user-written log to the Teach Pendant.  |
| 6.2.3 | <b>tp_progress</b> (cur_progress, total_progress)        | This function provides a message to users through the Teach Pendant.  |
| 6.2.4 | <b>tp_get_user_input</b> (message, input_type)           | This function receives the user input data through the Teach Pendant.   |
| 6.3.1 | <b>thread_run</b> (th_func_name, loop=False)             | This function creates and executes a thread. The features executed by the thread are determined by the functions specified in th_func_name. |
| 6.3.2 | <b>thread_stop</b> (th_id)                               | This function terminates a thread.  |
| 6.3.3 | <b>thread_pause</b> (th_id)                              | This function temporarily suspends a thread.  |
| 6.3.4 | <b>thread_resume</b> (th_id)                             | This function resumes a temporarily suspended thread.   |
| 6.3.5 | <b>thread_state</b> (th_id)                              | This function checks the status of a thread.  |
| 6.3.6 | <b>Integrated example</b>                                | This example explains how to use the thread.  |
| 6.4.1 | <b>wait</b> (time)                                       | This function waits for the specified time.   |
| 6.4.2 | <b>exit</b> ()   | This function terminates the currently running program.   |
| 6.4.3 | <b>sub_program_run</b> (name)                            | It executes a subprogram saved as a separate file.  |

#### • Mathematical Function

| No. | Function        | Description  |
|-----|-----------------|--|
| 7.1 | <b>sin</b> (x)  | This function returns the sine value of x radians.       |
| 7.2 | <b>cos</b> (x)  | This function returns the cosine value of x radians.     |
| 7.3 | <b>tan</b> (x)  | This function returns the tangent value of x radians.    |
| 7.4 | <b>asin</b> (x) | This function returns the arc sine value of x radians.   |
| 7.5 | <b>acos</b> (x) | This function returns the arc cosine value of x radians. |

| No.  | Function                              | Description   |
|------|---------------------------------------|---|
| 7.6  | <b>atan(x)</b>                        | This function returns the arc tangent value of x radians.   |
| 7.7  | <b>atan2(y, x)</b>                    | This function returns the arc tangent value of y/x radians.   |
| 7.8  | <b>ceil(x)</b>                        | This function returns the smallest integer value of integers larger than x.   |
| 7.9  | <b>floor(x)</b>                       | This function returns the largest integer value of integers smaller than x.   |
| 7.10 | <b>pow(x, y)</b>                      | Return x raised to the power of y.  |
| 7.11 | <b>sqrt(x)</b>                        | This function returns the square root of x.   |
| 7.12 | <b>log(x, b)</b>                      | This function returns the log of x with base b.   |
| 7.13 | <b>d2r(x)</b>                         | This function returns the x radians value to degrees.   |
| 7.14 | <b>r2d(x)</b>                         | This function returns the x radians value to degrees.   |
| 7.15 | <b>norm(x)</b>                        | This function returns the L2 norm of x.   |
| 7.16 | <b>random()</b>                       | This function returns a random number.  |
| 7.17 | <b>rotx(angle)</b>                    | This function returns a rotation matrix that rotates by the angle value along the x-axis.                                 |
| 7.18 | <b>roty(angle)</b>                    | This function returns a rotation matrix that rotates by the angle value along the y-axis.                                 |
| 7.19 | <b>rotz(angle)</b>                    | This function returns a rotation matrix that rotates by the angle value along the z-axis.                                 |
| 7.20 | <b>rotm2eul(rotm)</b>                 | This function receives a rotation matrix and returns the Euler angle (zyz order) to degrees.                              |
| 7.21 | <b>rotm2rotvec(rotm)</b>              | This function receives a rotation matrix and returns the rotation vector (angle/axis representation).                     |
| 7.22 | <b>eul2rotm([alpha,beta,gamma])</b>   | This function transforms a Euler angle (zyz order) to a rotation matrix.  |
| 7.23 | <b>eul2rotvec([alpha,beta,gamma])</b> | This function transforms a Euler angle (zyz order) to a rotation vector.  |
| 7.24 | <b>rotvec2eul([rx,ry,rz])</b>         | This function transforms a rotation vector to a Euler angle (zyz).  |
| 7.25 | <b>rotvec2rotm([rx,ry,rz])</b>        | This function transforms a rotation vector to a rotation matrix.  |
| 7.26 | <b>htrans(posx1,posx2)</b>            | This function returns the pose corresponding to T1*T2 assuming that the homogeneous transformation matrices obtained from |

| No.  | Function   | Description  |
|------|--|--|
|      |  | posx1 and posx2 are T1 and T2, respectively.   |
| 7.27 | <b>get_intermediate_pose</b> (posx1, posx2, alpha) | This function returns posx located at alpha of the linear transition from posx1 to posx2.  |
| 7.28 | <b>get_distance</b> (posx1, posx2)                 | This function returns the distance between two pose positions in [mm].   |
| 7.29 | <b>get_normal</b> (x1, x2, x3)                     | This function returns the normal vector of a surface consisting of three points (posx) in the task space. This direction is clockwise. |
| 7.30 | <b>add_pose</b> (posx1, posx2)                     | This function obtains the sum of two poses.  |
| 7.31 | <b>subtract_pose</b> (posx1, posx2)                | This function obtains the difference between two poses.  |
| 7.32 | <b>inverse_pose</b> (posx1)                        | This function returns posx corresponding to the inverse of the posx.   |
| 7.33 | <b>dot_pose</b> (posx1, posx2)                     | This function obtains the inner product of the translation component when two poses are given.   |
| 7.34 | <b>cross_pose</b> (posx1, posx2)                   | This function obtains the outer product of the translation component when two poses are given.   |
| 7.35 | <b>unit_pose</b> (posx1)                           | This function obtains the unit vector of the given posx translation component.   |

### • External Communication Commands

| No.   | Function  | Description   |
|-------|---|---|
| 8.1.1 | <b>serial_open</b> (port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE) | This function opens a serial communication port.  |
| 8.1.2 | <b>serial_close</b> (ser)   | This function closes a serial communication port.   |
| 8.1.3 | <b>serial_state</b> (ser)   | This function returns the status of a serial communication port.                                    |
| 8.1.4 | <b>serial_set_inter_byte_timeout</b> (ser, timeout=None)  | This function sets the timeout between the bytes (inter-byte) when reading and writing to the port. |
| 8.1.5 | <b>serial_set_inter_byte_timeout</b> (ser, timeout=None)  | This function records the data (tx_data) to a serial port.  |
| 8.1.6 | <b>serial_read</b> (ser, length=-1,   | This function reads the data from a serial port.  |

| No.   | Function   | Description   |
|-------|--|---|
|       | timeout=-1)  |   |
| 8.1.7 | <b>serial_get_count</b>  | This function reads the number of devices connected to USB to Serial.   |
| 8.1.8 | <b>serial_get_info</b> (id)  | This function reads the port information and device name of the connected USB to Serial.  |
| 8.1.9 | <b>Combined Example</b>  | This is an example for performing a self-loop-back test on RXD (#2 pin) and TXD (#3 pin) are connected with the serial port.                        |
| 8.2.1 | <b>client_socket_open</b> (ip, port)   | This function creates a socket and attempts to connect it to a server (ip, port).   |
| 8.2.2 | <b>client_socket_close</b> (sock)  | This function terminates communication with the server.   |
| 8.2.3 | <b>client_socket_state</b> (sock)  | This function returns the socket status.  |
| 8.2.4 | <b>client_socket_write</b> (sock, tx_data)   | This function transmits data to the server.   |
| 8.2.5 | <b>client_socket_read</b> (sock, length=1, timeout=-1)   | This function reads data from the server.   |
| 8.3.1 | <b>server_socket_open</b> (port)   | This function creates a socket and waits for the connection to the client. This function returns the connected socket when the client is connected. |
| 8.3.2 | <b>server_socket_close</b> (sock)  | This function terminates communication with the client.   |
| 8.3.3 | <b>server_socket_state</b> (sock)  | This function returns the socket status.  |
| 8.3.4 | <b>server_socket_write</b> (sock, tx_data)   | This function transmits data to the client.   |
| 8.3.5 | <b>server_socket_read</b> (sock, length=-1, timeout=-1)  | This function reads data from the client.   |
| 8.4.1 | <b>add_modbus_signal</b> (ip, port, name, reg_type, index, value=0, slaveid=255)                                   | This function registers the Modbus signal.  |
| 8.4.2 | <b>add_modbus_rtu_signal</b> (slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, | This function registers the ModbusRTU signal.   |



| No.    | Function   | Description  |
|--------|--|--|
|        | stopbits=DR_STOPBITS_ONE,<br>name, reg_type, index, value=0) |  |
| 8.4.3  | <b>del_modbus_signal</b> (name)                              | This function deletes the registered Modbus signal.  |
| 8.4.4  | <b>set_modbus_output</b> (iobus, val)                        | This function sends the signal to an external Modbus system.   |
| 8.4.5  | <b>set_modbus_outputs</b> (iobus_list,<br>val_list)          | This function sends the multiple signals to the digital contact points of the external Modbus digital I/O unit.                        |
| 8.4.6  | <b>get_modbus_input</b> (iobus)                              | This function reads the signal from the Modbus system.   |
| 8.4.7  | <b>get_modbus_inputs</b> (iobus_list)                        | This function reads multiple signals from the digital input contact points of the external Modbus digital I/O unit.                    |
| 8.4.8  | <b>get_modbus_inputs_list</b> (iobus_list<br>)               | It is the command for reading multiple register type open signals from an external Modbus Slave unit.                                  |
| 8.4.9  | <b>wait_modbus_input</b> (iobus, val,<br>timeout=None)       | This function waits until the specified signal contact point value of the Modbus digital I/O becomes val (ON or OFF).                  |
| 8.4.10 | <b>set_modbus_slave</b> (address, val)                       | It is used to export values to the General Purpose Register area of the Modbus TCP Slave.  |
| 8.4.11 | <b>get_modbus_slave</b> (address)                            | It is used to import values by approaching the General Purpose Register area of the Modbus TCP Slave.                                  |
| 8.5.1  | <b>set_output_register_bit</b> (address,<br>val)             | It is used to export values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.   |
| 8.5.2  | <b>set_output_register_int</b> (address,<br>val)             | It is used to export values to the Output Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave    |
| 8.5.3  | <b>set_output_register_float</b> (address<br>, val)          | It is used to export values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave. |
| 8.5.4  | <b>get_output_register_bit</b> (address)                     | It is used to import values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.   |
| 8.5.5  | <b>get_output_register_int</b> (address)                     | It is used to import values to the Output Int General Purpose Register area of the Industrial  |

| No.   | Function                                   | Description  |
|-------|--|--|
|       |  | Ethernet(EtherNet/IP, PROFINET) Slave.   |
| 8.5.6 | <b>get_output_register_float</b> (address) | It is used to import values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave. |
| 8.5.7 | <b>get_input_register_bit</b> (address)    | It is used to import values to the Input Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.    |
| 8.5.8 | <b>get_input_register_int</b> (address)    | It is used to import values to the Input Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.    |
| 8.5.9 | <b>get_input_register_float</b> (address)  | It is used to import values to the Input Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.  |

#### • External Vision Commands

| No.  | Function   | Description   |
|------|--|---|
| 9.1  | <b>vs_set_info</b> (type)  | This function set the type of vision system to use.   |
| 9.2  | <b>vs_connect</b> (ip_addr, port_num=9999)                           | This function establishes communication with the vision system.                                       |
| 9.3  | <b>vs_disconnect</b>   | This function terminates the connection to the vision system.   |
| 9.4  | <b>vs_get_job</b>  | This function loaded the task name, currently loaded in the vision system.                            |
| 9.5  | <b>vs_set_job</b> (job_name)   | This function loaded the entered task into the vision system.   |
| 9.6  | <b>vs_trigger</b>  | This function transmits the measurement command to the vision system.                                 |
| 9.7  | <b>vs_set_init_pos</b> (vision_posx_init, robot_posx_init, vs_pos=1) | Enter the initial position information of the object to perform the vision guidance operation.        |
| 9.8  | <b>vs_get_offset_pos</b> (vision_posx_meas, vs_pos=1)                | The coordinate of the robot is calculated using the coordinate values, measured in the vision system. |
| 9.9  | <b>vs_request</b> (cmd)  | This function sets the feature for the vision system to request                                       |
| 9.10 | <b>vs_result</b>   | This function retrieves the processing result of the vision   |

| No. | Function | Description |
|-----|----------|-------------|
|     |          | system.     |

### • Doosan Vision(SVM) Commands

| No.   | Function   | Description  |
|-------|--|--|
| 10.1  | <b>svm_connect</b> (ip="192.168.137.5", port=20)             | This function establishes communication with the SVM.  |
| 10.2  | <b>svm_disconnect</b> ()                                     | This function terminates the connection to the SVM.  |
| 10.3  | <b>svm_set_job</b> (job_id)                                  | This function loads the Vision task corresponding to the input id into the SVM.  |
| 10.4  | <b>svm_get_robot_pose</b> (job_id)                           | The robot pose information(joint coordinate system) set in the vision task is loaded.  |
| 10.5  | <b>svm_get_vision_info</b> (job_id)                          | Performs the measurement command corresponding to the input vision task.   |
| 10.6  | <b>svm_get_variable</b> (tool_id, var_type)                  | If the object detection/measurement is successful(1) by executing svm_get_vision_info, the detection/measurement data is loaded. |
| 10.7  | <b>svm_get_offset_pos</b> (posx_robot_init, job_id, tool_id) | The robot task coordinate information reflecting the vision measurement.   |
| 10.8  | <b>svm_set_init_pos_data</b> (Id_list, Pos_list)             | Enter the initial id_list and posx_list information of the object to perform the vision guidance operation.                      |
| 10.9  | <b>svm_set_tp_popup</b> (svm_flag)                           | This function selects whether to display tp_popup when a vision error occurs.  |
| 10.10 | <b>svm_set_led_brightness</b> (value)                        | Change the Vision LED value to the set value.  |
| 10.11 | <b>svm_get_led_brihtness</b> ()                              | Vision LED The current value is loaded.  |
| 10.12 | <b>svm_set_camera_exp_val</b> (value)                        | Change the vision exposure value to the set value.   |
| 10.13 | <b>svm_set_camera_gain_val</b> (value)                       | Change the Vision gain value to the set value.   |
| 10.14 | <b>svm_set_camera_load</b> (job_id)                          | The LED / exp / gain / focus information stored in the vision is retrieved.  |

## Application Commands

| No.    | Function              | Description   |
|--------|-----------------------|---|
| 11.1.1 | 오류! 참조 원본을 찾을 수 없습니다. | It configures the polarity of phase A, B and the trigger method of phase S, Z of the corresponding encoder channel.   |
| 11.1.2 | 오류! 참조 원본을 찾을 수 없습니다. | It configures the operation mode of phase A, B, Z and S of the corresponding encoder channel.   |
| 11.1.3 | 오류! 참조 원본을 찾을 수 없습니다. | Get the count value of the corresponding encoder channel.   |
| 11.1.4 | 오류! 참조 원본을 찾을 수 없습니다. | Reset the corresponding channel encoder counter value to 0.   |
| 11.2.1 | set_conveyor(name)    | If conveyor information is configured in the UI, obtain the conveyor name as ID to start the Conveyor Tracking Application from the program and execute the command for workpiece monitoring. |
| 11.2.2 | 오류! 참조 원본을 찾을 수 없습니다. | Configures the conveyor and obtains Conveyor ID to allow the Conveyor Tracking Application to start.  |
| 11.2.3 | 오류! 참조 원본을 찾을 수 없습니다. | It returns the workpiece coordinate ID available for the job from the corresponding conveyor.   |
| 11.2.4 | 오류! 참조 원본을 찾을 수 없습니다. | The robot starts Conveyor Tracking.   |
| 11.2.5 | 오류! 참조 원본을 찾을 수 없습니다. | The robot ends Conveyor Tracking.   |

### • A-Series Commands

| No.  | Function  | Description  |
|------|---|--|
| 12.1 | Get_function_input(index)   | This function reads a state of the function button from the process button device. |
| 12.2 | flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE) | This command is for opening the pseudo flange serial communication port.           |
| 12.3 | flange_serial_close()오류! 참조 원본을 찾을 수 없습니다.  | This function closes a flange serial communication port.                           |
| 12.4 | flange_serial_write (tx_data)   | This function records the data (data) to a flange serial port                      |

---

| No.  | Function                             | Description   |
|------|--------------------------------------|---|
| 12.5 | flange_serial_read(timeout=Non<br>e) | This function reads the data from a flange serial port. |

# Preface

This manual is composed of twelve chapters. Chapter 1 through 11 describe DRL commands common to M-series robot and A-series robot, chapter 12 describes DRL commands that apply only to A-series robot.

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

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

## Copyright

The copyright and intellectual property rights of the contents of this manual are held by Doosan Robotics. It is therefore prohibited to use, copy, or distribute the contents without written approval from Doosan Robotics. In the event of abuse or modification of the patent right, the user will be fully accountable for the consequences.

While the information in this manual is reliable, Doosan Robotics will not be held accountable for any damage that occurs due to errors or typos. The contents of this manual may be modified according to product improvement without prior notification.

For details of updated manuals, refer to the Doosan Robotics website.

© 2018 Doosan Robotics Inc., All rights reserved

# 1. DRL Basic Syntax

## Caution

The syntax of DRL is the same as the syntax of Python which means that DRL does not include all the syntax and features of Python.

DRL only supports the information described in this manual.

## 1.1 Indent

### ▪ Features

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

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

### ▪ Example

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

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

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

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

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

## 1.2 Comment

### ▪ Features

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

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

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

### ▪ Example

```
# Comment example 1
""
    Comment example 2
""
```



## 1.3 Variable name

### ▪ Features

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

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

### Caution

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

|              |                |               |              |                 |
|--------------|----------------|---------------|--------------|-----------------|
| <b>and</b>   | <b>assert</b>  | <b>break</b>  | <b>class</b> | <b>continue</b> |
| <b>def</b>   | <b>del</b>     | <b>elif</b>   | <b>else</b>  | <b>except</b>   |
| <b>exec</b>  | <b>finally</b> | <b>for</b>    | <b>from</b>  | <b>global</b>   |
| <b>if</b>    | <b>import</b>  | <b>in</b>     | <b>is</b>    | <b>lambda</b>   |
| <b>list</b>  | <b>not</b>     | <b>open</b>   | <b>or</b>    | <b>pass</b>     |
| <b>print</b> | <b>raise</b>   | <b>return</b> | <b>try</b>   | <b>while</b>    |
| <b>with</b>  | <b>yield</b>   |               |              |                 |

### ▪ Example

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

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

### 1.4 Numeric value

- **Features**

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

- **Example**

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

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

## 1.5 String

### 1.5.1 String

#### ▪ Features

All character strings are in Unicode.

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

#### ▪ Example

```
"string1"
'string2'

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

```
#expected result: s  
tp_log("string"[1:3])  
#expected result: tr
```

### 1.5.2 +, \*

- **Example**

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

### 1.5.3 Indexing & slicing

- **Example**

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

## 1.6 list

### ▪ Features

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

### ▪ Example

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

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

### 1.7 tuple

- **Features**

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

- **Example**

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

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

### 1.8 dictionary

- **Features**

Dictionary specifies the keys and values and lists the values.

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

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

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

## 1.9 Function

### ▪ Features

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

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

### Caution

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

|              |                |               |              |                 |
|--------------|----------------|---------------|--------------|-----------------|
| <b>and</b>   | <b>assert</b>  | <b>break</b>  | <b>class</b> | <b>continue</b> |
| <b>def</b>   | <b>del</b>     | <b>elif</b>   | <b>else</b>  | <b>except</b>   |
| <b>exec</b>  | <b>finally</b> | <b>for</b>    | <b>from</b>  | <b>global</b>   |
| <b>if</b>    | <b>import</b>  | <b>in</b>     | <b>is</b>    | <b>lambda</b>   |
| <b>list</b>  | <b>not</b>     | <b>open</b>   | <b>or</b>    | <b>pass</b>     |
| <b>print</b> | <b>raise</b>   | <b>return</b> | <b>try</b>   | <b>while</b>    |
| <b>with</b>  | <b>yield</b>   |               |              |                 |

### ▪ sentence

```
def <function name> (parameter 1, parameter 2, ... parameter N):
    <syntax> ...
    return <return value>
```

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

fn_Times (10, 10)

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

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

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

### 1.10 Scoping rule

#### ▪ Features

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

#### ▪ Example

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

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

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



## 1.11 Parameter mode

- **Features**

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

- **Example**

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

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

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

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

- **Example - Default parameter value**

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

fn_Times(5)
```

- **Example - Keyword parameter**

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

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

- **Example - Variable parameter**

```
def fn_myUnion(*ar)
    .....
```

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

## 1.12 pass

- **Features**

The 'pass' is used when an operation is not executed.

- **Example**

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

## 1.13 if

### ▪ Features

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

### ▪ sentense

**if <conditional statement>:**

**<syntax>**

**if <conditional statement 1>:**

**<Syntax 1>**

**elif <conditional statement 2>:**

**<Syntax 2>**

**else:**

**<Syntax 3>**

### ▪ Example - if, elif, else

```
numbers = [2,5,7]
for number in numbers:
    if number%2==0:
        tp_log(str(number) + " is even")
    else:
        tp_log (str(number) + " is odd")
#expected result:
#2 is even
#5 is odd
#7 is odd
```

### 1.14 while

- **Features**

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

- **syntax**

```
while <conditional statement>:  
    <syntax>
```

- **Example**

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

## 1.15 for

- **Features**

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

- **syntax**

```
for <item> in <sequential object S>:
    <syntax>
```

- **Example**

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

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

## 1.16 break

- **Features**

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

- **Example**

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

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

### 1.17 continue

#### ▪ Features

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

#### ▪ Example

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

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

### 1.18 Else in a loop

#### ▪ Features

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

#### ▪ Example

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

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

## 2. Motion-related Commands

### 2.1 `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<br>list<br>posj | 0             | 1-axis angle or<br>angle list or<br>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()`

## 2.2 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<br>list<br>posx | 0             | z position or<br>position list or<br>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)
move1(x2, v=100, a=200)
    
```

### ▪ Related commands

move1()/movec()/movejx()/amove1()/amovec()/amovejx()



## 2.3 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 returns the value based on input parameter(pos)'s coordinate without ref\_out coordinate.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| pos            | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| delta          | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| ref            | int             | None          | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate : user defined</li> </ul> |
| ref_out        | int             | DR_BASE       | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• user coordinate : user defined</li> </ul>                                      |

### ▪ Return

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

### ▪ Exception

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

- **Example**

```
p0 = posj(0,0,90,0,90,0)
movej(p0, v=30, a=30)

x1 = posx(200, 200, 200, 0, 180, 0)
delta = [100, 100, 100, 0, 0, 0]
x2 = trans(x1, delta, DR_BASE, DR_BASE)

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()**

## 2.4 `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                                    |
|-----------------------|-------------------|---------------|--|
| <code>seg_type</code> | Int               | -             | <code>DR_LINE</code><br><code>DR_CIRCLE</code> |
| <code>posx1</code>    | <code>posx</code> | -             | 1 <sup>st</sup> task <code>posx</code>         |
| <code>posx2</code>    | <code>posx</code> | -             | 2 <sup>nd</sup> task <code>posx</code>         |
| <code>radius</code>   | float             | 0             | Blending radius [mm]                           |

### ▪ Return

`posb`

### ▪ Exception

| Exception                             | Description                        |
|---------------------------------------|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code> | 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()

## 2.5 fkin(pos, ref)

▪ **Features**

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

▪ **Parameters**

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| pos            | posj            | -             | posj or  |
|                | list (float[6]) |               | position list  |
| ref            | int             | DR_BASE       | reference coordinate<br>• DR_BASE : base coordinate<br>• DR_WORLD : world coordinate |

▪ **Return**

| Value | Description      |
|-------|------------------|
| posx  | Task space point |

▪ **Exception**

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

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
q1 = posj(0, 0, 90, 0, 90, 0)
movej(q1,v=10,a=20)
q2 = posj(30, 0, 90, 0, 90, 0)
x2 = fkin(q2, DR_WORLD)
# x2: Space coordinate at the edge of the robot (TCP) corresponding to joint value q2
movel(x2,v=100,a=200,ref=DR_WORLD) # Linear motion to x2
```

- **Related commands**

**set\_tcp()** # The tcp data of the name registered in the teach pendant (TP) is reflected during an fkin operation.

**posj()/posx()**

## 2.6 ikin(pos, sol\_space, ref)

### ▪ Features

This function returns the joint position corresponding to sol\_space, which is equivalent to the robot pose in the operating space, among 8 joint shapes.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| pos            | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| sol_space      | int             | -             | solution space   |
| ref            | int             | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> </ul> |

### ▪ Robot configuration 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 |

### ▪ Return

| Value | Description       |
|-------|-------------------|
| posj  | Joint space point |

### ▪ Exception

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

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

- **Example**

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

- **Related commands**

**set\_tcp()/posj()/posx()**

## 2.7 addto(pos, add\_val=None)

### ▪ Features

This function creates a new posj object by adding add\_val to each joint value of posj.

### ▪ Parameters

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

### ▪ Return

| Value | Description       |
|-------|-------------------|
| posj  | Joint space point |

### ▪ Exception

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

### ▪ Example

```
q1 = posj(10, 20, 30, 40, 50, 60)
movej (q1, v=10, a=20)
q2 = addto(q1, [0, 0, 0, 0, 45, 0])
movej (q2, v=10, a=20) # The robot moves to the joint (10, 20, 30, 40, 95, 60).
q3 = addto(q2, [])
q4 = addto(q3)
```

### ▪ Related commands

posj()



## 2.8 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()

## 2.9 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/sec2).
movej(Q2) # The joint motion acceleration to Q2 is 60(deg/sec2) which is the global acceleration.
movej(Q1, vel=20, acc=40) # The joint motion acceleration to Q1 is 40(deg/sec2) which is the specified
acceleration.
#2
set_accj(30.55)
set_accj([30, 40, 30, 30, 30, 10])
```

### ▪ Related commands

set\_velj()/movej()/movejx()/movesj()/amovej()/amovejx()/amovesj()

## 2.10 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()

## 2.11 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 `movej()`, `amovej()`, `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 |
|------------------|-----------|---------------|-------------|
| <code>vel</code> | float     | -             | velocity    |

### ▪ Return

| Value | Description |
|-------|-------------|
| 0     | Success     |

### ▪ Exception

| Exception                             | Description                        |
|---------------------------------------|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code> | 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)
movej(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/sec2). The global task
angular acceleration is automatically determined.
movej(P2) # The task motion linear velocity to P2 is 30(mm/sec) which is the global
velocity.
movej(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()/movej()/movec()/movesx()/moveb()/move_spiral()/amovej()/amovec()/`  
`amovesx()/amoveb()/amove_spiral()`

## 2.12 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()/amove  
sx()/amoveb()/amove_spiral()`

## 2.13 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 `movej()`, `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/sec2). The global task angular acceleration
                  # is automatically determined.
movel(P2)        # The task motion linear acceleration to P2 is 60(mm/sec2) which is the global
                  # acceleration.
movel(P1, vel=20, acc=40) # The task motion linear acceleration to P1 is 40(mm/sec2) which is the
                  # specified acceleration.
```

### ▪ Related commands

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

## 2.14 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()

## 2.15 set\_ref\_coord(coord)

### ▪ Features

This function sets the reference coordinate system.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| coord          | int       | -             | Reference coordinate system <ul style="list-style-type: none"> <li>• DR_BASE: Base Coordinate</li> <li>• DR_WORLD: World Coordinate</li> <li>• DR_TOOL: Tool Coordinate</li> <li>• user Coordinate: User defined</li> </ul> |

### ▪ 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)
x1 = posx(370.9, 419.7, 651.5, 90,-180,0)
movel(x1, v=100, a=100) # Base Coordinate basis
uu1 = [-1, 1, 0] # x-axis vector of the user coordinate system (base coordinate basis)
vv1 = [1, 1, 0] # y-axis vector of the user coordinate system (base coordinate basis)
pos = posx(370.9, -419.7, 651.5, 0, 0, 0) # Origin point of the user coordinate system
DR_USER1 = set_user_cart_coord(uu1, vv1, pos) # Sets the user coordinate
system.
set_ref_coord(DR_USER1) # Sets DR_USER1 of the user coordinate system to
the global coordinate system.
movel([0,0,0,0,0,0],v=100,a=100)# The global coordinate system is used if the
```



reference coordinate system is not specified.

# Moves to the origin point and direction of the

DR\_USER1 coordinate system.

`move([0,200,0,0,0,0],v=100,a=100)` # Moves to the (0,200,0) point of the DR\_USER1 coordinate system.

- **Related commands**

`fkin()/ikin()/move()/movejx()/movec()/movesx()/moveb()/amove()/amovejx()/`

`amovec()/movesx()/moveb()`

## 2.16 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            | -                  | posj or  |
|                | list (float[6]) |                    | joint angle list   |
| vel (v)        | float           | None               | velocity (same to all axes) or   |
|                | list (float[6]) | None               | velocity (to an axis)  |
| acc (a)        | float           | None               | acceleration (same to all axes) or   |
|                | list (float[6]) | None               | 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<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>                 |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br><ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul> |

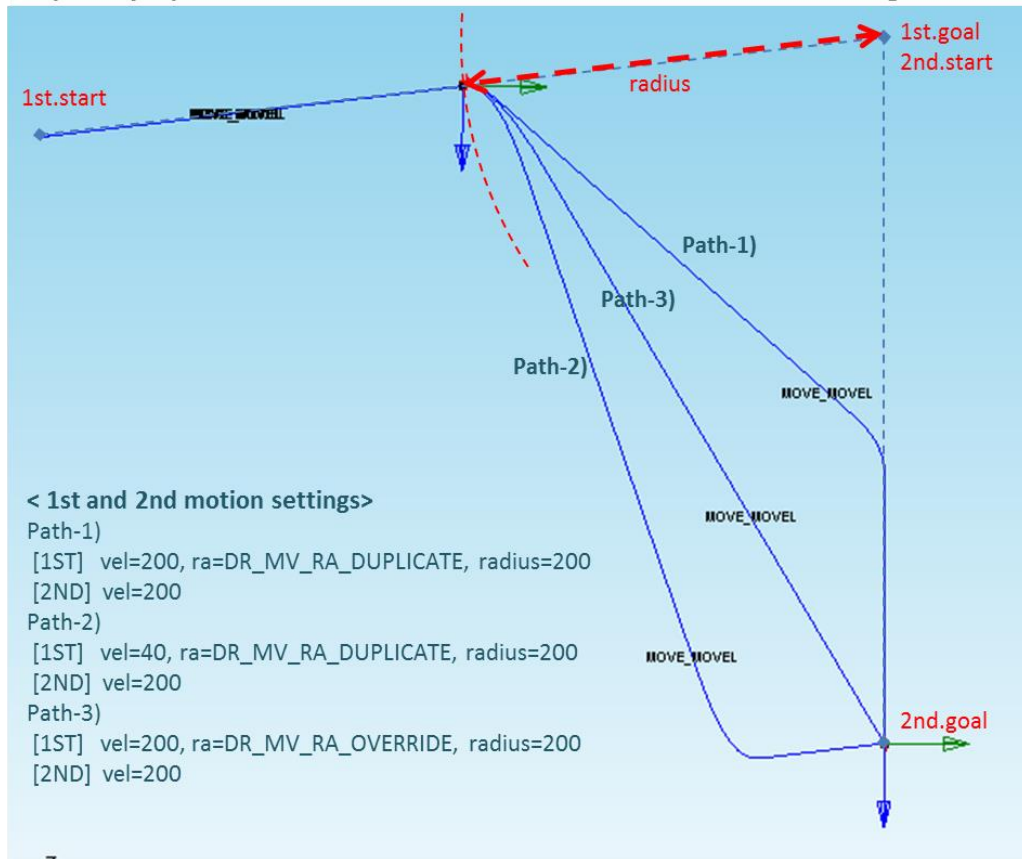
### Note

- 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()`

## 2.17 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            | -                  | posx or<br>position list  |
|                | list (float[6]) |                    |   |
| vel (v)        | float           | None               | velocity or<br>velocity1, velocity2   |
|                | list (float[2]) | None               |   |
| acc (a)        | float           | None               | acceleration or<br>acceleration1, acceleration2   |
|                | list (float[2]) | None               |   |
| time (t)       | float           | None               | Reach time [sec]<br>* 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<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br><ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul>  |

### Note

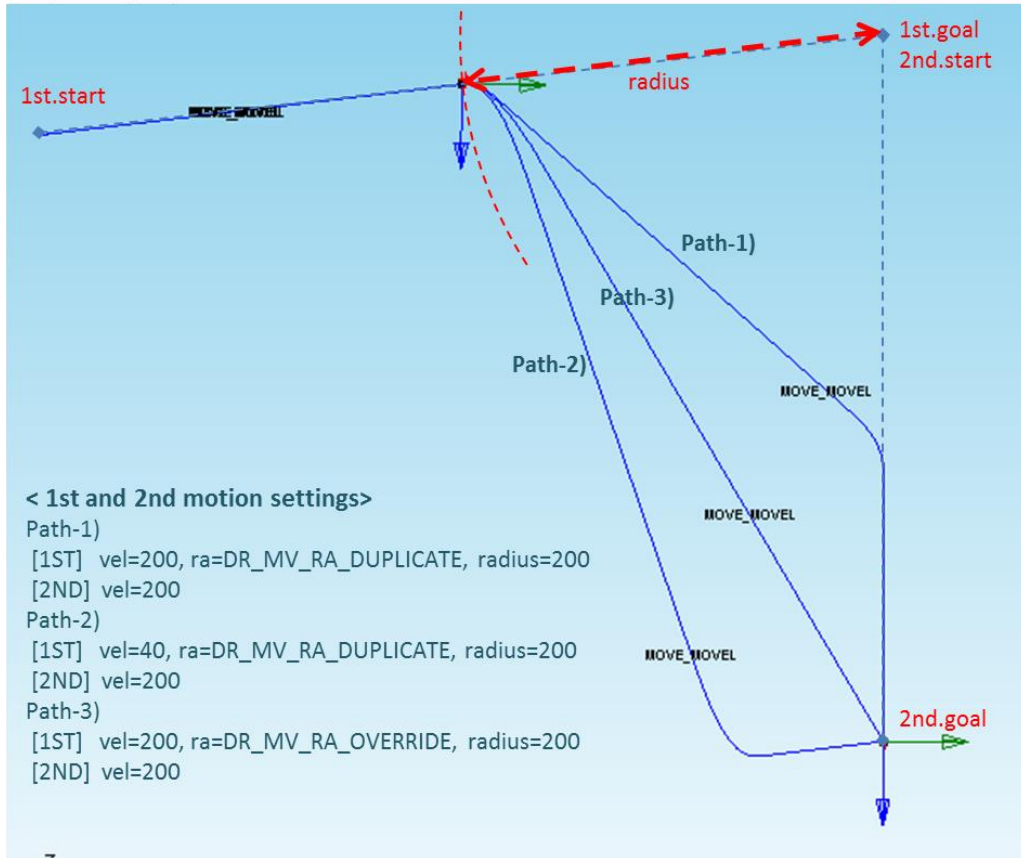
- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- `_global_velx` is applied if vel is None. (The initial value of `_global_velx` is 0.0 and can be set by `set_velx`.)

- `_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()`

## 2.18 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            | -                  | posx or<br>position list  |
|                | list (float[6]) |                    |   |
| vel (v)        | float           | None               | velocity (same to all axes) or<br>velocity (to an axis)   |
|                | list (float[6]) | None               |   |
| acc (a)        | float           | None               | acceleration (same to all axes) or<br>acceleration (acceleration to an axis)  |
|                | list (float[6]) | None               |   |
| time (t)       | float           | None               | Reach time [sec]  |
| radius (r)     | float           | None               | Radius for blending   |
| ref            | int             | None               | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br><ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul>  |
| 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/sec2), respectively,
# when the TCP edge is the P1 position (maintaining the solution space in the last P2
# position)
movejx(P2, time=5, sol=2)
# Moves to the joint angle with a reach time of 5 sec when the TCP edge is at the P2
# position
# (forcefully sets a solution space to 2)
movejx(P1, vel=[10, 20, 30, 40, 50, 60], acc=[20, 20, 30, 30, 40, 40], radius=100, sol=2)
# Moves the robot to the joint angle when the TCP edge is at the P1 position,
# and the next motion is executed when the distance from the P2 position is 100mm.
movejx(P2, v=30, a=60, ra= DR_MV_RA_OVERRIDE, sol=2)
# Immediately terminates the last motion and blends it to move to the joint angle
# when the TCP edge is at the P2 position.
```

▪ **Related commands**

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

2.19 **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            | -                  | posx or<br>position list   |
|                | list (float[6]) |                    |  |
| pos2           | posx            |                    | posx or<br>position list   |
|                | list (float[6]) |                    |  |
| vel (v)        | float           | None               | velocity or<br>velocity1, velocity2  |
|                | list (float[2]) | None               |  |
| acc (a)        | float           | None               | acceleration or<br>acceleration1, acceleration2  |
|                | list (float[2]) | None               |  |
| time (t)       | float           | None               | Reach time [sec]   |
| radius (r)     | float           | None               | Radius for blending  |
| ref            | int             | None               | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis <ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| angle (an)     | float           | None               | angle or<br>angle1, angle2   |
|                | list (float[2]) |                    |  |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul>  |

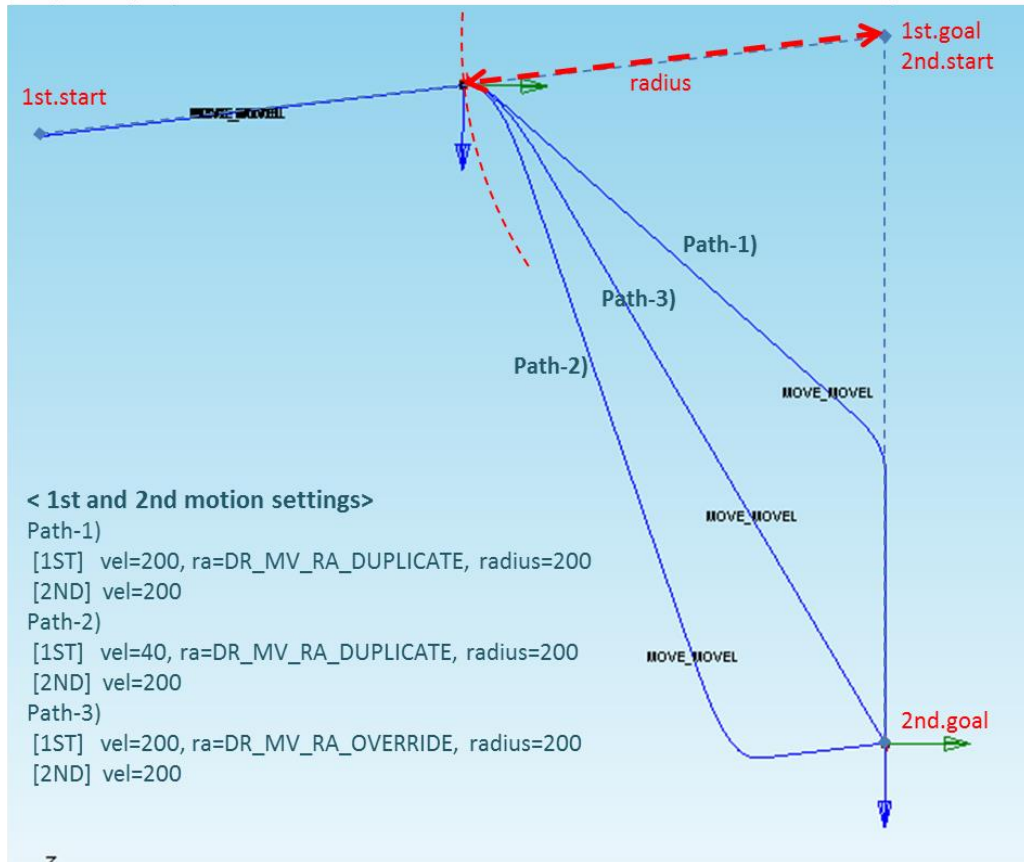
 **Note**

- 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/sec2) and 40(deg/sec2).

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

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

▪ **Related commands**

`posx()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/amovec()`

## 2.20 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  |
|-----------------------|-----------------|---------------|--|
| <code>pos_list</code> | list (posj)     | -             | posj list  |
| <code>vel (v)</code>  | float           | None          | velocity (same to all axes) or   |
|                       | list (float[6]) |               | velocity (to an axis)  |
| <code>acc (a)</code>  | float           | None          | acceleration (same to all axes) or   |
|                       | list (float[6]) |               | acceleration (acceleration to an axis)   |
| <code>time (t)</code> | float           | None          | Reach time [sec]   |
| <code>mod</code>      | int             | DR_MV_MOD_ABS | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul> |

### Note

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

▪ **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

▪ **Exception**

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

▪ **Example**

```
#CASE 1) Absolute angle input (mod= DR_MV_MOD_ABS)
q0 = posj(0,0,0,0,0,0)
movej(q0, vel=30, acc=60)      # Moves in joint motion to the initial position (q0).
q1 = posj(10, -10, 20, -30, 10, 20)      # Defines the posj variable (joint angle) q1.
q2 = posj(25, 0, 10, -50, 20, 40)
q3 = posj(50, 50, 50, 50, 50, 50)
q4 = posj(30, 10, 30, -20, 10, 60)
q5 = posj(20, 20, 40, 20, 0, 90)

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

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

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



```
dq5 = posj(-10, 10, 10, 40, -10, 30)      # Defines dq5 (q5=q4+dq5) as the relative
joint angle of q4

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

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

- **Related commands**

`posj()/set_velj()/set_accj()/amovesj()`

## 2.21 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           | None            | velocity or  |
|                | list (float[2]) |                 | velocity1, velocity2   |
| acc (a)        | float           | None            | acceleration or  |
|                | list (float[2]) |                 | acceleration1, acceleration2   |
| time (t)       | float           | None            | Reach time [sec]   |
| ref            | int             | None            | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS   | Movement basis <ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| vel_opt        | int             | DR_MVS_VEL_NONE | Velocity option <ul style="list-style-type: none"> <li>• DR_MVS_VEL_NONE: None</li> <li>• DR_MVS_VEL_CONST: Constant velocity</li> </ul>   |

 **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                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | Program terminated forcefully      |

## ▪ Example

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

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

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

```
#CASE 2) Relative coordinate input (mod= DR_MV_MOD_REL)
P0 = posj(0,0,90,0,90,0)
movej(P0)
x0 = posx(600, 43, 500, 0, 180, 0)      # Defines the posx variable (space
coordinate/pose) x0.
movel(x0, vel=100, acc=200)    # Linear movement to the initial position x0
dx1 = posx(0, 557, 100, 0, -5, 0)
    # Definition of relative coordinate dx1 to x0 (Homogeneous transformation of
dx1 based in x1= x0)
dx2 = posx(0, 150, 0, 0, 0, 0)
    # Definition of relative coordinate dx2 to x1 (Homogeneous transformation of dx2
based in x2= x1)
dx3 = posx(-450, -150, -150, 0, 0, 0)
    # Definition of relative coordinate dx3 to x2 (Homogeneous transformation of dx3
based in x3= x2)
dx4 = posx(-450, -300, -150, 0, 0, 0)
    # Definition of relative coordinate dx4 to x3 (Homogeneous transformation of dx4
based in x4= x3)
dx5 = posx(100, 400, 200, 0, 0, 0)
    # Definition of relative coordinate dx5 to x4 (Homogeneous transformation of dx5
based in x5= x4)
dx6 = posx(800, -100, -100, 0, 0, 0)
    # Definition of relative coordinate dx6 to x5 (Homogeneous transformation of dx6
based in x6= x5)
```

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

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

- **Related commands**

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

## 2.22 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           | None          | velocity or   |
|                | list (float[2]) |               | velocity1, velocity2  |
| acc (a)        | float           | None          | acceleration or   |
|                | list (float[2]) |               | acceleration1, acceleration2  |
| time (t)       | float           | None          | Reach time [sec]<br>* If the time is specified, values are processed based on time, ignoring vel and acc.   |
| ref            | int             | None          | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |

 **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()**

## 2.23 move\_spiral

### ▪ Features

Motion along a spiral trajectory on a plane which is perpendicular to the input 'axis' is performed on the specified coordinate system 'ref'. Additional input, travel distance 'lmax' can cause the robot to move around a cone, starting from the apex of it

### ▪ 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<br><ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>  |
| ref            | Int       | DR_TOOL       | -        | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: user defined</li> </ul> |

 **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       |

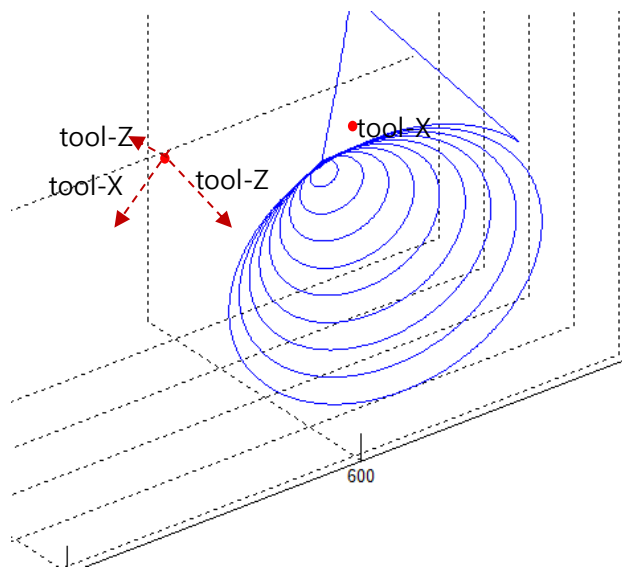
▪ **Exception**

| 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()
```

## 2.24 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])          | -             | $0 \leq \text{amp}$    | Amplitude (motion between -amp and +amp) [mm] or [deg]   |
| period         | float or list (float[6]) |               | $0 \leq \text{period}$ | Period (time for 1 cycle) [sec]  |
| atime          | float                    | 0.0           | $0 \leq \text{atime}$  | Acc-, dec- time [sec]  |
| repeat         | int                      | 1             | $> 0$                  | Repetition count   |
| ref            | int                      | DR_TOOL       | -                      | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: user defined</li> </ul> |

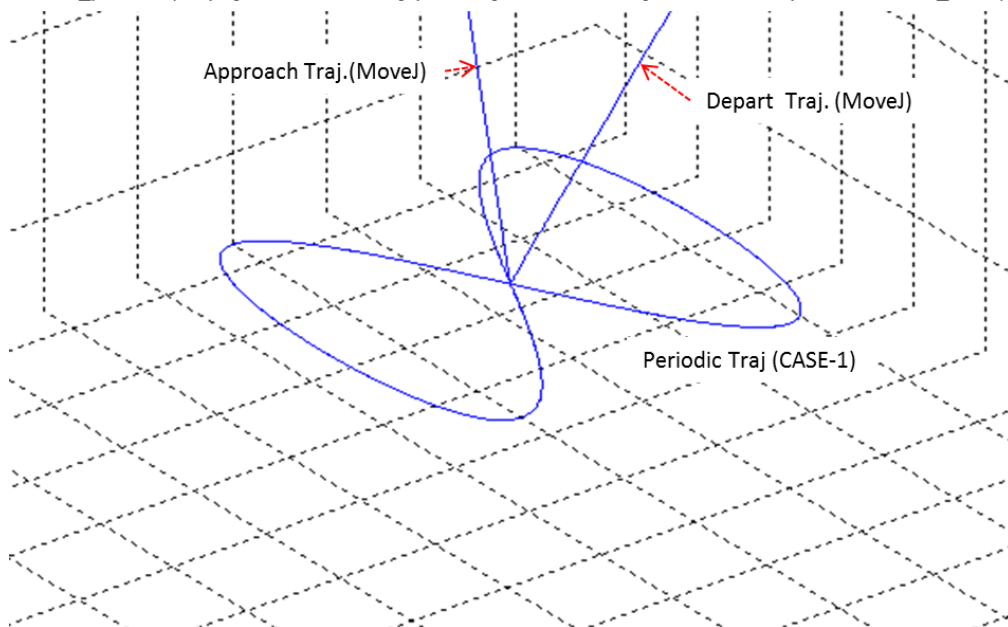
### Note

- 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.

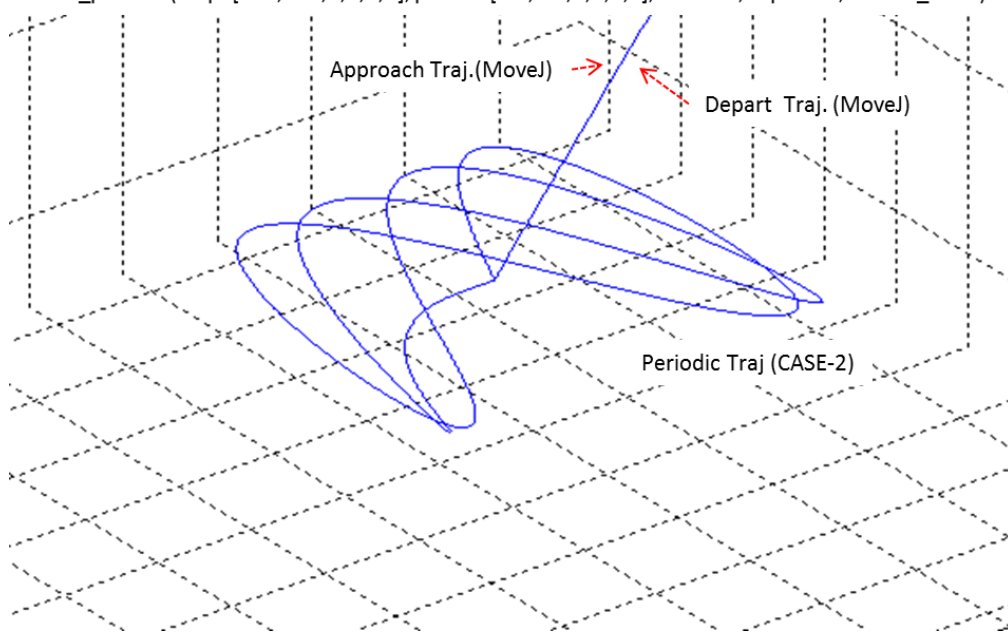
### CASE-1) All-axis motions end at the same time

```
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()

## 2.25 move\_home

### ▪ Features

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

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Range  | Description  |
|----------------|-----------|---------------|--------|--|
| target         | int       | -             | 0 or 1 | Target of home position<br>. DR_HOME_TARGET_MECHANIC ;<br>Mechanical home, joint angle<br>(0,0,0,0,0,0)<br>. DR_HOME_TARGET_USER :<br>user home. |

### Note

- Homing motion is divided into two steps and performed sequentially.
  - step1) Move to the homing position at the speed specified in the system
  - step2) Finding the home position precisely
- Safety should be ensured so that there is no danger of collision in the vicinity of homing operation..

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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



- **Example**

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

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

## 2.26 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            | -                  | posj or<br>joint angle list  |
|                | list (float[6]) |                    |  |
| vel (v)        | float           | None               | velocity (same to all axes) or<br>velocity (to an axis)  |
|                | list (float[6]) |                    |  |
| acc (a)        | float           | None               | acceleration (same to all axes) or<br>acceleration (acceleration to an axis)   |
|                | list (float[6]) |                    |  |
| time (t)       | float           | None               | Reach time [sec]   |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>                 |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br><ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul> |

 **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                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | 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()`

## 2.27 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            | -                  | posx or<br>position list   |
|                | list (float[6]) |                    |  |
| vel (v)        | float           | None               | velocity or<br>velocity1, velocity2  |
|                | list (float[2]) |                    |  |
| acc (a)        | float           | None               | acceleration or<br>acceleration1, acceleration2  |
|                | list (float[2]) |                    |  |
| time (t)       | float           | None               | Reach time [sec]<br>* If the time is specified, values are processed based on time, ignoring vel and acc.  |
| ref            | int             | None               | reference coordinate<br>• DR_BASE : base coordinate<br>• DR_WORLD : world coordinate<br>• DR_TOOL : tool coordinate<br>• user coordinate: User defined |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis<br>• DR_MV_MOD_ABS: Absolute<br>• DR_MV_MOD_REL: Relative   |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br>• DR_MV_RA_DUPLICATE: duplicate<br>• 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                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | 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()`

## 2.28 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            | -                  | posx or  |
|                | list (float[6]) |                    | position list  |
| vel (v)        | float           | None               | velocity (same to all axes) or   |
|                | list (float[6]) | None               | velocity (to an axis)  |
| acc (a)        | float           | None               | acceleration (same to all axes) or   |
|                | list (float[6]) | None               | acceleration (acceleration to an axis)   |
| time (t)       | float           | None               | Reach time [sec]   |
| ref            | int             | None               | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>   |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode<br><ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul>   |
| 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                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | 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()`

## 2.29 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            | -                  | posx or<br>position list   |
|                | list (float[6]) |                    |  |
| pos2           | posx            | -                  | posx or<br>position list   |
|                | list (float[6]) |                    |  |
| vel (v)        | float           | None               | velocity or<br>velocity1, velocity2  |
|                | list (float[2]) |                    |  |
| acc (a)        | float           | None               | acceleration or<br>acceleration1, acceleration2  |
|                | list (float[2]) |                    |  |
| time (t)       | float           | None               | Reach time [sec]   |
| ref            | int             | None               | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS      | Movement basis <ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| angle (an)     | float           | None               | angle or<br>angle1, angle2   |
|                | list (float[2]) |                    |  |
| ra             | int             | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> <li>• DR_MV_RA_DUPLICATE: duplicate</li> <li>• DR_MV_RA_OVERRIDE: override</li> </ul>  |

 **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  $\text{angle1} + 2 \times \text{angle2}$ .
- 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                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>  | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code> | 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()`

## 2.30 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           | None          | velocity (same to all axes) or   |
|                | list (float[6]) |               | velocity (to an axis)  |
| acc (a)        | float           | None          | acceleration (same to all axes) or                                       |
|                | list (float[6]) |               | acceleration (acceleration to an axis)                                   |
| time (t)       | float           | None          | Reach time [sec]   |
| mod            | int             | DR_MV_MOD_ABS | Movement basis<br>• DR_MV_MOD_ABS: Absolute<br>• 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()`



## 2.31 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           | None            | velocity or  |
|                | list (float[2]) |                 | velocity1, velocity2   |
| acc (a)        | float           | None            | acceleration or  |
|                | list (float[2]) |                 | acceleration1, acceleration2   |
| time (t)       | float           | None            | Reach time [sec]   |
| ref            | int             | None            | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS   | Movement basis <ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |
| vel_opt        | int             | DR_MVS_VEL_NONE | Velocity option <ul style="list-style-type: none"> <li>• DR_MVS_VEL_NONE: None</li> <li>• DR_MVS_VEL_CONST: Constant velocity</li> </ul>   |



**Note**

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

 **Caution**

The constant velocity motion according to the distance and velocity between the waypoints cannot be used if the "vel\_opt= DR\_MVS\_VEL\_CONST" option (constant velocity option) is selected, and the motion is automatically switched to the variable velocity motion (vel\_opt= DR\_MVS\_VEL\_NONE) in that case.

▪ **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

▪ **Exception**

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

- **Example**

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

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

amovesx(xlist, vel=[100, 30], acc=[200, 60], vel_opt=DR_MVS_VEL_NONE)
# Moves the spline curve that connects the waypoints defined in the xlist
# with a maximum velocity of 100, 30(mm/sec, deg/sec) and maximum
acceleration of 200(mm/sec2) and
# 60(deg/sec2). The next command is executed immediately after the motion
starts.
wait(3)      # Temporarily suspends the program execution for 3 seconds (while the
motion continues).
set_digital_output(1, 1)      # D-Out (no. 1 channel) ON
mwait(0)      # Temporarily suspends the program execution until
the motion is terminated.
```

- **Related commands**

**posx()/set\_vel()/set\_accx()/set\_tcp()/set\_ref\_coord()/mwait()/movesx()**

## 2.32 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           | None          | velocity or   |
|                | list (float[2]) |               | velocity1, velocity2  |
| acc (a)        | float           | None          | acceleration or   |
|                | list (float[2]) |               | acceleration1, acceleration2  |
| time (t)       | float           | None          | Reach time [sec]<br>* If the time is specified, values are processed based on time, ignoring vel and acc.   |
| ref            | int             | None          | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• DR_TOOL: tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS | Movement basis<br><ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>  |

 **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()**

## 2.33 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.

Motion along a spiral trajectory on a plane which is perpendicular to the input 'axis' is performed on the specified coordinate system 'ref'. Additional input, travel distance 'lmax' can cause the robot to move around a cone, starting from the apex of it

### 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 <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>  |
| ref            | Int       | DR_TOOL       | -        | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: user defined</li> </ul> |

### 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

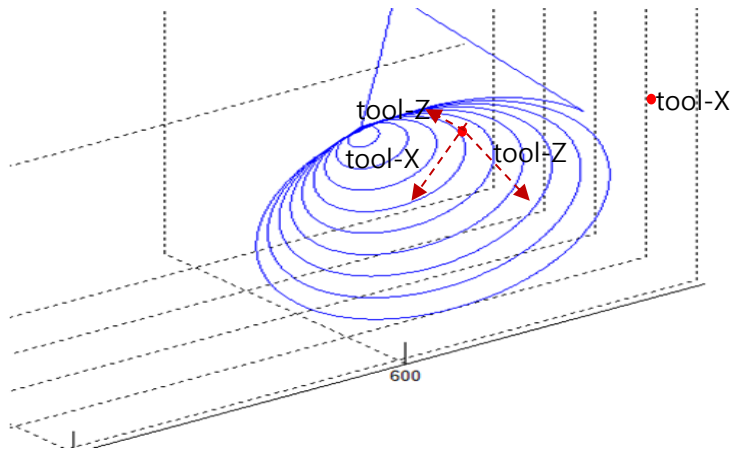
| Exception                   | Description                        |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE)   | Parameter value is invalid         |
| DR_Error (DR_ERROR_RUNTIME) | 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)
moveJ(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()`

## 2.34 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

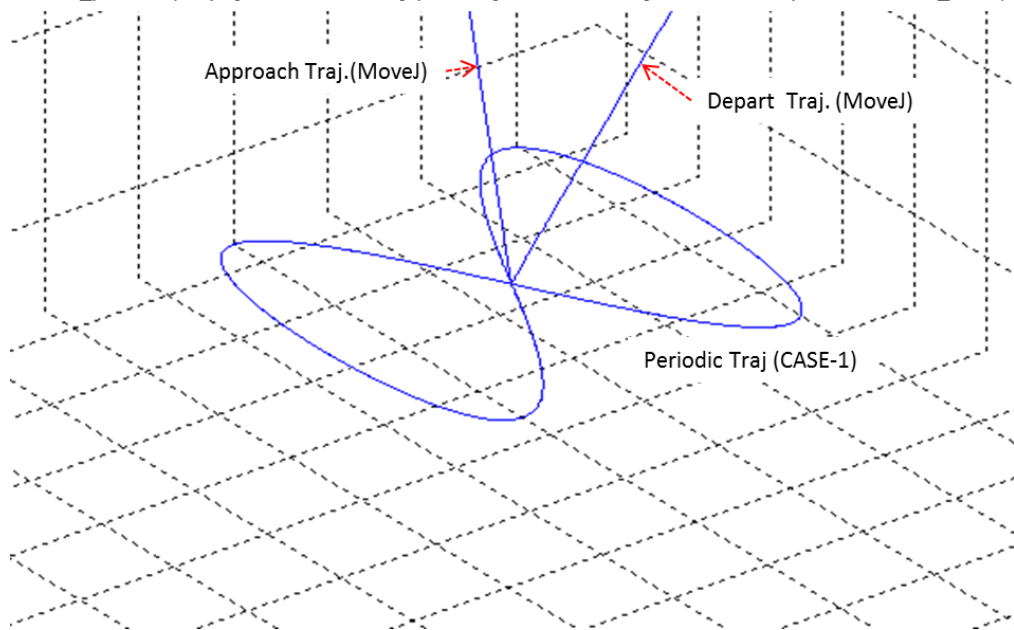
| Parameter Name | Data Type                | Default Value | Range                  | Description  |
|----------------|--------------------------|---------------|------------------------|--|
| amp            | list (float[6])          | -             | $0 \leq \text{amp}$    | Amplitude (motion between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

## 2.35 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()**

## 2.36 `begin_blend(radius=0)`

### ▪ Features

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

### ▪ Parameters

| Parameter Name      | Data Type | Default Value | Description         |
|---------------------|-----------|---------------|---------------------|
| <code>radius</code> | float     | 0             | Radius for blending |

### ▪ Return

| Value | Description |
|-------|-------------|
| 0     | Success     |

### ▪ Exception

| Exception                             | Description                        |
|---------------------------------------|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code> | Parameter data type error occurred |

## begin\_blend(radius=0)

---

### ▪ Example

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

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

### ▪ Related commands

end\_blend()/movej()/movei()/movejx()/movec()



## 2.37 end\_blend()

- **Features**

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

- **Parameters**

Not applicable

- **Return**

| Value | Description |
|-------|-------------|
| 0     | Success     |

- **Exception**

Not applicable

- **Example**

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

- **Related commands**

begin\_blend()/movej()/movei()/movejx()/movec()

## 2.38 check\_motion()

- **Features**

This function checks the status of the currently active motion.

- **Parameters**

Not applicable

- **Return (TBD)**

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

- **Exception**

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

- **Example**

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

- **Related commands**

movej()/moveI()/movejx()/movec()/movesj()/movesx()/moveb()/move\_spiral()  
/move\_periodic()/amovej()/amoveI()/amovejx()/amovec()/amovesj()/amovesx()/amoveb  
()/amove\_spiral()/amove\_periodic()

## 2.39 stop(st\_mode)

### ▪ Features

This stops the current motion. The stop time is determined based on the st\_mode, which is received as a factor, and it does not deviate from the current path.

This command is used to stop the robot's motion, but it does not function to stop program execution. To stop program execution, use the exit() command. Values DR\_QSTOP\_STO and DR\_QSTOP respond to Stop Category 1 (torque off after maximum deceleration) and 2 (maximum deceleration), but they are not linked with motions, such as torque off, after stopping. DR\_SSTOP stops over a deceleration time approximately 1.5 times longer than the maximum deceleration time. DR\_HOLD stops immediately without a deceleration phase.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| st_mode        | int       | -             | stop mode <ul style="list-style-type: none"> <li>• DR_QSTOP_STO: Quick stop (Stop Category 1 without STO(Safe Torque Off))</li> <li>• DR_QSTOP: Quick stop (Stop Category 2)</li> <li>• DR_SSTOP: Soft Stop</li> <li>• DR_HOLD: HOLE stop</li> </ul> |

### ▪ 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()
```

## 2.40 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\_riodic()/amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb()/amove\_spiral()/amove\_periodic()**

## 2.41 wait\_manual\_guide()

### ▪ Features

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

1) The user presses the "OK" or "Finish" button on the "Hand Guiding Execution" popup window generated from the TP.

2) A signal is applied to the digital input channel specified for "Manual guide release" in the safety I/O settings.

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

### ▪ Parameters

Not applicable

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

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

j00 = posj(0,0,90,0,90,0)
movej(j00,vel=20,acc=40) # Enters the collaborative workspace.
wait_manual_guide() # Direct teaching until the "Finish" button on the popup window
```

generated by the TP is pressed.

```
pos1 = get_current_posx() # Stores the directly taught point in pos1.
```

```
dposa = posx(0,0,-100,0,0,0)
```

```
movel(dposa, vel=300, acc=600, ref=DR_TOOL)
```

```
# Retract 100 mm in the tool-Z direction from the taught position.
```

### ▪ Related commands

`movej()/movel()/movejx()/movec()/movesj()/movesx()/moveb()/move_spiral()/move_riodic()/amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb()/amove_spiral()/amove_periodic()`



## 2.42 wait\_nudge()

### ▪ Features

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

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

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

### ▪ Parameters

Not applicable

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

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

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

- **Related commands**

movej()/movel()/movejx()/movec()/movesj()/movesx()/moveb()/move\_spiral()/move\_periodic()/amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb()/amove\_spiral()/amove\_periodic()

## 2.43 enable\_alter\_motion(n,mode,ref,limit\_dPOS,limit\_dPOS\_per)

### Features

enable\_alter\_motion() and alter\_motion() functions enable to alter motion trajectory. This function sets the configurations for altering function and allows the input quantity of alter\_motion() to be applied to motion trajectory. The unit cycle time of generating alter motion is 100msec. Cycle time( $n \times 100\text{msec}$ ) can be changed through input parameter n. This function provide 2 modes(Accumulation mode, Increment mode). Input quantity of alter\_motion() can be applied to motion trajectory in two ways as accumulated value or increment value. In accumulation mode, the input quantity means absolute altering amount(dX,dY,dZ,dRX,dRY,dRZ) from current motion trajectory. On the contrary in increment mode, the quantity means increment value from the previous absolute altering amount. The reference coordinate can be changed through input parameter ref. Limitations of accumulation amount and increment amount can be set through input parameter limit\_dPOS (accumulated limit) and limit\_dPOS\_per(increment input limit during 1 cycle). The actual alter amount is limited to these limits.

### Parameters

| Parameter Name | Data Type      | Default Value | Description  |
|----------------|----------------|---------------|--|
| n              | int            | None          | Cycle time number  |
| mode           | Int            | None          | Mode <ul style="list-style-type: none"> <li>DR_DPOS : accumulation amount</li> <li>DR_VEL : increment amount</li> </ul>  |
| ref            | int            | None          | reference coordinate <ul style="list-style-type: none"> <li>DR_BASE : base coordinate</li> <li>DR_WORLD : world coordinate</li> <li>DR_TOOL : tool coordinate</li> </ul> user coordinate: user defined |
| limit_dPOS     | list(float[2]) | None          | First value : limitation of position[mm]<br>Second value : limitation of orientation[deg]  |
| Limit_dPOS_per | list(float[2]) | None          | Fist value : limitation of position[mm]<br>Second value : limitation of orientation[deg]   |

### Note

- \_global\_ref is applied if ref is None
- Accumulation amount or increment amount isn't be limited if limit\_dPOS or limit\_dPOS\_per is None.
- alter\_motion() can be executed only in user thread.

▪ **Exception**

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

▪ **Example**

```
def alter_thread():
    alter_motion(dX) #dX : amount of alter motion

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

J00 = posj(0,0,90,0,90)
X1 = posx(559.0, 200, 651.5, 180, -180.0, 180)
X2 = posx(559.0, 200, 400, 180, -180.0, 180)

movej(J00,vel=50,acc=100)

enable_alter_motion(n=10,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[50,50])
# cycle time:(5*100)msec, mode:accumulate, reference coordination:base
coordination
# Lmitation of accumulation amount :50mm,50deg
# Limitation of increment amount :10mm, 10deg

th_id = thread_run(alter_thread, loop=True)

movel(X1,v=50,a=100,r=30)
movel(X2,v=50,a=100)

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

- **Related commands**

`alter_motion(pos)`, `disable_alter_motion()`

## 2.44 alter\_motion([x,y,z,rx,ry,rz])

### ▪ Features

This function applies altering amount of motion trajectory when the alter function is activated. The meaning of the input values is defined from enable\_alter\_motion().

#### Caution

- alter\_motion() can be executed only in user thread.

#### Note

- alter\_motion() can be executed only in user thread.
- Alter motion can be adjusted through setting value limit\_dPOS or limit\_dPOS\_per in enable\_alter\_motion function.
- Orientation of Input pose follows fixed XYZ notation.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description   |
|----------------|-----------------|---------------|---------------|
| pos            | list (float[6]) | None          | position list |

### ▪ Exception

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

### ▪ Example

```
def alter_thread():
    alter_motion(dX) #dX : amount of alter motion

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

J00 = posj(0,0,90,0,90)
```

```
X1 = posx(559.0, 200, 651.5, 180, -180.0, 180)
X2 = posx(559.0, 200, 400, 180, -180.0, 180)

movej(J00,vel=50,acc=100)

enable_alter_motion(n=5,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[10,10])
# cycle time:(5*100)msec, mode:accumulate, reference coordination:base
coordination
# Limitation of accumulation amount :50mm,90deg
# Limitation of increment amount :10mm, 10deg

th_id = thread_run(alter_thread, loop=True)

movel(X1,v=50,a=100,r=30)
movel(X2,v=50,a=100)

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

### ▪ Related commands

**enable\_alter\_motion(n,mode,ref,limit\_dPOS,limit\_dPOS\_per), disable\_alter\_motion()**

## 2.45 disable\_alter\_motion()

- **Features**

This function deactivates alter motion.

- **Exception**

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

- **Example**

```
def alter_thread():
    alter_motion(dX) #dX : amount of alter motion

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

J00 = posj(0,0,90,0,90)
X1 = posx(559.0, 200, 651.5, 180, -180.0, 180)
X2 = posx(559.0, 200, 400, 180, -180.0, 180)

movej(J00,vel=50,acc=100)

enable_alter_motion(n=10,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[50,50])
# cycle time:(5*100)msec, mode:accumulate, reference coordination:base
coordination
# Lmitation of accumulation amount :50mm,50deg
# Limitation of increment amount :10mm, 10deg

th_id = thread_run(alter_thread, loop=True)
```



```
move1(X1,v=50,a=100,r=30)
move1(X2,v=50,a=100)

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

- **Related commands**

`enable_alter_motion(n,mode,ref,limit_dPOS,limit_dPOS_per), alter_motion(pos)`

## 3. Auxiliary Control Commands

### 3.1 get\_control\_mode()

- **Features**

This function returns the current control mode.

- **Parameters**

Not applicable

- **Return**

| Value | Description  |
|-------|--|
| int   | Control mode<br>3 : Position control mode<br>4 : Torque control mode |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
mode = get_control_mode()
```

- **Related commands**

Not applicable

## 3.2 `get_control_space()`

- **Features**

This function returns the current control space.

- **Parameters**

Not applicable

- **Return**

| Value | Description   |
|-------|---|
| int   | Control mode<br>1 : Joint space control<br>2 : Task space control |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
x1 = get_control_space()
```

- **Related commands**

Not applicable

### 3.3 get\_current\_posj()

- **Features**

This function returns the current joint angle.

- **Parameters**

Not applicable

- **Return**

| Value | Description |
|-------|-------------|
| posj  | Joint angle |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
q1 = get_current_posj()
```

- **Related commands**

`get_desired_posj()`

## 3.4 `get_current_velj()`

- **Features**

This function returns the current joint velocity.

- **Parameters**

Not applicable

- **Return**

| Value    | Description |
|----------|-------------|
| float[6] | Joint speed |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
velj1 = get_current_velj()
```

- **Related commands**

`get_desired_velj()`

### 3.5 get\_desired\_posj()

- **Features**

This function returns the current target joint angle. It cannot be used in the movej, movec, movesx, moveb, move\_spiral, or move\_periodic command.

- **Parameters**

Not applicable

- **Return**

| Value | Description |
|-------|-------------|
| posj  | Joint angle |

- **Exception**

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

- **Example**

```
jp1 = get_desired_posj()
```

- **Related commands**

`get_current_posj()`

### 3.6 `get_desired_velj()`

- **Features**

This function returns the current target joint velocity. It cannot be used in the `move1`, `movec`, `movesx`, `moveb`, `move_spiral`, or `move_periodic` command.

- **Parameters**

Not applicable

- **Return**

| Value    | Description           |
|----------|-----------------------|
| float[6] | Target joint velocity |

- **Exception**

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

- **Example**

```
velj1 = get_desired_velj()
```

- **Related commands**

`get_current_velj()`

### 3.7 get\_current\_posx(ref)

#### ▪ Features

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

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| ref            | Int       | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• user coordinate: User defined</li> </ul> |

#### Note

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

#### ▪ Return

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

#### ▪ Robot configuration 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    |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |



- **Example**

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

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

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

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

- **Related commands**

**get\_desired\_posx()**

### 3.8 get\_current\_tool\_flange\_posx(ref)

▪ **Features**

This function returns the pose of the current tool flange based on the ref coordinate. In other words, it means the return to tcp=(0,0,0,0,0,0).

▪ **Parameters**

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| ref            | int       | DR_BASE       | reference coordinate<br>• DR_BASE : base coordinate<br>• DR_WORLD : world coordinate |

▪ **Return**

| Value | Description         |
|-------|---------------------|
| posx  | Pose of tool flange |

▪ **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

▪ **Example**

```
x1 = get_current_tool_flange_posx()
#x1 : Flange pose base on the base coordinate(default value)
x2 = get_current_tool_flange_posx(DR_BASE)
#x2 : Flange pose based on the base coordinate
x3 = get_current_tool_flange_posx(DR_WORLD)
#x3 : Flange pose based on the world coordinate
```

▪ **Related commands**

Not applicable

### 3.9 `get_current_velx(ref)`

#### ▪ Features

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

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| ref            | Int       | DR_BASE       | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> </ul> |

#### ▪ Return

| Value    | Description   |
|----------|---------------|
| float[6] | Tool velocity |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

#### ▪ Example

```

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

```

#### ▪ Related commands

`get_desired_velx()`

### 3.10 get\_desired\_posx(ref)

▪ **Features**

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

▪ **Parameters**

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| ref            | Int       | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• user coordinate: User defined</li> </ul> |

 **Note**

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

▪ **Return**

| Value    | Description   |
|----------|---------------|
| float[6] | Tool velocity |

▪ **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

▪ **Example**

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

xa = get_desired_posx(DR_USR1) #xa w.r.t. DR_USR1

xb = get_desired_posx(DR_WORLD) #xb w.r.t. DR_WORLD
```

▪ **Related commands**

get\_desired\_posx()

### 3.11 `get_desired_velx(ref)`

#### ▪ Features

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

#### ▪ Parameters

| 인수명 | 자료형 | 기본값     | 설명   |
|-----|-----|---------|--|
| ref | Int | DR_BASE | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> </ul> |

#### ▪ Return

| Value    | Description   |
|----------|---------------|
| float[6] | Tool velocity |

#### ▪ Exception

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

#### ▪ Example

```

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

```

#### ▪ Related commands

`get_current_velx()`

### 3.12 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 |

- **Example**

```
sol = get_current_solution_space()
```

- **Related commands**

`get_solution_space()`

### 3.13 `get_current_rotm(ref)`

#### ▪ Features

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

#### ▪ Parameters

| 인수명 | 자료형 | 기본값     | 설명   |
|-----|-----|---------|--|
| ref | Int | DR_BASE | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> </ul> |

#### ▪ Return

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

#### ▪ Example

```
rotm1 = get_current_rotm(DR_WORLD)
#rotm1 : rotation matrix(3x3) based on the world coordinate
# The result value is stored in a 3x3 matrix.
rotm1 =  $\begin{bmatrix} \text{rotm1}[0][0] & \text{rotm1}[0][1] & \text{rotm1}[0][2] \\ \text{rotm1}[1][0] & \text{rotm1}[1][1] & \text{rotm1}[1][2] \\ \text{rotm1}[2][0] & \text{rotm1}[2][1] & \text{rotm1}[2][2] \end{bmatrix}$ 
```

#### ▪ Related commands

Not applicable

### 3.14 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()`



### 3.15 `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()`

### 3.16 get\_tool\_force(ref)

▪ **Features**

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

▪ **Parameters**

| 인수명 | 자료형 | 기본값     | 설명   |
|-----|-----|---------|--|
| ref | Int | DR_BASE | reference coordinate<br>• DR_BASE : base coordinate<br>• DR_WORLD : world coordinate |

▪ **Return**

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

▪ **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

▪ **Example**

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

▪ **Related commands**

get\_joint\_torque()/get\_external\_torque()

### 3.17 `get_solution_space(pos)`

- **Features**

Calculates the solution space of the entered pos (posj).

- **Parameters**

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

- **Return**

| Value | Description    |
|-------|----------------|
| 0 ~ 7 | Solution space |

- **Exception**

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

- **Example**

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

- **Related commands**

`get_current_solution_space()`

### 3.18 get\_orientation\_error(xd, xc, axis)

▪ **Features**

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

▪ **Parameters**

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| xd             | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| xc             | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul> |

▪ **Return**

| Value | Description             |
|-------|-------------------------|
| float | Orientation error value |

▪ **Exception**

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

▪ **Example**

```

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

```

▪ **Related commands**

**get\_current\_rotm()**

## 4. Other Settings and Safety-related Commands

### 4.1 `get_workpiece_weight()`

- **Features**

This function measures and returns the weight of the workpiece.

- **Parameters**

Not applicable

- **Return**

| Value          | Description     |
|----------------|-----------------|
| Positive value | Measured weight |
| Negative value | Error           |

- **Exception**

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

- **Example**

```
weight = get_workpiece_weight()
```

- **Related commands**

`set_workpiece_weight()/reset_workpiece_weight()`

## 4.2 reset\_workpiece\_weight()

### ▪ Features

This function initializes the weight data of the material to initialize the algorithm before measuring the weight of the material.

### ▪ Parameters

Not applicable

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```
reset_workpiece_weight()
```

### ▪ Related commands

set\_workpiece\_weight()/get\_workpiece\_weight()

### 4.3 set\_tool(name)

#### ▪ Features

This function activates the tool of the entered name among the tool information registered in the Teach Pendant.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                               |
|----------------|-----------|---------------|---|
| name           | string    | -             | Tool name registered in the Teach Pendant |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

#### ▪ Exception

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

#### ▪ Example

```
set_tool ("tool1") # Activate the information of "tool1" registered in TP.
```

#### ▪ Related commands

set\_tcp()

## 4.4 set\_tool\_shape(name)

### ▪ Features

This function activates the tool shape information of the entered name among the tool shape information registered in the Teach Pendant.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                               |
|----------------|-----------|---------------|---|
| name           | string    | -             | Tool name registered in the Teach Pendant |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```
set_tool_shape("tool_shape1") # Activate the geometry of "tool_shape1".
```

### ▪ Related commands

set\_tcp()



## 4.5 set\_singularity\_handling(mode)

### ▪ Features

In case of path deviation due to the effect of singularity in task motion, user can select the response policy. The mode can be set as follows.

- Automatic avoidance mode(Default) : DR\_AVOID
- Path first mode : DR\_TASK\_STOP
- Variable velocity mode : DR\_VAR\_VEL

The default setting is automatic avoidance mode, which reduces instability caused by singularity, but reduces path tracking accuracy. In case of path first setting, if there is possibility of instability due to singularity, a warning message is output after deceleration and then the corresponding task is terminated. In case of variable velocity mode setting, TCP velocity would be changed in singular region to reduce instability and maintain path tracking accuracy.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| mode           | int       | DR_AVOID      | DR_AVOID : Automatic avoidance mode<br>DR_TASK_STOP : Deceleration/<br>Warning/ Task termination<br>DR_VAR_VEL : Variable velocity mode |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

- **Example**

```
.P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
P3 = posx(400,500,200,0,180,0)
set_singularity_handling(DR_AVOID) # Automatic avoidance mode for singularity
movel(P1, vel=10, acc=20)
set_velx(30)
set_accx(60)
set_singularity_handling(DR_TASK_STOP) # Task motion path first
movel(P2)
set_singularity_handling(DR_VAR_VEL) # Variable velocity mode for singularity
movel(P3)
```

- **Related commands**

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

## 5. Force/Stiffness Control and Other User-Friendly Features

### 5.1 `parallel_axis(x1, x2, x3, axis, ref)`

#### ▪ Features

This function matches the normal vector of the plane consists of points(x1, x2, x3) based on the ref coordinate(refer to `get_normal(x1, x2, x3)`) and the designated axis of the tool frame. The current position is maintained as the TCP position of the robot.

#### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description   |
|----------------|-----------------|---------------|---|
| x1             | posx            | -             | posx or   |
|                | list (float[6]) |               | position list   |
| x2             | posx            | -             | posx or   |
|                | list (float[6]) |               | position list   |
| x3             | posx            | -             | posx or   |
|                | list (float[6]) |               | position list   |
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>  |
| ref            | int             | DR_BASE       | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• user coordinate : user difined</li> </ul> |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

▪ **Exception**

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

▪ **Example**

```
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
x1 = posx(0, 500, 700, 30, 0, 90)
x2 = posx(500, 0, 700, 0, 0, 45)
x3 = posx(300, 100, 500, 45, 0, 45)
parallel_axis(x1, x2, x3, DR_AXIS_X, DR_WORLD)
# match the tool x axis and the normal vector of the plane consists of points(x1,x2,x3) #
based on the world coordinate
```

▪ **Related commands**

**get\_normal()/parallel\_axis()/align\_axis()/align\_axis()**

## 5.2 parallel\_axis(vect, axis, ref)

### ▪ Features

This function matches the given vect direction based on the ref coordinate and the designated axis of the tool frame. The current position is maintained as the TCP position of the robot.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| vect           | list (float[3]) | -             | vector   |
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>   |
| ref            | int             | DR_BASE       | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• user coordinate: user defined</li> </ul> |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
parallel_axis([1000, 700, 300], DR_AXIS_X, DR_WORLD)
# match the tool x axis and the vector([1000,700,300]) based on the world coordinate
```

`parallel_axis(vect, axis, ref)`

---

- **Related commands**

`parallel_axis()/align_axis()/align_axis()`

### 5.3 align\_axis(x1, x2, x3, pos, axis, ref)

#### ▪ Features

This function matches the normal vector of the plane consists of points(x1, x2, x3) based on the ref coordinate(refer to get\_normal(x1, x2, x3)) and the designated axis of the tool frame. The robot TCP moves to the pos position.

#### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| x1             | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| x2             | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| x3             | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| pos            | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>   |
| ref            | int             | DR_BASE       | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> <li>• user coordinate: user defined</li> </ul> |

#### ▪ 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(0,0,45,0,90,0)
movej(p0, v=30, a=30)

x1 = posx(0, 500, 700, 30, 0, 0)
x2 = posx(500, 0, 700, 0, 0, 0)
x3 = posx(300, 100, 500, 0, 0, 0)
pos = posx(400, 400, 500, 0, 0, 0)
align_axis(x1, x2, x3, pos, DR_AXIS_X, DR_BASE)
# match the tool x axis and the normal vector in the plane consists of points(x1, x2,
# x3) based on the base coordinate
```

▪ **Related commands**

`get_normal()/align_axis()/parallel_axis()/parallel_axis()`



## 5.4 align\_axis(vect, pos, axis, ref)

### ▪ Features

This function matches the given vect direction based on the ref coordinate and the designated axis of the tool frame. The robot TCP moves to the pos position.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| vect           | list (float[3]) | -             | vector   |
| pos            | posx            | -             | posx or  |
|                | list (float[6]) |               | position list  |
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul> |

### ▪ 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(0,0,45,0,90,0)
movej(p0, v=30, a=30)

vect = [10, 20, 30]
pos = posx(100, 500, 700, 45, 0, 0)
align_axis(vect, pos, DR_AXIS_X)
```

### ▪ Related commands

align\_axis()/parallel\_axis()/parallel\_axis()

## 5.5 is\_done\_bolt\_tightening(m=0, timeout=0, axis=None)

### ▪ Features

This function monitors the tightening torque of the tool and returns True if the set torque (m) is reached within the given time and False if the given time has passed.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| m              | float     | 0             | Target torque   |
| timeout        | float     | 0             | Monitoring duration [sec]   |
| axis           | int       | -             | axis<br><ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul> |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

| Exception                   | Description                        |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred |
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred  |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully      |

- **Example**

```
p0 = posj(0,0,90,0,90,0)
movej(p0, v=30, a=30)

task_compliance_ctrl()
xd = posx(559, 34.5, 651.5, 0, 180.0, 60)
amovel(xd, vel=50, acc=50) # Bolt tightening motion

res = is_done_bolt_tightening(10, 5, DR_AXIS_Z)
    # Returns True if the tightening torque of 10Nm is reached within 5 seconds.
    # Returns False otherwise.
if res==True:
    # some action comes here for the case that bolt tightening is done
    x=1
else:
    # some action comes here for the case that it fails
    x=2
```

- **Related commands**

**amovel()**

## 5.6 release\_compliance\_ctrl()

- **Features**

This function terminates compliance control and begins position control at the current position.

- **Parameters**

Not applicable

- **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

- **Exception**

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

- **Example**

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
task_compliance_ctrl()
set_stiffnessx([100, 100, 300, 100, 100, 100])
release_compliance_ctrl()
```

- **Related commands**

`task_compliance_ctrl()/set_stiffnessx()`

## 5.7 task\_compliance\_ctrl(stx, time)

### ▪ Features

This function begins task compliance control based on the preset reference coordinate system.

### ▪ Parameters (Stiffness TBD)

| Parameter Name | Data Type | Default Value                     | Description   |
|----------------|-----------|-----------------------------------|---|
| stx            | float[6]  | [3000, 3000, 3000, 200, 200, 200] | Three translational stiffnesses<br>Three rotational stiffnesses                                 |
| time           | float     | 0                                 | Stiffness varying time [sec]<br>Range: 0 - 1.0<br>* Linear transition during the specified time |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
task_compliance_ctrl()          # Begins with the default stiffness
set_stiffnessx([500, 500, 500, 100, 100, 100], time=0.5)
# Switches to the user-defined stiffness for 0.5 sec.
release_compliance_ctrl()

task_compliance_ctrl([500, 500, 500, 100, 100, 100])
# Begins with the user-defined stiffness.
release_compliance_ctrl()
```

`task_compliance_ctrl(stx, time)`

---

- **Related commands**  
`set_stiffnessx()/release_compliance_ctrl()`

## 5.8 set\_stiffnessx(stx, time)

### ▪ Features

This function sets the stiffness value based on the global coordinate(refer to set\_ref\_coord()). The linear transition from the current or default stiffness is performed during the time given as STX. The user-defined ranges of the translational stiffness and rotational stiffness are 0-20000N/m and 0-400Nm/rad, respectively.

### ▪ Parameters

| Parameter Name | Data Type | Default Value                  | Description   |
|----------------|-----------|--------------------------------|---|
| stx            | float[6]  | [500, 500, 500, 100, 100, 100] | Three translational stiffnesses<br>Three rotational stiffnesses                                 |
| time           | float     | 0                              | Stiffness varying time [sec]<br>Range: 0 - 1.0<br>* Linear transition during the specified time |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```
set_ref_coord(DR_WORLD) # Global coordinate is the world coordinate
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
task_compliance_ctrl()
stx = [1, 2, 3, 4, 5, 6]
set_stiffnessx(stx) # Set the stiffness value based on the global coordinate(world coordinate)
release_compliance_ctrl()
```

### ▪ Related commands

**task\_compliance\_ctrl()/release\_compliance\_ctrl()**

## 5.9 calc\_coord(x1, x2, x3, x4, ref, mod)

### ▪ Features

This function returns a new user cartesian coordinate system by using up to 4 input poses ([x1]~[x4]), input mode [mod] and the reference coordinate system [ref]. The input mode is only valid when the number of input robot poses is 2.

In the case that the number of input poses is 1, the coordinate system is calculated using the position and orientation of x1.

In the case that the number of input poses is 2 and the input mode is 0, X-axis is defined by the direction from x1 to x2, and Z-axis is defined by the projection of the current Tool-Z direction onto the plane orthogonal to the x-axis. The origin is the position of x1.

In the case that the number of input poses is 2 and the input mode is 1, X-axis is defined by the direction from x1 to x2, and Z-axis is defined by the projection of the z direction of x1 onto the plane orthogonal to the X-axis. The origin is the position of x1.

In the case that the number of input poses is 3, X-axis is defined by the direction from x1 to x2. If a vector v is the direction from x1 to x3, Z-axis is defined by the cross product of X-axis and v (X-axis cross v). The origin is the position of x1.

In the case that the number of input poses is 4, the definition of axes is identical to the case that the number of input poses is 3, but the origin is the position of x4.

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description   |
|----------------|-------------------------|---------------|---|
| x1, x2, x3, x4 | Posx<br>list (float[6]) | -             | Posx or position list   |
| ref            | int                     | -             | reference coordinate<br><ul style="list-style-type: none"> <li>DR_BASE: base coordinate</li> <li>DR_WORLD: world coordinate</li> </ul>  |
| mod            | int                     | -             | input mode<br>(only valid when the number of input poses is 2)<br><ul style="list-style-type: none"> <li>0: defining z-axis based on the current Tool-z direction</li> <li>1: defining z-axis based on the z direction of x1</li> </ul> |

### ▪ Return

| Value | Description  |
|-------|--|
| posx  | Successful coordinate calculation<br>Position information of the calculated coordinate |



- **Exception**

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

- **Example**

```
pos1 = posx(500, 30, 500, 0, 0, 0)
pos2 = posx(400, 30, 500, 0, 0, 0)
pos3 = posx(500, 30, 600, 45, 180, 45)
pos4 = posx(500, -30, 600, 0, 180, 0)
pose_user1 = calc_coord(pos1, ref=DR_BASE, mod=0)
pose_user21 = calc_coord(pos1, pos2, ref=DR_WORLD, mod=0)
%% Define z-axis based on the Tool-z direction.
pose_user22 = calc_coord(pos1, pos2, ref=DR_BASE, mod=1)
%% Define z-axis based on the z direction of pos1
pose_user3 = calc_coord(pos1, pos2, pos3, ref=DR_BASE, mod=0)
pose_user4 = calc_coord(pos1, pos2, pos3, pos4, ref=DR_WORLD, mod=0)
ucart1 = set_user_cart_coord(pose_user1, ref=DR_BASE)
ucart2 = set_user_cart_coord(pose_user21, ref=DR_WORLD)
```

- **Related commands**

`set_user_cart_coord()`

## 5.10 set\_user\_cart\_coord(pos, ref)

### ▪ Features

This function set a new user cartesian coordinate system using input pose [pos] and reference coordinate system[ref]. Up to 20 user coordinate systems can be set including the coordinate systems set within Workcell Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| pos            | Posx            | -             | coordinate information<br>(position and orientation)   |
|                | list (float[6]) |               |  |
| ref            | int             | -             | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> </ul> |

### ▪ Return

| Value            | Description  |
|------------------|--|
| Positive integer | Successful coordinate setting<br>Set coordinate ID (101 - 200) |
| -1               | Failed coordinate setting                                      |

### ▪ Exception

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

### ▪ Example

```
pos1 = posx(10, 20, 30, 0, 0, 0)
pos2 = posx(30, 50, 70, 45, 180, 45)
user_id1 = set_user_cart_coord(pos1, ref=DR_BASE)
user_id2 = set_user_cart_coord(pos2, ref=DR_WORLD)
```

- **Related commands**

- set\_ref\_coord()

## 5.11 set\_user\_cart\_coord(x1, x2, x3, pos, ref)

### ▪ Features

This function sets a new user cartesian coordinate system using [x1], [x2], and [x3] based on ref coordinate system[ref]. Creates a user coordinate system with ux, uy, and uz as the vector for each axis and origin position is the position of [pos] based on [ref]. <sup>1)</sup>ux is defined as the unit vector of x1x2, uz is defined as the unit vector defined by the cross product of x1x2 and x1x3 (x1x2 cross x1x3). uy is can be determined by right hand rule (uz cross ux). Up to 20 user coordinate systems can be set including the coordinate systems set within Workcell Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

<sup>1)</sup>Before M2.0.2 software version, ux is the unit vector of x2x1

### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
| x1             | Posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| x2             | Posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| x3             | Posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| pos            | Posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |
| ref            | int             | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>• DR_BASE: base coordinate</li> <li>• DR_WORLD: world coordinate</li> </ul> |

### ▪ Return

| Value            | Description  |
|------------------|--|
| Positive integer | Successful coordinate setting<br>Set coordinate ID (101 - 200) |
| -1               | Failed coordinate setting                                      |

- **Exception**

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

- **Example**

```
x1 = posx(0, 500, 700, 0, 0, 0) # Ignores the Euler angle.
x2 = posx(500, 0, 700, 0, 0, 0)
x3 = posx(300, 100, 500, 0, 0)
x4 = posx(300, 110, 510, 0, 0)
pos = posx(10, 20, 30, 0, 0, 0)
user_tc1 = set_user_cart_coord(x1, x2, x3, pos, DR_BASE)
user_tc2 = set_user_cart_coord(x2, x3, x4, pos, DR_WORLD)
```

- **Related commands**

set\_ref\_coord()

## 5.12 set\_user\_cart\_coord(u1, v1, pos, ref)

### ▪ Features

This function sets a new user cartesian coordinate system using [u1] and [v1] based on [ref] coordinate system. The origin position the position of [pos] based on the [ref] coordinate while the direction of x-axis and y-axis bases are given in the vectors u1 and v1, respectively. Other directions are determined by  $u1 \text{ cross } v1$ . If u1 and v1 are not orthogonal,  $v1'$ , that is perpendicular to u1 on the surface spanned by u1 and v1, is set as the vector in the y-axis direction. Up to 20 user coordinate systems can be set including the coordinate systems set within Workcell Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description  |
|----------------|-------------------------|---------------|--|
| u1             | float[3]                | -             | X-axis unit vector   |
| v1             | float[3]                | -             | Y-axis unit vector   |
| pos            | posx<br>list (float[6]) | -             | posx or<br>position list   |
| ref            | int                     | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>· DR_BASE: base coordinate</li> <li>· DR_WORLD: world coordinate</li> </ul> |

### ▪ Return

| Value            | Description  |
|------------------|--|
| Positive integer | Successful coordinate setting<br>Set coordinate ID (101 - 200) |
| -1               | Failed coordinate setting                                      |

### ▪ Exception

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

- **Example**

```
u1 = [1, 1, 0]
v1 = [-1, 1, 0]
pos = posx(10, 20, 30, 0, 0, 0)
user_tc1 = set_user_cart_coord(u1, v1, pos)
user_tc2 = set_user_cart_coord(u1, v1, pos, ref=DR_WORLD)
```

- **Related commands**

set\_ref\_coord()

## 5.13 overwrite\_user\_cart\_coord(id, pos, ref)

### ▪ Features

This function changes the pose and reference coordinate system of the requested user coordinate system [id] with the [pos] and [ref], respectively.

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description  |
|----------------|-------------------------|---------------|--|
| id             | int                     | -             | coordinate ID  |
| pos            | posx<br>list (float[6]) | -             | posx or<br>position list   |
| ref            | int                     | DR_BASE       | reference coordinate<br><ul style="list-style-type: none"> <li>· DR_BASE: base coordinate</li> <li>· DR_WORLD: world coordinate</li> </ul> |

### ▪ Return

| Value            | Description  |
|------------------|--|
| Positive integer | Successful coordinate setting<br>Set coordinate ID (101 - 200) |
| -1               | Failed coordinate setting                                      |

### ▪ Exception

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

### ▪ Example

```
pose_user1 = posx(30, 40, 50, 0, 0, 0)
id_user = set_user_cart_coord(pose_user1, ref=DR_BASE)
pose_user2 = posx(100, 150, 200, 45, 180, 0)
overwrite_user_cart_coord(id_user, pose_user2, ref=DR_BASE)
```

### ▪ Related commands

`set_user_cart_coord()`



## 5.14 `get_user_cart_coord(id)`

### ▪ Features

This function returns the pose and reference coordinate system of the requested user coordinate system [id].

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---------------|
| id             | int       | -             | coordinate ID |

### ▪ Return

| Value | Description   |
|-------|---|
| posx  | Position and orientation information of the coordinate to get |
| ref   | Reference coordinate of the coordinate to get                 |

### ▪ Exception

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

### ▪ Example

```
pose_user1 = posx(10, 20, 30, 0, 0, 0)
id_user = set_user_cart_coord(pose_user1, ref=DR_BASE)
pose, ref = get_user_cart_coord(id_user)
```

### ▪ Related commands

`set_user_cart_coord()`

## 5.15 set\_desired\_force(Fd, dir, time, mod)

### ▪ Features

This function defines the target force, direction, translation time, and mode for force control based on the global coordinate.

### ▪ Parameters

| Parameter Name | Data Type | Default Value      | Description   |
|----------------|-----------|--------------------|---|
| fd             | float[6]  | [0, 0, 0, 0, 0, 0] | Three translational target forces<br>Three rotational target moments  |
| dir            | int[6]    | [0, 0, 0, 0, 0, 0] | Force control in the corresponding direction if 1<br>Compliance control in the corresponding direction if 0   |
| time           | float     | 0                  | Transition time of target force to take effect [sec]<br>Range: 0 - 1.0  |
| mod            | int       | DR_FC_MOD_ABS      | DR_FC_MOD_ABS: Force control with absolute value<br>DR_FC_MOD_REL: force control with relative value to initial state (the instance when this function is called) |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

 **Note**

- The value of external force refers to the sensor measurement at terminating the force control (control mode transition to compliance control) by the command `release_force()`. Therefore, the variation in external force can occur if the option `mod=DR_FC_MOD_REL` is applied.
- Tool weight and external force value refer to the sensor measurement regardless of the setting for 'mod'

 **Caution**

- To retain the accuracy in force control, it is recommended to start force control with setting `mod=DR_FC_MOD_REL` near the contact point.

▪ **Example**

```
set_ref_coord(DR_TOOL)
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
task_compliance_ctrl(stx=[500, 500, 500, 100, 100, 100])
fd = [0, 0, 0, 0, 0, 10]
fctrl_dir= [0, 0, 1, 0, 0, 1]
set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL)
# Executed in the global coordinate(tool coordinate)
# Zero force control in the z-axis direction of the tool, moment control in the z-axis
direction of the tool, and compliance control in the other directions
# Force control with the relative value to the sensor measurement at starting the force
control
```

▪ **Related commands**

`release_force()/task_compliance_ctrl()/set_stiffnessx()/release_compliance_ctrl()`

## 5.16 release\_force(time=0)

### ▪ Features

This function reduces the force control target value to 0 through the time value and returns the task space to adaptive control.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                                       |
|----------------|-----------|---------------|---|
| time           | float     | 0             | Time needed to reduce the force<br>Range: 0 - 1.0 |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

### ▪ Exception

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

### ▪ Example

```

j0 = posj(0, 0, 90, 0, 90, 0)
x0 = posx(0, 0, 0, 0, 0, 0)
x1 = posx(0, 500, 700, 0, 180, 0)
x2 = posx(300, 100, 700, 0, 180, 0)
x3 = posx(300, 100, 500, 0, 180, 0)
set_velx(100,20)
set_accx(100,20)
movej(j0, vel=10, acc=10)
movel(x2)
task_compliance_ctrl(stx = [500, 500, 500, 100, 100, 100])
fd = [0, 0, 0, 0, 0, 10]
fctrl_dir= [0, 0, 1, 0, 0, 1]
set_desired_force(fd, dir=fctrl_dir, time=1.0)
movel(x3, v=10)
release_force(0.5)
release_compliance_ctrl()
    
```

### ▪ Related commands

**set\_desired\_force()/task\_compliance\_ctrl()/set\_stiffnessx()/release\_compliance\_ctrl()**

## 5.17 `check_position_condition(axis, min, max, ref, mod, pos)`

### ▪ Features

This function checks the status of the given position. This condition can be repeated with the while or if statement. Axis and pos of input paramets are based on the ref coordinate.

In case of ref=DR\_TOOL, pos should be defined in BASE coordinate.

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description   |
|----------------|-------------------------|---------------|---|
| axis           | int                     | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> </ul>  |
| min            | float                   | DR_COND_NONE  | Minimum value   |
| max            | float                   | DR_COND_NONE  | Maximum value   |
| ref            | int                     | None          | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| mod            | int                     | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> <li>• DR_MV_MOD_ABS: Absolute</li> <li>• DR_MV_MOD_REL: Relative</li> </ul>   |
| pos            | posx<br>list (float[6]) | -             | posx or<br>position list  |

### Note

- The absolution position is used if the mod is DR\_MV\_MOD\_ABS.
- The pos position is used if the mod is DR\_MV\_MOD\_REL.
- Pos is meaningful only if the mod is DR\_MV\_MOD\_REL.

### ▪ Return

| Value | Description            |
|-------|------------------------|
| True  | The condition is True. |

## check\_position\_condition(axis, min, max, ref, mod, pos)

| Value | Description             |
|-------|-------------------------|
| False | The condition is False. |

### ▪ Exception

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

### ▪ Example

```
CON1= check_position_condition(DR_AXIS_X, min=-5, max=0, ref=DR_WORLD)
CON2= check_position_condition(DR_AXIS_Y, max=700)
CON3= check_position_condition(DR_AXIS_Z, min=-10, max=-5)    # -10 ≤ z ≤ -5
CON4= check_position_condition(DR_AXIS_Z, min=30)            # 30 ≤ z

CON5= check_position_condition(DR_AXIS_Z,min=-10,max=-5, ref=DR_BASE)
                                           # -10 ≤ z ≤ -5

CON6= check_position_condition(DR_AXIS_Z,min=-10,max=-5,
mod=DR_MV_MOD_ABS)
      # -10 ≤ z ≤ -5

posx1 = posx(400, 500, 800, 0, 180,0)
CON7= check_position_condition(DR_AXIS_Z,min=-10,max=-5,mod =
DR_MV_MOD_REL, pos=posx1)                    # posx1(z) -
10 ≤ z ≤ posx1(z) - 5
```

### ▪ Related commands

`check_force_condition()/check_orientation_condition()/set_ref_coord()`

## 5.18 check\_force\_condition(axis, min, max, ref)

### ▪ Features

This function checks the status of the given force. It disregards the force direction and only compares the sizes. This condition can be repeated with the while or if statement. Measuring the force, axis is based on the ref coordinate and measuring the moment, axis is based on the tool coordinate.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| axis           | int       | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis</li> <li>• DR_AXIS_Y: y-axis</li> <li>• DR_AXIS_Z: z-axis</li> <li>• DR_AXIS_A: x-axis rotation</li> <li>• DR_AXIS_B: y-axis rotation</li> <li>• DR_AXIS_C: z-axis rotation</li> </ul> |
| min            | float     | DR_COND_NONE  | Minimum value (min ≥ 0)  |
| max            | float     | DR_COND_NONE  | Maximum value (max ≥ 0)  |
| ref            | int       | None          | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: User defined</li> </ul>                                |

### ▪ Return

| Value | Description             |
|-------|-------------------------|
| True  | The condition is True.  |
| False | The condition is False. |

### ▪ Exception

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

- **Example**

```
fcon1 = check_force_condition(DR_AXIS_Z, min=5, max=10, DR_WORLD)
#  $5 \leq f_z \leq 10$ 

while (fcon1):
    fcon2 = check_force_condition(DR_AXIS_C, min=30)           #  $30 \leq m_z$ 
    pcon1 = check_position_condition(DR_AXIS_X, min=0, max=0.1) #  $0 \leq x \leq 0.1$ 

    if (fcon2 and pcon1):
        break
```

- **Related commands**

`check_position_condition()/check_orientation_condition()/set_ref_coord()`

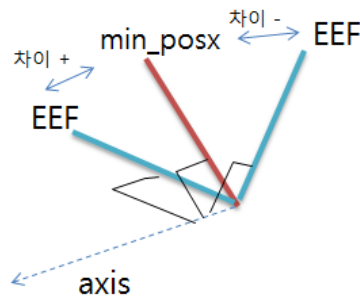


## 5.19 check\_orientation\_condition(axis, min, max, ref, mod)

### ▪ Features

This function checks the difference between the current pose and the specified pose of the robot end effector. It returns the difference between the current pose and the specified pose in rad with the algorithm that transforms it to a rotation matrix using the "AngleAxis" technique. It returns True if the difference is positive (+) and False if the difference is negative (-). It is used to check if the difference between the current pose and the rotating angle range is + or -. For example, the function can use the direct teaching position to check if the difference from the current position is + or - and then create the condition for the orientation limit. This condition can be repeated with the while or if statement.

- Setting Min only: True if the difference is + and False if -
- Setting Min and Max: True if the difference from min is - while the difference from max is + and False otherwise
- Setting Max only: True if the maximum difference is + and False otherwise



### ▪ Parameters

| Parameter Name | Data Type       | Default Value | Description   |
|----------------|-----------------|---------------|---|
| axis           | int             | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_A: x-axis rotation</li> <li>• DR_AXIS_B: y-axis rotation</li> <li>• DR_AXIS_C: z-axis rotation</li> </ul>   |
| min            | posx            | -             | posx or position list   |
|                | list (float[6]) |               |   |
| max            | posx            | -             | posx or position list   |
|                | list (float[6]) |               |   |
| ref            | int             | None          | reference coordinate <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |

## check\_orientation\_condition(axis, min, max, ref, mod)

| Parameter Name | Data Type | Default Value | Description                                 |
|----------------|-----------|---------------|---|
| mod            | int       | DR_MV_MOD_ABS | Movement basis<br>· DR_MV_MOD_ABS: Absolute |

### Return

| Value | Description             |
|-------|-------------------------|
| True  | The condition is True.  |
| False | The condition is False. |

### Exception

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

### Example

```
posx1 = posx(400,500,800,0,180,30)
posx2 = posx(400,500,500,0,180,60)
```

```
CON1= check_orientation_condition(DR_AXIS_C, min=posx1, max= posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=True since posx1 Rz=30 < posxc Rz=40 < posx2 Rz=60
```

```
CON2= check_orientation_condition(DR_AXIS_C, min=posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 15)
# CON2=False since posx1 Rz=30 > posxc Rz=15
```

```
CON3= check_orientation_condition(DR_AXIS_C, max= posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 75)
# CON2=False since posx1 Rz=75 > posxc Rz=60
```

### Related commands

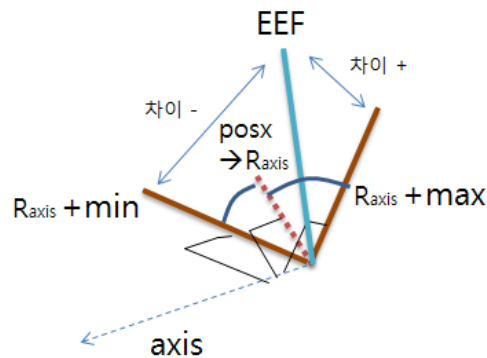
```
check_position_condition()/check_force_condition()/check_orientation_condition()
/set_ref_coord()
```

## 5.20 `check_orientation_condition(axis, min, max, ref, mod, pos)`

### ▪ Features

This function checks the difference between the current pose and the rotating angle range of the robot end effector. It returns the difference (in rad) between the current pose and the rotating angle range with the algorithm that transforms it to a rotation matrix using the "AngleAxis" technique. It returns True if the difference is positive (+) and False if the difference is negative (-). It is used to check if the difference between the current pose and the rotating angle range is + or -. For example, the function can be used to set the rotating angle range to min and max at any reference position, and then determine the orientation limit by checking if the difference from the current position is + or -. This condition can be repeated with the while or if statement.

- Setting Min only: True if the difference is + and False if -
- Setting Min and Max: True if the difference from min is - while the difference from max is + and False if the opposite.
- Setting Max only: True if the maximum difference is + and False otherwise



### Note

Range of rotating angle: This means the relative angle range (min, max) based on the specified axis from a given position based on the ref coordinate.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| axis           | int       | -             | axis <ul style="list-style-type: none"> <li>• DR_AXIS_X: x-axis rotation</li> <li>• DR_AXIS_Y: y-axis rotation</li> <li>• DR_AXIS_Z: z-axis rotation</li> </ul> |
| min            | float     | DR_COND_NONE  | Minimum value   |
| max            | float     | DR_COND_NONE  | Maximum value   |
| ref            | int       | None          | reference coordinate  |

## check\_orientation\_condition(axis, min, max, ref, mod, pos)

| Parameter Name | Data Type       | Default Value | Description  |
|----------------|-----------------|---------------|--|
|                |                 |               | <ul style="list-style-type: none"> <li>DR_BASE : base coordinate</li> <li>DR_WORLD : world coordinate</li> <li>DR_TOOL : tool coordinate</li> <li>user coordinate: User defined</li> </ul> |
| mod            | int             | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> <li>DR_MV_MOD_REL: Relative</li> </ul>   |
| pos            | posx            | -             | posx or position list  |
|                | list (float[6]) |               |  |

### Return

| Value | Description             |
|-------|-------------------------|
| True  | The condition is True.  |
| False | The condition is False. |

### Exception

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

### Example

```

posx1 = posx(400,500,800,0,180,15)
CON1= check_orientation_condition(DR_AXIS_C, min=-5, mod=DR_MV_MOD_REL,
pos=posx1, DR_WORLD)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=True since posx1 Rz=15 - (min=5) < posxc Rz=40

CON1= check_orientation_condition(DR_AXIS_C, max=5, mod=DR_MV_MOD_REL,
pos=posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=False since posxc Rz=40 > posx1 Rz=15 + (max=5)

```

### Related commands

check\_position\_condition()/check\_force\_condition()/check\_orientation\_condition()  
/set\_ref\_coord()

## 5.21 coord\_transform(pose\_in, ref\_in, ref\_out)

### ▪ Features

This function transforms given task position expressed in reference coordinate, 'ref\_in' to task position expressed in reference coordinate, 'ref\_out'. It returns transformed task position. It supports calculation of coordinate transformation for the following cases.

- (ref\_in) world reference coordinate → (ref\_out) world reference coordinate
- (ref\_in) world reference coordinate → (ref\_out) base reference coordinate
- (ref\_in) world reference coordinate → (ref\_out) tool reference coordinate
- (ref\_in) world reference coordinate → (ref\_out) user reference coordinate
- (ref\_in) base reference coordinate → (ref\_out) base reference coordinate
- (ref\_in) base reference coordinate → (ref\_out) tool reference coordinate
- (ref\_in) base reference coordinate → (ref\_out) user reference coordinate
- (ref\_in) tool reference coordinate → (ref\_out) world reference coordinate
- (ref\_in) tool reference coordinate → (ref\_out) base reference coordinate
- (ref\_in) tool reference coordinate → (ref\_out) tool reference coordinate
- (ref\_in) tool reference coordinate → (ref\_out) user reference coordinate
- (ref\_in) user reference coordinate → (ref\_out) world reference coordinate
- (ref\_in) user reference coordinate → (ref\_out) base reference coordinate
- (ref\_in) user reference coordinate → (ref\_out) tool reference coordinate
- (ref\_in) user reference coordinate → (ref\_out) user reference coordinate

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| Pose_in        | posx      | -             | posx  |
| ref_in         | float     | DR_COND_NONE  | reference coordinate before transformation <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: User defined</li> </ul> |
| ref_out        | float     | DR_COND_NONE  | reference coordinate after transformation <ul style="list-style-type: none"> <li>• DR_BASE : base coordinate</li> <li>• DR_WORLD : world coordinate</li> <li>• DR_TOOL : tool coordinate</li> <li>• user coordinate: User defined</li> </ul>  |

## coord\_transform(pose\_in, ref\_in, ref\_out)

---

### ▪ Return

| Value | Description |
|-------|-------------|
| pos   | posx        |

### ▪ Exception

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

### ▪ Example

```
base_pos = posx(400,500,800,0,180,15)
# If task position based on base reference coordinate base_pos =
posx(400,500,800,0,180,15)

tool_pos = coord_transform(base_pos, DR_BASE, DR_TOOL)
# Transform task position(base_pos) expressed in base reference coordinate to task
position expressed in tool reference coordinate
# This command returns task position expressed in tool reference coordinate and the
result value is stored in tool_pos
```

### ▪ Related commands

`set_user_cart_coord()/get_current_posx()/get_desired_posx()/set_ref_coord()`

## 6. System Commands

### 6.1 IO Related

#### 6.1.1 `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 <ul style="list-style-type: none"> <li>• Val argument existing: A number between 1 and 16</li> <li>• No val argument: 1 ~ 16 , -1 ~ -16</li> </ul> (A positive number means ON while a negative number means OFF.) |
| val            | int       | -             | I/O value <ul style="list-style-type: none"> <li>• ON: 1</li> <li>• OFF: 0</li> </ul>   |

##### 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.)
```



## 6.1.2 set\_digital\_outputs(bit\_list)

### ▪ Features

This function sends a signal to multiple digital output contact points of the controller. The digital signals of the contact points defined in bit\_list are output at one.

### ▪ Parameters

| Parameter Name | Data Type  | Default Value | Description  |
|----------------|------------|---------------|--|
| bit_list       | list (int) | -             | List of multiple output contacts <ul style="list-style-type: none"> <li>• The positive contact number outputs ON: 1~16</li> <li>• The negative contact number outputs OFF: -1~-16</li> </ul> |

### ▪ 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_outputs(bit_list=[1,2,3,4,5,6,7,8]) # Contact number 1-8 ON
set_digital_outputs([-1,-2,-3,-4,-5,-6,-7,-8]) # Contact number 1-8 OFF
set_digital_outputs([1,-2,3]) # Contact no. 1 ON, no. 2 OFF, and no. 3 ON
set_digital_outputs([4,-9,-12]) # Contact no. 4 ON, no. 9 OFF, and no. 12 OFF
```

### 6.1.3 set\_digital\_outputs(bit\_start, bit\_end, val)

#### ▪ Features

This function sends multiple signals at once from the digital output start contact point (bit\_start) to the end contact point (bit\_end) of the controller.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                                       |
|----------------|-----------|---------------|---|
| bit_start      | int       | -             | Beginning contact number for output signal (1~16) |
| bit_end        | int       | -             | Ending contact number for output signal (1~16)    |
| val            | int       | -             | Output value                                      |

#### Note

- Bit\_end must be a larger number than bit\_start.
- Val is the value of the combination of bits where bit\_start =LSB and bit\_end=MSB.  
Ex) bit\_start =1, bit\_end=4, val=0b1010 # No. 4=ON, no. 3=OFF, no. 2=ON, and no. 1=OFF

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

- **Example**

```
# Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact 4=OFF.  
set_digital_outputs(bit_start=1, bit_end=4, val=0b0011) # 0b means a binary number.
```

```
# Outputs contact 3=ON and contact 4=OFF.  
set_digital_outputs(bit_start=3, bit_end=4, val=0b01) # 0b means a binary number.
```

```
# Outputs the ON signal from contacts 1 through 8.  
set_digital_outputs(1, 8, 0xff) # 0x means a hexadecimal  
number.
```

### 6.1.4 `get_digital_input(index)`

#### ▪ Features

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

#### ▪ Parameters

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

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 1              | ON          |
| 0              | OFF         |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
in1 = get_digital_input(1)    # Reads the no. 1 contact
in8 = get_digital_input(8)    # Reads the no. 8 contact
```

### 6.1.5 `get_digital_inputs(bit_list)`

#### ▪ Features

This function reads the signals from multiple digital contact points of the controller. The digital signals of the contact points defined in `bit_list` are input at one.

#### ▪ Parameters

| Parameter Name     | Data Type               | Default Value | Description   |
|--------------------|-------------------------|---------------|---|
| <code>index</code> | <code>list (int)</code> | -             | List of contact points to read<br>A number 1–16 which means the I/O contact number mounted on the controller. |

#### ▪ Return

| Value                     | Description  |
|---------------------------|--|
| <code>int (&gt;=0)</code> | Multiple contacts to be read at once<br>(the value of the combination of the bit list where <code>bit_start</code> =LSB and <code>bit_end</code> =MSB) |
| Negative number           | Failed   |

#### ▪ Exception

| Exception                                | Description                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | Program terminated forcefully      |

#### ▪ Example

```
# input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_inputs(bit_list=[1,2,3,4])
#res expected value = 0b1100 (binary number), 12 (decimal number), or 0x0C (hexadecimal number)

# input contacts: No. 5=ON, No. 6=ON, No. 7=OFF, and No. 8=ON
res = get_digital_inputs([5,6,7,8])
#res expected value = 0b1011 (binary number), 11 (decimal number), or 0x0B (hexadecimal number)
```

### 6.1.6 `get_digital_inputs(bit_start, bit_end)`

#### ▪ Features

This function reads multiple signals at once from the digital input start contact point (`start_index`) to the end contact point (`end_index`) of the controller.

#### ▪ Parameters

| Parameter Name         | Data Type | Default Value | Description                                       |
|------------------------|-----------|---------------|---|
| <code>bit_start</code> | int       | -             | Beginning contact number for input signals (1~16) |
| <code>bit_end</code>   | int       | -             | Ending contact number for input signals (1~16)    |



#### Note

`Bit_end` must be a larger number than `bit_start`.

#### ▪ Return

| Value           | Description   |
|-----------------|---|
| int (>=0)       | Multiple contacts to be read at once<br>Value of the combination of bits where <code>bit_start</code> =LSB and <code>bit_end</code> =MSB. |
| Negative number | Failed  |

#### ▪ Exception

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

#### ▪ Example

```
# input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_inputs(bit_start=1, bit_end=4)
#res expected value = 0b1100 (binary number), 12 (decimal number), or 0x0C
(hexadecimal number)
```

### 6.1.7 wait\_digital\_input(index, val, timeout=None)

#### ▪ Features

This function waits until the signal value of the digital input register of the controller becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| index          | int       | -             | A number 1 - 16 which means the I/O index mounted on the controller.                       |
| value          | int       | -             | I/O value<br><ul style="list-style-type: none"> <li>• ON : 1</li> <li>• OFF : 0</li> </ul> |
| timeout        | float     | -             | Waiting time (sec)<br>This function waits indefinitely if the timeout is not set.          |

#### ▪ Return

| Value | Description       |
|-------|-------------------|
| 0     | Success           |
| -1    | Failed (time-out) |

#### ▪ Exception

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

#### ▪ Example

```
wait_digital_input(1, ON) # Indefinite wait until the no. 1 contact becomes ON
wait_digital_input(2, OFF) # Indefinite wait until the no. 2 contact becomes OFF
res = wait_digital_input(1, ON, 3) # Wait for up to 3 seconds until the no. 1 contact
becomes ON
    # Waiting is terminated and res = 0 if the no. 1 contact becomes ON within 3
seconds.
    # Waiting is terminated and res = -1 if the no. 1 contact does not become ON within
3 seconds.
```

### 6.1.8 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 <ul style="list-style-type: none"> <li>• Val argument existing: A number between 1 and 6</li> <li>• No val argument: 1 ~ 6 , -1 ~ -6</li> </ul> (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.)
```



### 6.1.9 set\_tool\_digital\_outputs(bit\_list)

#### ▪ Features

This function sends the signal of the robot tool from the digital contact point. The digital signals of the contact points defined in `bit_list` are output at one.

#### ▪ Parameters

| Parameter Name        | Data Type  | Default Value | Description  |
|-----------------------|------------|---------------|--|
| <code>bit_list</code> | list (int) | -             | List of multiple output contacts <ul style="list-style-type: none"> <li>• The positive contact number outputs ON: 1~6</li> <li>• The negative contact number outputs OFF: -1~-6</li> </ul> |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

#### ▪ Exception

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

#### ▪ Example

```
set_tool_digital_outputs(bit_list=[1,2,3,4,5,6]) # Sets the contacts 1-6 ON
set_tool_digital_outputs([-1,-2,-3,-4,-5,-6]) # Sets the contacts 1-6 OFF
set_digital_outputs([1,-2,3]) # Contact no. 1 ON, no. 2 OFF, and no. 3 ON
```

### 6.1.10 `set_tool_digital_outputs(bit_start, bit_end, val)`

#### ▪ Features

This function sends the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (`bit_start`) to the last contact point (`bit_end`) are output at one.

#### ▪ Parameters

| Parameter Name         | Data Type | Default Value | Description                                      |
|------------------------|-----------|---------------|--|
| <code>bit_start</code> | int       | -             | Beginning contact number for output signal (1~6) |
| <code>bit_end</code>   | int       | -             | Ending contact number for output signal (1~6)    |
| <code>Val</code>       | int       | -             | Output value                                     |

#### Note

- `Bit_end` must be a larger number than `bit_start`.
- `Val` is the value of the combination of bits where `bit_start` =LSB and `bit_end`=MSB.  
Ex) `bit_start` =1, `bit_end`=4, `val`=0b1010 # No. 4=ON, no. 3=OFF, no. 2=ON, and no. 1=OFF

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Error       |

#### ▪ Exception

| Exception                                | Description                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | Program terminated forcefully      |

- **Example**

```
# Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact 4=OFF.  
set_tool_digital_outputs(bit_start=1, bit_end=4, val=0b0011) # 0b means a binary  
number.
```

```
# Outputs contact 3=ON and contact 4=OFF.  
set_tool_digital_outputs(bit_start=3, bit_end=4, val=0b01) # 0b means a binary number.
```

```
# Outputs the ON signal from contacts 1 through 8.  
set_tool_digital_outputs(1, 8, 0xff) # 0x means a hexadecimal  
number.
```

### 6.1.11 `get_tool_digital_input(index)`

#### ▪ Features

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

#### ▪ Parameters

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

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 1              | ON          |
| 0              | OFF         |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
get_tool_digital_input(1)    # Reads the no. 1 contact of tool I/O
get_tool_digital_input(6)    # Reads the no. 6 contact of tool I/O
```

### 6.1.12 `get_tool_digital_inputs(bit_list)`

#### ▪ Features

This function reads the signal of the robot tool from the digital contact point. The digital signals of the contact points defined in `bit_list` are input at one.

#### ▪ Parameters

| Parameter Name        | Data Type  | Default Value | Description   |
|-----------------------|------------|---------------|---|
| <code>bit_list</code> | list (int) | -             | List of contact points to read <ul style="list-style-type: none"> <li>• (I/O contact numbers (1-6) mounted on the robot arm)</li> </ul> |

#### ▪ Return

| Value           | Description  |
|-----------------|--|
| int (>=0)       | (the value of the combination of the bit list where <code>bit_start</code> =LSB and <code>bit_end</code> =MSB) |
| Negative number | Failed   |

#### ▪ Exception

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

#### ▪ Example

```
# input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
res = get_tool_digital_inputs(bit_list=[1,2,3]) # Reads the contacts 1, 2, and 3 at once.
#res expected value = 0b100 (binary number), 4 (decimal number), or 0x04 (hexadecimal number)
```

```
# input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
res = get_tool_digital_inputs([4,5,6])
#res expected value = 0b011 (binary number), 3 (decimal number), or 0x03 (hexadecimal number)
```

### 6.1.13 `get_tool_digital_inputs(bit_start, bit_end)`

#### ▪ Features

This function reads the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (`start_index`) to the last contact point (`end_index`) are input at one.

#### ▪ Parameters

| Parameter Name         | Data Type | Default Value | Description                                      |
|------------------------|-----------|---------------|--|
| <code>bit_start</code> | int       | -             | Beginning contact number for input signals (1~6) |
| <code>bit_end</code>   | int       | -             | Ending contact number for input signals (1~6)    |

#### ▪ Return

| Value           | Description   |
|-----------------|---|
| int (>=0)       | Multiple contacts to be read at once<br>Value of the combination of bits where <code>bit_start</code> =LSB and <code>bit_end</code> =MSB. |
| Negative number | Failed  |

#### ▪ Exception

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

#### ▪ Example

```
# input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
res = get_tool_digital_inputs(bit_start=1, bit_end=3)
#res expected value = 0b100 (binary number), 4 (decimal number), or 0x04
(hexadecimal number)

# input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
res = get_tool_digital_inputs(4, 6)
#res expected value = 0b011 (binary number), 3 (decimal number), or 0x03
(hexadecimal number)
```

### 6.1.14 wait\_tool\_digital\_input(index, val, timeout=None)

#### ▪ Features

This function waits until the digital input signal value of the robot tool becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| index          | int       | -             | A number in 1 - 6 which means the I/O index mounted on the robot arm                       |
| value          | int       | -             | I/O value<br><ul style="list-style-type: none"> <li>• ON : 1</li> <li>• OFF : 0</li> </ul> |
| timeout        | float     | -             | Waiting time (sec)<br>This function waits indefinitely if the timeout is not set.          |

#### ▪ Return

| Value | Description       |
|-------|-------------------|
| 0     | Success           |
| -1    | Failed (time-out) |

#### ▪ Exception

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

- **Example**

```
wait_tool_digital_input(1, ON) # Indefinite wait until the no. 1 contact becomes ON  
wait_tool_digital_input(2, OFF) # Indefinite wait until the no. 2 contact becomes OFF
```

```
res = wait_tool_digital_input(1, ON, 3) # Wait for up to 3 seconds until the no. 1 contact  
becomes ON
```

```
    # Waiting is terminated and res = 0 if the no. 1 contact becomes ON within 3  
seconds.
```

```
    # Waiting is terminated and res = -1 if the no. 1 contact does not become ON within  
3 seconds.
```



### 6.1.15 `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       | -             | <ul style="list-style-type: none"> <li>• 1 : channel 1</li> <li>• 2 : channel 2</li> </ul>  |
| mod            | int       | -             | analog io mode <ul style="list-style-type: none"> <li>• DR_ANALOG_CURRENT: Current mode</li> <li>• DR_ANALOG_VOLTAGE: Voltage mode</li> </ul> |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
# Sets analog_output channel 1 to the current mode.
set_mode_analog_output(ch=1, mod=DR_ANALOG_CURRENT)

# Sets analog_output channel 2 to the voltage mode.
set_mode_analog_output(ch=2, mod=DR_ANALOG_VOLTAGE)
```

### 6.1.16 `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       | -             | <ul style="list-style-type: none"> <li>1 : channel 1</li> <li>2 : channel 2</li> </ul>  |
| mod            | int       | -             | analog io mode <ul style="list-style-type: none"> <li>DR_ANALOG_CURRENT: Current mode</li> <li>DR_ANALOG_VOLTAGE: Voltage mode</li> </ul> |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
# Sets analog_input channel 1 to the current mode.
set_mode_analog_input(ch=1, mod=DR_ANALOG_CURRENT)

# Sets analog_input channel 2 to the voltage mode.
set_mode_analog_input(ch=2, mod=DR_ANALOG_VOLTAGE)
```

### 6.1.17 `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       | -             | <ul style="list-style-type: none"> <li>• 1 : channel 1</li> <li>• 2 : channel 2</li> </ul>  |
| val            | float     | -             | analog output value <ul style="list-style-type: none"> <li>• Current mode: 4.0~20.0 [mA]</li> <li>• Voltage mode: 0~10.0 [V]</li> </ul> |

#### ▪ 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
```

### 6.1.18 `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       | -             | <ul style="list-style-type: none"> <li>• 1 : channel 1</li> <li>• 2 : channel 2</li> </ul> |

#### ▪ Return

| Value | Description   |
|-------|---|
| float | The analog input value of the specified channel <ul style="list-style-type: none"> <li>• Current mode: 4.0~20.0 [mA]</li> <li>• Voltage mode: 0~10.0 [V]</li> </ul> |

#### ▪ Exception

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

#### ▪ Example

```
set_mode_analog_input(ch=1, mod=DR_ANALOG_CURRENT) #input ch1=current mode
set_mode_analog_input(ch=2, mod=DR_ANALOG_VOLTAGE) #input ch2=voltage mode
```

```
Cur = get_analog_input(1) # Reads the analog input current value of channel 1
Vol = get_analog_input(2) # Reads the analog input voltage value of channel 2.
```

## 6.2 TP Interface

### 6.2.1 `tp_popup(message, pm_type=DR_PM_MESSAGE, type=0)`

#### ▪ Features

This function provides a message to users through the Teach Pendant. The higher level controller receives the string and displays it in the popup window, and the window must be closed by a user's confirmation.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| message        | string    | -             | Message provided to the user<br>(The message is limited to less than 256 bytes.)  |
| pm_type        | int       | DR_PM_MESSAGE | Message type <ul style="list-style-type: none"> <li>• DR_PM_MESSAGE</li> <li>• DR_PM_WARNING</li> <li>• DR_PM_ALARM</li> </ul>                      |
| type           | int       | 0             | button type of TP pop message <ul style="list-style-type: none"> <li>• 0 : show Stop &amp; Resume button</li> <li>• 1 : show Stop button</li> </ul> |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

- **Example**

```
tp_popup("move done", DR_PM_MESSAGE)
tp_popup("Error!! ", DR_PM_ALARM)
a=1
b=2
c=3
tp_popup("a={0}, b={1}, c={2}".format(a,b,c) ,DR_PM_MESSAGE)
tp_popup("critical error!! ", DR_PM_ALARM, 1)
```

## 6.2.2 `tp_log(message)`

### ▪ Features

This function records the user-written log to the Teach Pendant.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| message        | string    | -             | Log message<br>(The message is limited to less than 256 bytes.) |

### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

### ▪ Exception

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

### ▪ Example

```
tp_log("movej() is complete! ")
```

### 6.2.3 **tp\_progress(cur\_progress, total\_progress)**

▪ **Features**

This function provides a message to users through the Teach Pendant. The higher level controller receives the runtime data when a patterned program is run and is displayed in the GUI.

▪ **Parameters**

| Parameter Name | Data Type | Default Value | Description               |
|----------------|-----------|---------------|---------------------------|
| cur_progress   | int       | -             | Value at the current step |
| total_progress | int       | -             | Value at the final step   |

▪ **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

▪ **Exception**

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

▪ **Example**

```
tp_progress(1, 100)
tp_progress(99, 100)
```



## 6.2.4 `tp_get_user_input(message, input_type)`

### ▪ Features

This function receives the user input data through the Teach Pendant.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| message        | string    | -             | Character string message to be displayed on the TP user input window  |
| input_type     | int       | -             | TP user input message type <ul style="list-style-type: none"> <li>• DR_VAR_INT: Integer type</li> <li>• DR_VAR_FLOAT: Real number type</li> <li>• DR_VAR_STR: Character string</li> </ul> |

### ▪ Return

| Value           | Description                          |
|-----------------|--------------------------------------|
| User input data | User input data received from the TP |

### ▪ Exception

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

### ▪ Example

```
q1 = posj(10, 10, 10, 10, 10, 10)
q2 = posj(20, 20, 20, 20, 20, 20)
q3 = posj(30, 30, 30, 30, 30, 30)
q4 = posj(40, 40, 40, 40, 40, 40)
q5 = posj(50, 50, 50, 50, 50, 50)
q6 = posj(60, 60, 60, 60, 60, 60)

int_y= tp_get_user_input("message1", input_type= DR_VAR_INT)
if int_y==1:      # Moves to q1 if the TP user input is 1.
    movej(q1, vel=30, acc=30)
else:            # Moves to q2 if the TP user input is not 1.
    movej(q2, vel=30, acc=30)

float_y= tp_get_user_input("message2", input_type= DR_VAR_FLOAT)
if float_y==3.14:      # Moves to q3 if the TP user input is 3.14.
    movej(q3, vel=30, acc=30)
else:                # Moves to q4 if the TP user input is not 3.14.
    movej(q4, vel=30, acc=30)

str_y= tp_get_user_input("message3", input_type= DR_VAR_STR)
if str_y=="a":      # Moves to q5 if the TP user input is "a".
    movej(q5, vel=30, acc=30)
else:                # Moves to q6 if the TP user input is not "a".
    movej(q6, vel=30, acc=30)
```

## 6.3 Thread

### 6.3.1 `thread_run(th_func_name, loop=False)`

#### ▪ Features

This function creates and executes a thread. The features executed by the thread are determined by the functions specified in `th_func_name`.

#### Note

The following constraints are applied when using the thread command.

- Up to 4 threads can be used.
- The following motion command cannot be used to move the robot in the thread.
  - `movej`, `amovej`, `movejx`, `amovejx`, `movel`, `amovel`, `movec`, `amovec`, `movesj`, `amovesj`,
  - `movesx`, `amovesx`, `moveb`, `amoveb`, `move_spiral`, `amove_spiral`,
  - `move_periodic`, `amove_periodic`, `move_home`
- The thread commands do not operate normally when the `loop=True` during `thread_run` and the block is an indefinite loop within the thread function. (The thread is normally stopped when the stop command is executed through the TP.)

#### ▪ Parameters

| Parameter Name            | Data Type | Default Value | Description  |
|---------------------------|-----------|---------------|--|
| <code>th_func_name</code> | callable  | -             | Name of the function run by the thread   |
| <code>loop</code>         | bool      | False         | Flag indicates whether the thread will be repeated <ul style="list-style-type: none"> <li>• True: Repeated calling of <code>th_func_name</code> (interval 0.01second)</li> <li>• False: One-time calling of <code>th_func_name</code></li> </ul> |

#### ▪ Return

| Value          | Description          |
|----------------|----------------------|
| int            | Registered thread ID |
| Negative value | Failed               |

### ▪ Exception

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

### ▪ Example

```
#---- Thread -----
def fn_th_func():
    if check_motion()==0:    # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

#---- Main routine -----
th_id = thread_run(fn_th_func, loop=True) # Thread run

while 1:
    # do something...
    wait(0.1)
```

### 6.3.2 `thread_stop(th_id)`

#### ▪ Features

This function terminates a thread.

The program is automatically terminated when the DRL program is terminated even if the `thread_stop()` command is not used.

#### ▪ Parameters

| Parameter Name     | Data Type | Default Value | Description       |
|--------------------|-----------|---------------|-------------------|
| <code>th_id</code> | int       | -             | Thread ID to stop |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
def fn_th_func():
    if check_motion()==0:    # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)
#----- Main routine -----
th_id = thread_run(fn_th_func, loop=True)

# do something...
thread_stop(th_id) # Stops the thread.
```

### 6.3.3 thread\_pause(th\_id)

#### ▪ Features

This function temporarily suspends a thread.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| th_id          | int       | -             | Thread ID to suspend |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
def fn_th_func():
    if check_motion()==0:    # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)
#---- Main routine -----
th_id = thread_run(fn_th_func, loop=True)

# do something...

thread_pause(th_id) # Suspends the thread.
```

### 6.3.4 `thread_resume(th_id)`

#### ▪ Features

This function resumes a temporarily suspended thread.

#### ▪ Parameters

| Parameter Name     | Data Type        | Default Value | Description                       |
|--------------------|------------------|---------------|-----------------------------------|
| <code>th_id</code> | <code>int</code> | -             | Suspended thread ID to be resumed |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

| Exception                                | Description                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | Program terminated forcefully      |

#### ▪ Example

```
def fn_th_func():
    if check_motion()==0:      # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

#---- Main routine -----
th_id = thread_run(fn_th_func, loop=True)

# do something...
thread_pause(th_id) # Suspends the thread.

# do something...
thread_resume(th_id) # Resumes the suspended thread.
```

### 6.3.5 thread\_state(th\_id)

- **Features**

This function checks the status of a thread.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                   |
|----------------|-----------|---------------|-------------------------------|
| th_id          | int       | -             | Thread ID to check the status |

- **Return**

| Value | Description            |
|-------|------------------------|
| 1     | RUN (TH_STATE_RUN)     |
| 2     | PAUSE (TH_STATE_PAUSE) |
| 3     | STOP (TH_STATE_STOP)   |

- **Exception**

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

- **Example**

```
def fn_th_func():
    if check_motion()==0:    # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

th_id = thread_run(fn_th_func, loop=True)
state1 = thread_state(th_id)

thread_pause(th_id)
state2 = thread_state(th_id)
```



### 6.3.6 Integrated example

This example explains how to use the thread.

- **Example 1: Thread example**

```
#----- thread 1: client comm. -----
def fn_th_client():
    global g_sock
    global g_cmd
    res, rx_data = client_socket_read(g_sock)
    if res > 0:
        g_cmd = rx_data.decode() #decode: Converts byte type into a string.
    else: # Communication error
        client_socket_close(g_sock)
        exit() # Terminates the program.
    wait(0.1)
    return 0

#----- thread 2: check IO -----
def fn_th_check_io():
    if get_digital_input(1) == ON:
        exit() # Terminates the program.
    wait(0.1)
    return 0

#----- main -----
g_sock = client_socket_open("192.168.137.2", 20002) # Connects to the server.
g_cmd = ""

g_th_id1 = thread_run(th_client, loop=True) # Runs the th_client thread.
g_th_id2 = thread_run(th_check_io, loop=True) # Runs the th_check_io thread.

p1 = posj(0, 0, 90, 0, 90, 0)
p2 = posj(10, 0, 90, 0, 90, 0)
p3 = posj(20, 0, 90, 0, 90, 0)

while 1:
    if g_cmd == "a":
        g_cmd = ""
        movej(p1, vel=100, acc=100)
        client_socket_write(g_sock, b"end")
    if g_cmd == "b":
        g_cmd = ""
        movej(p2, vel=100, acc=100)
        client_socket_write(g_sock, b"end")
    if g_cmd == "c":
        g_cmd = ""
        movej(p3, vel=100, acc=100)
        client_socket_write(g_sock, b"end")
    wait(0.1)
```

## Thread

---

th\_client thread: Converts the data received from the server into a string and saves it in g\_cmd.

th\_check\_io thread: Checks the state of contact no. 1 and terminates the program if it is ON.

main: Connects to the server.

2 threads run: th\_client and th\_check\_io

If "a" is received from the server, it moves to p1 and sends "end" to the servers.

If "b" is received from the server, it moves to p2 and sends "end" to the servers.

If "c" is received from the server, it moves to p3 and sends "end" to the servers.

## 6.4 Others

### 6.4.1 wait(time)

#### ▪ Features

This function waits for the specified time.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| Time           | Float     | -             | Time [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

```
wait(1.3) # Waits for 1.3 seconds.

while 1: # Checks contact no. 1 every 0.1 second.
    if get_digital_input(1) == ON:
        set_digital_output(1, ON)
    wait(0.1)
```

## 6.4.2 **exit()**

- **Features**

This function terminates the currently running program.

- **Return**

| Value | Description |
|-------|-------------|
| 0     | Success     |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred |

- **Example**

```
exit()
```

### 6.4.3 sub\_program\_run(name)

#### ▪ Features

It executes a subprogram saved as a separate file.

#### ▪ Parameter

| Parameter Name | Data Type | Default Value | Description        |
|----------------|-----------|---------------|--------------------|
| name           | string    | -             | Name of subprogram |

#### ▪ Return

| Value  | Description                          |
|--------|--------------------------------------|
| module | Module object of executed subprogram |

#### ▪ Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

#### Note

- The first line of the subprogram must have the phrase "from DRCF import \*".
- When programming with a teaching pendant, this phrase is automatically inserted.
- The functions and global variables of the main program and sub-program can access each other. The functions and global variables of the sub-program can be accessed from the main program, and the functions and global variables of the main program can be accessed from the sub-program. Refer to Example 2 below.
- Note that even if the global variables of the main program and sub-program have identical names, they are executed as separate variables. In this case, they cannot be referenced by each other.
- If the functions of the main program and sub-program have identical names, the function of whichever file is called according to the scope rule is called.

### ▪ Example

# subprogramA and subprogramB must be created and saved in advance.

**<subProgramA.drl>**

```
from DRCF import *  
movej([0,0,90,0,90,0], vel=30, acc=30)
```

**<subProgramB.drl>**

```
from DRCF import *  
movej([10,0,90,0,90,0], vel=30, acc=30)
```

**<main program>**

```
while True:  
    var_select = tp_get_user_input("Select File", DR_VAR_INT)  
    if var_select == 0:  
        sub_program_run("subProgramA") # execute subProgramA  
    elif var_select == 1:  
        sub_program_run("subProgramB") # execute subProgramB
```

### 6.4.4 drl\_report\_line(option)

#### ▪ Features

This command is used to turn ON / OFF the execution line display function when the DRL script is running. When the run line display function is turned OFF, the time required to execute the run line display function is reduced, which significantly speeds up the execution of the DRL.

#### Caution

The following features do not operate in the section where the execution line display function is turned OFF.

- Execution time display by line
- Variable monitoring
- System Variable Update
- Step by Step in Debug mode
- Brake Point in Debug mode

#### ▪ Parameter

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| option         | Int       | -             | Whether to display the DRL execution line<br>ON(1)<br>OFF(0) |

#### ▪ Return

| Value | Description |
|-------|-------------|
| None  | None        |

#### ▪ Example

```
x=0
y=0

drl_report_line(OFF) # Execution line display function OFF
while x < 1000:     # Execution line not displayed (speed up execution)
  x += 1           # Execution line not displayed (speed up execution)
drl_report_line(ON) # Execution line display function ON
x=0                # Execution line shown
y=0                # Execution line shown
```

### 6.4.5 set\_fm(key, value)

#### ▪ Features

This command is used when interworking is required for information on variables (global variables, system variables, etc.) created when the program is executed, in addition to the system information already defined and linked with KT Smart Factory.

#### Caution

Please note that this function will not work if the linkage information is not set in the KT Smart Factory menu in the Setup menu.

The KT Smart Factory menu only appears when setting up KT-specific licenses.

#### ▪ Parameter

| Parameter Name | Data Type              | Default Value | Description  |
|----------------|------------------------|---------------|--|
| key            | string                 | -             | Data Name  |
| value          | int<br>float<br>string | -             | Interlocking Data Variable<br>Possible data types<br>.Integer data<br>.Real data<br>.string data |

#### ▪ Return

| Value | Description |
|-------|-------------|
| None  | None        |

#### ▪ Example

```
count = 0

movej(posj(0, 0, 90, 0,90,0), vel=30, acc=30)
while True:
    movej(posj(0, 0, -90, 0,90,0), vel=30, acc=30)
    movej(posj(0, 0, 90, 0,90,0), vel=30, acc=30)
    count = count + 1
    set_fm("TotalCount", count)
```



## 7. Mathematical Function

### 7.1 $\sin(x)$

- **Features**

This function returns the sine value of x radians.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

- **Return**

| Value         | Description |
|---------------|-------------|
| the sine of x | -           |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.2 cos(x)

### ▪ Features

This function returns the cosine value of x radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

### ▪ Return

| Value           | Description |
|-----------------|-------------|
| the cosine of x |             |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.3 $\tan(x)$

### ▪ Features

This function returns the tangent value of  $x$  radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| $x$            | float     | -             | -           |

### ▪ Return

| Value              | Description |
|--------------------|-------------|
| the tangent of $x$ | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.4 asin(x)

### ▪ Features

This function returns the arc sine value of x radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             |             |

### ▪ Return

| Value             | Description |
|-------------------|-------------|
| the arc sine of x | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.5 **acos(x)**

### ▪ Features

This function returns the arc cosine value of x radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             |             |

### ▪ Return

| Value               | Description |
|---------------------|-------------|
| the arc cosine of x | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.6 atan(x)

### ▪ Features

This function returns the arc tangent value of x radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

### ▪ Return

| Value                | Description |
|----------------------|-------------|
| the arc tangent of x | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.7 atan2(y, x)

### ▪ Features

This function returns the arc tangent value of  $y/x$  radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| y              | float     | -             | -           |
| x              | float     | -             | -           |

### ▪ Return

| Value                    | Description                            |
|--------------------------|--|
| the arc tangent of $y/x$ | The result is between $-\pi$ and $\pi$ |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.8 ceil(x)

- **Features**

This returns the smallest integer larger than or equal to x. It truncates up to the integer.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

- **Return**

| Value           | Description |
|-----------------|-------------|
| rounded integer | -           |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |



## 7.9 floor(x)

### ▪ Features

This returns the largest integer smaller than or equal to x. It rounds down to the nearest one.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

### ▪ Return

| Value           | Description |
|-----------------|-------------|
| rounded integer | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.10 pow(x, y)

- **Features**

Return x raised to the power of y.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             |             |
| y              | float     | -             |             |

- **Return**

| Value                   | Description |
|-------------------------|-------------|
| x raised to the power y | -           |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.11 `sqrt(x)`

### ▪ Features

This function returns the square root of x.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| x              | float     | -             | -           |

### ▪ Return

| Value                | Description |
|----------------------|-------------|
| the square root of x | Success     |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.12 log(x, b)

- **Features**

This function returns the log of x with base b.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                 |
|----------------|-----------|---------------|-----------------------------|
| x              | float     | -             | -                           |
| b              | float     | -             | base, e (natural logarithm) |

- **Return**

| Value                                | Description |
|--------------------------------------|-------------|
| the logarithm of f to the base of b. | -           |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.13 d2r(x)

### ▪ Features

This returns x degrees as radians.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| x              | float     | -             | The angle in degrees |

### ▪ Return

| Value                | Description |
|----------------------|-------------|
| The angle in radians | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.14 r2d(x)

### ▪ Features

This function returns the x radians value to degrees.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| x              | float     |               | The angle in radians |

### ▪ Return

| Value                | Description |
|----------------------|-------------|
| The angle in degrees | -           |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.15 `norm(x)`

- **Features**

This function returns the L2 norm of x.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                |
|----------------|-----------|---------------|----------------------------|
| x              | float[3]  | -             | Point coordinate (x, y, z) |

- **Return**

| Value | Description                         |
|-------|-------------------------------------|
| float | Size of the point coordinate vector |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

## 7.16 random()

- **Features**

This function returns a random number between 0 and 1.

- **Return**

| Value         | Description                           |
|---------------|---------------------------------------|
| random number | Random number between 0 and 1 (float) |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |



## 7.17 `rotx(angle)`

### ▪ Features

This function returns a rotation matrix that rotates by the angle value along the x-axis.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| angle          | float     | 0             | Rotating angle [deg] |

### ▪ Return

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotm = rotx(30)
```

## 7.18 roty(angle)

- **Features**

This function returns a rotation matrix that rotates by the angle value along the y-axis.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| angle          | float     | 0             | Rotating angle [deg] |

- **Return**

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
rotm = roty(30)
```

## 7.19 rotz(angle)

### ▪ Features

This function returns a rotation matrix that rotates by the angle value along the z-axis.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description          |
|----------------|-----------|---------------|----------------------|
| angle          | float     | 0             | Rotating angle [deg] |

### ▪ Return

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotm = rotz(30)
```

## 7.20 rotm2eul(rotm)

### ▪ Features

This function receives a rotation matrix and returns the Euler angle (zyz order) to degrees. Of the Euler angle (rx, ry, rz) returned as a result, ry is always a positive number.

### ▪ Parameters

| Parameter Name | Data Type   | Default Value | Description     |
|----------------|-------------|---------------|-----------------|
| rotm           | Float[3][3] | -             | Rotation matrix |

### ▪ Return

| Value    | Description           |
|----------|-----------------------|
| float[3] | ZYZ Euler angle [deg] |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotm = [[1,0,0],[0,0.87,-0.5],[0,0.5,0.87]]  
eul = rotm2eul(rotm)
```

## 7.21 rotm2rotvec(rotm)

### ▪ Features

This function receives a rotation matrix and returns the rotation vector (angle/axis representation).

### ▪ Parameters

| Parameter Name | Data Type   | Default Value | Description     |
|----------------|-------------|---------------|-----------------|
| rotm           | float[3][3] | -             | Rotation matrix |

### ▪ Return

| Value    | Description     |
|----------|-----------------|
| float[3] | rotation vector |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotm = [[1,0,0],[0,0.87,-0.5],[0,0.5,0.87]]
eul = rotm2rotvec(rotm)
```

## 7.22 eul2rotm([alpha,beta,gamma])

- **Features**

This function transforms a Euler angle (zyz order) to a rotation matrix.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description             |
|----------------|-----------|---------------|-------------------------|
| eul            | float[3]  | [0 0 0]       | Euler angle (zyz) [deg] |

- **Return**

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
eul = [90, 90, 0]
rotm = eul2rotm (eul)
```

## 7.23 eul2rotvec([alpha,beta,gamma])

### ▪ Features

This function transforms a Euler angle (zyz order) to a rotation vector.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description             |
|----------------|-----------|---------------|-------------------------|
| eul            | float[3]  | [0 0 0]       | Euler angle (zyz) [deg] |

### ▪ Return

| Value    | Description     |
|----------|-----------------|
| float[3] | rotation vector |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
eul = [90, 90, 0]
rotvec = eul2rotvec (eul)
```

## 7.24 rotvec2eul([rx,ry,rz])

### ▪ Features

This function transforms a rotation vector to a Euler angle (zyz).

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description     |
|----------------|-----------|---------------|-----------------|
| rotvec         | float[3]  | -             | rotation vector |

### ▪ Return

| Value    | Description           |
|----------|-----------------------|
| float[3] | ZYZ Euler angle [deg] |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotvec = [0.7854, 0, 0]  
eul = rotvec2eul(rotvec) # eul=[45,0,0]
```



## 7.25 rotvec2rotm([rx,ry,rz])

### ▪ Features

This function transforms a rotation vector to a rotation matrix.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description     |
|----------------|-----------|---------------|-----------------|
| rotvec         | float[3]  | -             | rotation vector |

### ▪ Return

| Value       | Description     |
|-------------|-----------------|
| float[3][3] | Rotation matrix |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
rotm = rotvec2eul([0.7854,0,0])
```

## 7.26 htrans(posx1,posx2)

### ▪ Features

This function returns the pose corresponding to T1\*T2 assuming that the homogeneous transformation matrices obtained from posx1 and posx2 are T1 and T2, respectively.

$$H_1H_2 = \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1R_2 & r_1 + R_1r_2 \\ 0 & 1 \end{bmatrix}$$

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

### ▪ Return

| Value | Description |
|-------|-------------|
| posx  | [mm, deg]   |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
posx = htrans(posx1,posx2)
```

## 7.27 `get_intermediate_pose(posx1,posx2,alpha)`

### ▪ Features

This function returns `posx` located at `alpha` of the linear transition from `posx1` to `posx2`. It returns `posx1` if `alpha` is 0, the median value of two poses if `alpha` is 0.5, and `posx2` if `alpha` is 1.

### ▪ Parameters

| Parameter Name     | Data Type                            | Default Value | Description                                     |
|--------------------|--------------------------------------|---------------|---|
| <code>posx1</code> | <code>posx</code><br>list (float[6]) | -             | <code>posx</code> or<br>position list [mm, deg] |
| <code>posx2</code> | <code>posx</code><br>list (float[6]) | -             | <code>posx</code> or<br>position list [mm, deg] |
| <code>alpha</code> | float                                | -             | $0.0 \leq \alpha \leq 1.0$                      |

### ▪ Return

| Value             | Description |
|-------------------|-------------|
| <code>posx</code> | [mm, deg]   |

### ▪ Exception

| Exception                             | Description                        |
|---------------------------------------|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code> | Parameter data type error occurred |

### ▪ Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
alpha = 0.5
posx = get_intermediate_pose(posx1,posx2,alpha)
```

## 7.28 get\_distance(posx1, posx2)

- **Features**

This function returns the distance between two pose positions in [mm].

- **Parameters**

| Parameter Name | Data Type               | Default Value | Description                   |
|----------------|-------------------------|---------------|-------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm] |

- **Return**

| Value | Description |
|-------|-------------|
| float | [mm]        |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
dis_posx = get_distance(posx1, posx2)
```

## 7.29 `get_normal(x1, x2, x3)`

### ▪ Features

This function returns the normal vector of a surface consisting of three points (`posx`) in the task space. This direction is clockwise.

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description              |
|----------------|-------------------------|---------------|--------------------------|
| x1             | posx<br>list (float[6]) | -             | posx or<br>position list |
| x2             | posx<br>list (float[6]) | -             | posx or<br>position list |
| x3             | posx<br>list (float[6]) | -             | posx or<br>position list |

### ▪ Return

| Value    | Description   |
|----------|---------------|
| float[3] | normal vector |

### ▪ Exception

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

### ▪ Example

```
x1 = posx(0, 500, 700, 30, 0, 90)
x2 = posx(500, 0, 700, 0, 0, 45)
x3 = posx(300, 100, 500, 45, 0, 45)
vect = get_normal(x1, x2, x3)
```

### 7.30 add\_pose(posx1, posx2)

▪ **Features**

This function obtains the sum of two poses.

$$\text{add\_pose}\left(\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} R_1 R_2 & r_1 + r_2 \\ 0 & 1 \end{bmatrix}$$

▪ **Parameters**

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

▪ **Return**

| Value | Description |
|-------|-------------|
| posx  | [mm, deg]   |

▪ **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

▪ **Example**

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
add_posx = add_pose(posx1, posx2)
```

### 7.31 `subtract_pose(posx1, posx2)`

#### ▪ Features

This function obtains the difference between two poses.

$$\text{subtract\_pose}\left(\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} R_2^T R_1 & r_1 - r_2 \\ 0 & 1 \end{bmatrix}$$

#### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

#### ▪ Return

| Value | Description |
|-------|-------------|
| posx  | [mm, deg]   |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
subtract_posx = subtract_pose(posx1, posx2)
```

## 7.32 inverse\_pose(posx1)

### ▪ Features

This function returns the posx value that represents the inverse of posx.

$$\text{inv\_pose} \left( \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_1^T & -R_1^T r_1 \\ 0 & 1 \end{bmatrix}$$

### ▪ Parameters

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

### ▪ Return

| Value | Description |
|-------|-------------|
| posx  | [mm, deg]   |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

### ▪ Example

```
posx1 = [100, 20, 300, 90, 0, 180]
inv_posx = inverse_pose(posx1)
```



### 7.33 dot\_pose(posx1, posx2)

- **Features**

This function obtains the inner product of the translation component when two poses are given.

- **Parameters**

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

- **Return**

| Value | Description                 |
|-------|-----------------------------|
| float | Inner product of two poses. |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
res= dot_pose(posx1, posx2)
```

### 7.34 cross\_pose(posx1, posx2)

▪ **Features**

This function obtains the outer product of the translation component when two poses are given.

▪ **Parameters**

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |
| posx2          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

▪ **Return**

| Value    | Description                 |
|----------|-----------------------------|
| float[3] | Outer product of two poses. |

▪ **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

▪ **Example**

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
res= cross_pose(posx1, posx2)
```

### 7.35 `unit_pose(posx1)`

- **Features**

This function obtains the unit vector of the given posx translation component.

- **Parameters**

| Parameter Name | Data Type               | Default Value | Description                        |
|----------------|-------------------------|---------------|------------------------------------|
| posx1          | posx<br>list (float[6]) | -             | posx or<br>position list [mm, deg] |

- **Return**

| Value    | Description                   |
|----------|-------------------------------|
| float[3] | Unit vector of the given posx |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
posx1 = [100, 20, 300, 90, 0, 180]
res = unit_pose(posx1)
```

## 8. External Communication Commands

### 8.1 Serial

#### 8.1.1 `serial_open(port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)`

##### ▪ Features

This function opens a serial communication port.

##### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| port           | string    | None          | E.g.) "COM1", "COM2"   |
| baudrate       | int       | 115200        | Baud rate  |
| bytesize       | int       | 8             | Number of data bits<br><ul style="list-style-type: none"> <li>• DR_FIVEBITS: 5</li> <li>• DR_SIXBITS: 6</li> <li>• DR_SEVENBITS: 7</li> <li>• DR_EIGHTBITS: 8</li> </ul>   |
| parity         | str       | "N"           | Parity checking<br><ul style="list-style-type: none"> <li>• DR_PARITY_NONE: "N"</li> <li>• DR_PARITY_EVEN: "E"</li> <li>• DR_PARITY_ODD: "O"</li> <li>• DR_PARITY_MARK: "M"</li> <li>• DR_PARITY_SPACE: "S"</li> </ul> |
| stopbits       | int       | 1             | Number of stop bits<br><ul style="list-style-type: none"> <li>• DR_STOPBITS_ONE = 1</li> <li>• DR_STOPBITS_ONE_POINT_FIVE = 1.5</li> <li>• DR_STOPBITS_TWO = 2</li> </ul>  |

##### ▪ Return

| Value                  | Description           |
|------------------------|-----------------------|
| serial.Serial instance | Successful connection |

##### ▪ Exception

| Exception                   | Description                           |
|-----------------------------|---------------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred    |
| DR_Error (DR_ERROR_VALUE)   | Parameter value is invalid            |
| DR_Error (DR_ERROR_RUNTIME) | Serial.SerialException error occurred |

- **Example**

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,  
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
```

```
if type(ser) == serial.Serial:  
    serial_write(ser, b"123456789")
```

```
serial_close(ser)
```

### 8.1.2 serial\_close(*ser*)

#### ▪ Features

This function closes a serial communication port.

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description     |
|----------------|---------------|---------------|-----------------|
| <i>ser</i>     | serial.Serial | -             | Serial instance |

#### ▪ Return

| Value | Description                         |
|-------|-------------------------------------|
| 0     | Successful closing of a serial port |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,  
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
```

```
if type(ser) == serial.Serial:  
    serial_write(ser, b"123456789")
```

```
serial_close(ser)
```

### 8.1.3 `serial_state(ser)`

- **Features**

This function returns the status of a serial communication port.

- **Parameters**

| Parameter Name | Data Type     | Default Value | Description     |
|----------------|---------------|---------------|-----------------|
| ser            | serial.Serial | -             | Serial instance |

- **Return**

| Value | Description        |
|-------|--------------------|
| 1     | Serial port opened |
| 0     | Serial port closed |

- **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

- **Example**

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,
                 parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

state = serial_state(ser)
serial_close(ser)
```

### 8.1.4 serial\_set\_inter\_byte\_timeout(*ser*, *timeout=None*)

#### ▪ Features

This function sets the timeout between the bytes (inter-byte) when reading and writing to the port.

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description  |
|----------------|---------------|---------------|--|
| <i>ser</i>     | serial.Serial | -             | Serial instance  |
| <i>timeout</i> | float         | None          | Timeout between bytes during reading or writing <ul style="list-style-type: none"> <li>Continued processing of data that was processed before the timeout</li> <li>None: inter-byte timeout not specified</li> </ul> |

#### ▪ Return

| Value | Description |
|-------|-------------|
| 0     | Success     |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,
                 parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

state = serial_set_inter_byte_timeout(ser, 0.1)

serial_close(ser)
```



### 8.1.5 `serial_write`(ser, tx\_data)

#### ▪ Features

This function records the data (tx\_data) to a serial port.

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| ser            | serial.Serial | -             | Serial instance   |
| tx_data        | byte          | -             | Data to be transmitted <ul style="list-style-type: none"> <li>• The data type must be a byte.</li> <li>• Refer to the example below.</li> </ul> |

#### ▪ Return

| Value | Description                           |
|-------|---------------------------------------|
| 0     | Success                               |
| -1    | The port is not open.                 |
| -2    | serial.SerialException error occurred |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,
    parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

serial_write(ser, b"123456789") # b means the byte type.

serial_close(ser)
```

### 8.1.6 serial\_read(*ser*, *length*=-1, *timeout*=-1)

#### ▪ Features

This function reads the data from a serial port.

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| <i>ser</i>     | serial.Serial | -             | Serial instance   |
| <i>length</i>  | int           | -1            | Number of bytes to read <ul style="list-style-type: none"> <li>-1: Not specified (The number of bytes to read is not specified.)</li> <li><math>n(\geq 0)</math>: The specified number of bytes is read.</li> </ul> |
| <i>timeout</i> | int<br>float  | -1            | Read waiting time <ul style="list-style-type: none"> <li>-1: Indefinite wait</li> <li><math>n(&gt;0)</math>: <math>n</math> seconds</li> </ul>  |

#### ▪ Return

| Value ( <i>res</i> , <i>rx_data</i> ) | Description                      |                                       |
|---------------------------------------|----------------------------------|---------------------------------------|
| <i>res</i>                            | 0                                | Number of bytes of the received data  |
|                                       | -1                               | The port is not open.                 |
|                                       | -2                               | serial.SerialException error occurred |
| <i>rx_data</i>                        | Number of bytes read (byte type) |                                       |

#### ▪ Exception

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

- **Example**

```
ser = serial_open(port="COM2", baudrate=115200, bytesize=DR_EIGHTBITS,  
                 parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)  
serial_read(ser)  
serial_read(ser, 100)  
serial_read(ser, 100, 3)  
serial_read(ser, -1, 3)  
serial_close(ser)
```

### 8.1.7 serial\_get\_count()

- **Features**

This function reads the number of devices connected to USB to Serial.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| None           | -         | -             |             |

- **Return**

| Value (port_info, device_name) | Description                      |
|--------------------------------|----------------------------------|
| count                          | Number of connected serial ports |

- **Exception**

| Exception                 | Description                      |
|---------------------------|----------------------------------|
| DR_Error (DR_ERROR_TYPE)  | Parameter data error occurred    |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value occurred |

- **Example**

```
count = serial_get_count() # read number of connected serial ports

for i in range(count):
    port_info, device_name = serial_get_info(i+1)
    tp_popup("i={}, port ={}, dev ={}".format(i, port_info, device_name))
```

8.1.8 **serial\_get\_info(id)**▪ **Features**

This function reads the port information and device name of the connected USB to Serial.

▪ **Parameter**

| Parameter Name | Data Type | Default Value | Description                          |
|----------------|-----------|---------------|--------------------------------------|
| id             | int       | 1             | ID of "USB to Serial" to read (1-10) |

▪ **Return**

| Value (port_info, device_name) | Description  |
|--------------------------------|--|
| port_info                      | Port information (NULL means no device is connected) |
| device_name                    | Device name (NULL means no device is connected)      |

▪ **Exception**

| Exception                 | Description                      |
|---------------------------|----------------------------------|
| DR_Error (DR_ERROR_TYPE)  | Parameter data error occurred    |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value occurred |

▪ **Example**

```
port_info, device_name = serial_get_info(1) #1 connected device information
#port_info = "COM_USB"
ser = serial_open(port=port_info, baudrate=115200, bytesize=DR_EIGHTBITS,
    parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
```

### 8.1.9 Combined Example

This is an example for performing a self-loop-back test on RXD (#2 pin) and TXD (#3 pin) are connected with the serial port.

- **Example 1: Self-loop-back test example**

```
# serial port open
# if D-SUB (9pin) is connected          : port="COM"
# if USB is connected with USB to Serial : port="COM_USB"
ser = serial_open(port="COM_USB", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
wait(1)

# SEND DATA : "123ABC"
res = serial_write(ser, b"123ABC") # b means byte type
wait(1)

# READ DATA
res, rx_data = serial_read(ser)
# RXD and TXD are H/W connected res=6 (byte) rx_data = b"123ABC" are received

tp_popup("res ={0}, rx_data={1}".format(res, rx_data))

# close corresponding serial port
serial_close(ser)
```

Received data is collected as is and the result is outputted as a TP pop-up message. If executed properly, it outputs a result of res=6 rx\_data = b'123ABC'.

## 8.2 Tcp/Client

### 8.2.1 client\_socket\_open(ip, port)

#### ▪ Features

This function creates a socket and attempts to connect it to a server (ip, port). It returns the connected socket when the client is connected.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                                 |
|----------------|-----------|---------------|---|
| ip             | str       | -             | Server IP address: (E.g.) "192.168.137.200" |
| port           | int       | -             | Server port number (e.g.) 20002             |

#### ▪ Return

| Value                  | Description           |
|------------------------|-----------------------|
| socket.socket instance | Successful connection |

#### ▪ Exception

| Exception                   | Description                               |
|-----------------------------|---|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred        |
| DR_Error (DR_ERROR_VALUE)   | Parameter value is invalid                |
| DR_Error (DR_ERROR_RUNTIME) | socket.error occurred during a connection |

#### ▪ Example

```
sock = client_socket_open("192.168.137.200", 20002)
# An indefinite connection is attempted to the server (ip="192.168.137.200",
port=20002).
# The connected socket is returned if the connection is successful.
# The data is read, written, and closed using the returned socket as shown below.

client_socket_write(sock, b"123abc") # Sends data to the server (b represents the
byte type).
res, rx_data = client_socket_read(sock) # Receives the data from the server.
client_socket_close(sock) # Closes the connection to the server.
```

## 8.2.2 client\_socket\_close(sock)

- **Features**

This function terminates communication with the server. To reconnect to the server, the socket must be closed with client\_socket\_close(sock) and reopened.

- **Parameters**

| Parameter Name | Data Type     | Default Value | Description  |
|----------------|---------------|---------------|--|
| sock           | socket.socket | -             | Socket instance returned from client_socket_open() |

- **Return**

| Value | Description              |
|-------|--------------------------|
| 0     | Successful disconnection |

- **Exception**

| Exception                   | Description                                |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred         |
| DR_Error (DR_ERROR_RUNTIME) | socket.error occurred during disconnection |

- **Example**

```
sock = client_socket_open("192.168.137.200", 20002)
# An indefinite connection is attempted to the server (ip="192.168.137.200",
port=20002).

# do something...

client_socket_close(sock)          # Closes the connection to the server.
```



### 8.2.3 client\_socket\_state(sock)

#### ▪ Features

This function returns the socket connection status. This checks the return value of client\_socket\_read or client\_socket\_write to identify the server connection status. (refer to Example 2)

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description  |
|----------------|---------------|---------------|--|
| sock           | socket.socket | -             | Socket instance returned from client_socket_open() |

#### ▪ Return

| Value | Description            |
|-------|------------------------|
| 1     | Connected state        |
| 0     | Abnormal Socket Status |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example1

```
sock = client_socket_open("192.168.137.200", 20002)

state = client_socket_state(sock) # Reads the socket state.

client_socket_close(sock)
```

#### ▪ 예제 2

```
sock = client_socket_open("192.168.137.200", 20002)

res, rx_data = client_socket_read(sock)
tp_log("[RX] res={0}, rx_data = {1}".format(res, rx_data))
if (res < 0):
    tp_log("[RX] server disconnect") #If communication with the server is lost
    client_socket_close(sock)
    exit()
```

```
client_socket_close(sock)
```

### 8.2.4 client\_socket\_write(sock, tx\_data)

▪ **Features**

This function transmits data to the server.

▪ **Parameters**

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| sock           | socket.socket | -             | Socket instance returned from client_socket_open()  |
| tx_data        | byte          | -             | Data to be transmitted <ul style="list-style-type: none"> <li>• The data type must be a byte.</li> <li>• Refer to the example below.</li> </ul> |

▪ **Return**

| Value | Description   |
|-------|---|
| 0     | Success   |
| -1    | The server is not connected.  |
| -2    | The server connection is lost or a socket.error exception occurred during transmission processing |

▪ **Exception**

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

▪ **Example**

```
sock = client_socket_open("192.168.137.200", 20002)

client_socket_write(sock, b"1234abcd") # b means the byte type.

msg = "abcd" # msg is a string variable.
client_socket_write(sock, msg.decode()) # decode() converts a string type to a byte type.

client_socket_close(sock)
```

## 8.2.5 `client_socket_read(sock, length=-1, timeout=-1)`

### ▪ Features

This function receives data from the server.

### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| sock           | socket.socket | -             | Socket instance returned from <code>client_socket_open()</code>   |
| length         | int           | -1            | Number of bytes of the received data <ul style="list-style-type: none"> <li>-1: Not specified (The number of bytes to read is not specified.)</li> <li>n(&gt;=0): The specified number of bytes is read.</li> </ul> |
| timeout        | int<br>float  | -1            | Waiting time for receipt <ul style="list-style-type: none"> <li>-1: Indefinite wait</li> <li>n(&gt;0): n seconds</li> </ul>   |

### ▪ Return

| Value (res, rx_data) | Description               |   |
|----------------------|---------------------------|---|
| res                  | >0                        | Number of bytes of the received data        |
|                      | -1                        | The server is not connected.                |
|                      | -2                        | socket.error occurred during data reception |
|                      | -3                        | Timeout during data reception               |
| rx_data              | Received data (byte type) |   |

### ▪ Exception

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

### ▪ Example

```
sock = client_socket_open("192.168.137.200", 20002)

res, rx_data = client_socket_read(sock)    # Indefinite wait until the data is received
# Reads all received data since the length is omitted.
# Waits indefinitely until the data is received since timeout is omitted.
# (res = size of received data, rx_data=received data) is returned when the data is
received.

res, rx_data = client_socket_read(sock, timeout=3) # Waits for up to 3 seconds until the
data is received.
# (res = size of received data, rx_data=received data) is returned if the data is received
within 3 seconds.
# (res = -3, rx_data=None) is returned if the data is not received within 3 seconds.

res, rx_data = client_socket_read(sock, length=64)    # Reads 64 bytes of the received
data.

res, rx_data = client_socket_read(sock, length=64, timeout=3)
# Reads 64 bytes of the received data within the 3-second timeout.

rx_msg = rx_data.decode() # rx_data is a byte type and can be converted to a string
type
                                # using decode().
                                # For example, if rx_data = b"abcd",
                                # rx_msg= "abcd".

client_socket_close(sock)
```

## 8.2.6 Integrated example

Assume that server IP = 192.168.137,200 and open port =20002 and that the received packets are sent to the client as they are (mirroring).

### ▪ Example 1: Example of a default TCP client

```
# Assume server IP = 192.168.137,200 and open port =20002.
g_sock = client_socket_open("192.168.137.200", 20002)

tp_popup("connect O.K!",DR_PM_MESSAGE)
while 1:
    client_socket_write(g_sock, b"abcd") # The string "abcd" is sent in a byte type.
    wait(0.1)
    res, rx_data = client_socket_read(g_sock) # Waits for the data from the server.
    tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
    wait(0.1)
```

The example connects to the server and sends the string "abcd". (b converts the string to a byte type.)

The message received from the server is output to the TP.

res = 4 and rx\_data=b"abcd" since the server transmits the received data as is.

### ▪ Example 2: Examples of a packet transfer

```
Transmission packet: "MEAS_START" +data1[4byte]+data2[4byte]
    data1: Conversion of the integer to 4 byte. ex) 1 → 00000001
    data2: Conversion of the integer to 4 byte. ex) 2 → 00000002
ex) data1=1 and data2=2: "MEAS_START"+00000001+00000002
    Actual packet: 4D4541535F53544152540000000100000002
Received packet: res=18, rx_data="MEAS_START"+00000001+00000002
    rxd1 extraction: Conversion of 10th - 14th bytes to an integer
    rxd2 extraction: Conversion of 14th - 18th bytes to an integer
```

```
g_sock = client_socket_open("192.168.137.100", 20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

send_data = b"MEAS_START"
data1 =1
data2 =2
send_data += (data1).to_bytes(4, byteorder='big')
send_data += (data2).to_bytes(4, byteorder='big')

client_socket_write(g_sock, send_data)

wait(0.1)

res, rx_data = client_socket_read(g_sock)
tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)

rxd1 = int.from_bytes(rx_data[10:10+4], byteorder='big', signed=True)
rxd2 = int.from_bytes(rx_data[14:14+4], byteorder='big', signed=True)
```

```
tp_popup("res={0}, rxd1={1}, rxd2={2}".format(res, rxd1, rxd2), DR_PM_MESSAGE)

client_socket_close(g_sock)
```

The example connects to the server and sends a byte type `send_data`.  
`res = 18` and `rx_data=send_data` since the server transmits the received data as is.  
`rx_d1` extraction: Conversion of 10th - 14th bytes to an integer  
`rx_d2` extraction: Conversion of 14th - 18th bytes to an integer  
The final result is `res=18`, `rx_d1=1`, and `rx_d2=2`.

### ▪ Example 3: Reconnection

```
def fn_reconnect():
    global g_sock
    client_socket_close(g_sock)
    g_sock = client_socket_open("192.168.137.200", 20002)
    return

g_sock = client_socket_open("192.168.137.200", 20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

client_socket_write(g_sock, b"abcd")
wait(0.1)

while 1:
    res, rx_data = client_socket_read(g_sock)
    if res < 0:
        fn_reconnect()
    else:
        tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
        wait(0.1)
```

The example checks the return value of the `client_socket_read()` command.  
A negative value is returned if the connection to the server is terminated or there is a communication problem.  
The function `reconnect()` is called to attempt a reconnection if a negative value is returned.  
Note that the opened socket is closed when a reconnection is attempted.

## 8.3 Tcp/Server

### 8.3.1 server\_socket\_open(port)

#### ▪ Features

The robot controller creates a server socket and waits for the connection to the client. The connected socket is returned when the client is connected.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description         |
|----------------|-----------|---------------|---------------------|
| port           | int       | -             | Port number to open |

#### ▪ Return

| Value                  | Description           |
|------------------------|-----------------------|
| socket.socket instance | Successful connection |

#### ▪ Exception

| Exception                   | Description                               |
|-----------------------------|---|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred        |
| DR_Error (DR_ERROR_VALUE)   | Parameter value is invalid                |
| DR_Error (DR_ERROR_RUNTIME) | socket.error occurred during a connection |

#### ▪ Example

```

sock = server_socket_open(20002)
# Opens the port 20002 and waits until the client connects.
# The connected socket is returned if the connection is successful.
# The data is read, written, and closed using the returned socket as shown below.

server_socket_write(sock, b"123abc") # Sends data to the client (b represents the byte
type).
res, rx_data = server_socket_read(sock) # Receives the data from the client.

server_socket_close(sock) # Closes the connection to the client.

```

### 8.3.2 server\_socket\_close(sock)

▪ **Features**

This function terminates communication with the client. To reconnect to the client, the socket must be closed with server\_socket\_close(sock) and reopened.

▪ **Parameters**

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| sock           | socket.socket | -             | Socket instance returned from server_socket_open()<br>socket instance |

▪ **Return**

| Value | Description              |
|-------|--------------------------|
| 0     | Successful disconnection |

▪ **Exception**

| Exception                   | Description                                |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred         |
| DR_Error (DR_ERROR_RUNTIME) | socket.error occurred during disconnection |

▪ **Example**

```
sock = server_socket_open(20002)
# Opens the port 20002 and waits until the client connects.

# do something...

server_socket_close(sock) # Closes the connection to the client.
```



### 8.3.3 `server_socket_state(sock)`

#### ▪ Features

This function returns the socket connection status. This checks the return value of `server_socket_read` or `server_socket_write` to identify the client connection status. (refer to Example 2)

#### ▪ Parameters

| Parameter Name | Data Type     | Default Value | Description  |
|----------------|---------------|---------------|--|
| sock           | socket.socket | -             | Socket instance returned from <code>server_socket_open()</code><br>socket instance |

#### ▪ Return

| Value | Description        |
|-------|--------------------|
| 1     | Connected state    |
| 0     | Disconnected state |

#### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

#### ▪ Example1

```
sock = server_socket_open(20002)

state = server_socket_state(sock) # Reads the socket state.

server_socket_close(sock)
```

▪ 예제 2

```

sock = server_socket_open(20002)

res, rx_data =server_socket_read(sock)
tp_log("[RX] res={0}, rx_data ={1}".format(res, rx_data))
if (res < 0):    # If communication with the client is lost
    tp_log("[RX] client disconnect")
    server_socket_close(sock)
    exit()

server_socket_close(sock)
    
```

8.3.4 **server\_socket\_write(sock, tx\_data)**

▪ Features

This function transmits data to the client.

▪ Parameters

| Parameter Name | Data Type     | Default Value | Description   |
|----------------|---------------|---------------|---|
| sock           | socket.socket | -             | Socket instance returned from server_socket_open()<br>socket instance   |
| tx_data        | byte          | -             | Data to be transmitted <ul style="list-style-type: none"> <li>• The data type must be a byte.</li> <li>• Refer to the example below.</li> </ul> |

▪ Return

| Value | Description   |
|-------|---|
| 0     | Success   |
| -1    | The client is not connected.  |
| -2    | The client connection is lost or a socket.error exception occurred during transmission processing |

▪ Exception

| Exception | Description |
|-----------|-------------|
|-----------|-------------|

---

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

▪ **Example**

```
sock = server_socket_open(20002)

server_socket_write(sock, b"1234abcd") # b means the byte type.

msg = "abcd"      # msg is a string variable.
server_socket_write(sock, msg.decode()) # decode() converts a string type to a byte
type.

server_socket_close(sock)
```

### 8.3.5 server\_socket\_read(sock, length=-1, timeout=-1)

▪ **Features**

This function reads data from the client.

▪ **Parameters**

| Parameter Name | Data Type     | Default Value | Description  |
|----------------|---------------|---------------|--|
| sock           | socket.socket | -             | Socket instance returned from server_socket_open()<br>socket instance  |
| length         | int           | -1            | Number of bytes of the received data<br><ul style="list-style-type: none"> <li>-1: Not specified (The number of bytes to read is not specified.)</li> <li>n(&gt;=0): The specified number of bytes is read.</li> </ul> |
| timeout        | int<br>float  | -1            | Waiting time for receipt<br><ul style="list-style-type: none"> <li>-1: Indefinite wait</li> <li>n(&gt;0): n seconds</li> </ul>   |

▪ **Return**

| Value (res, rx_data) |    | Description                                 |
|----------------------|----|---|
| res                  | 0  | Number of bytes of the received data        |
|                      | -1 | The client is not connected.                |
|                      | -2 | socket.error occurred during data reception |
|                      | -3 | Timeout during data reception               |
| rx_data              |    | Received data (byte type)                   |

▪ **Example**

```

sock = server_socket_open(20002)

res, rx_data = server_socket_read(sock) # Indefinite wait until the data is received
# Reads all received data since the length is omitted.
# Waits indefinitely until the data is received since timeout is omitted.
# (res = size of received data, rx_data=received data) is returned when the data is
received.

res, rx_data = server_socket_read(sock, timeout=3) # Waits for up to 3 seconds until
the data is received.
# (res = size of received data, rx_data=received data) is returned if the data is received
within 3 seconds.
# (res = -3, rx_data=None) is returned if the data is not received within 3 seconds.
    
```

```
res, rx_data = server_socket_read(sock, length=64) # Reads 64 bytes of the
received data.

res, rx_data = server_socket_read(sock, length=64, timeout=3)
# Reads 64 bytes of the received data within the 3-second timeout.

rx_msg = rx_data.decode() # rx_data is a byte type and can be converted to a string
type
                        # using decode().
                        # For example, if rx_data = b"abcd",
                        # rx_msg= "abcd".

server_socket_close(sock)
```

### 8.3.6 Integrated example

The example assumes that the client connects to the controller with IP = 192,168,137.100 and port = 20002 and that the received packets are sent to the server as they are (mirroring).

- **Example 1: Default TCP server example**

```
g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

while 1:
    server_socket_write(g_sock, b"abcd") # The string "abcd" is sent in a byte type.
    wait(0.1)
    res, rx_data = server_socket_read(g_sock) # Waits for the data from the server.
    tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
    wait(0.1)
```

The example opens the port 20002 and waits until the client connects. It connects to the client and sends the string "abcd". The message received from the client is output to the TP. res = 4 and rx\_data=b"abcd" since the client transmits the received data as is.

- **Example 2: Examples of a packet transfer**

Transmission packet: "MEAS\_START" +data1[4byte]+data2[4byte]  
 data1: Conversion of the integer to 4 byte. ex) 1 → 00000001  
 data2: Conversion of the integer to 4 byte. ex) 2 → 00000002  
 ex) data1=1 and data2=2: "MEAS\_START"+00000001+00000002  
 Actual packet: 4D4541535F53544152540000000100000002  
 Received packet: res=18, rx\_data="MEAS\_START"+00000001+00000002  
 rxd1 extraction: Conversion of 10th - 14th bytes to an integer  
 rxd2 extraction: Conversion of 14th - 18th bytes to an integer

```
g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

send_data = b"MEAS_START"
data1 =1
data2 =2
send_data += (data1).to_bytes(4, byteorder='big')
send_data += (data2).to_bytes(4, byteorder='big')

server_socket_write(g_sock, send_data)

wait(0.1)

res, rx_data = server_socket_read(g_sock)
tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)

rxd1 = int.from_bytes(rx_data[10:10+4], byteorder='big', signed=True)
rxd2 = int.from_bytes(rx_data[14:14+4], byteorder='big', signed=True)
```

```
tp_popup("res={0}, rxd1={1}, rxd2={2}".format(res, rxd1, rxd2), DR_PM_MESSAGE)

server_socket_close(g_sock)
```

The example sends the byte type `send_data`.  
`res = 18` and `rx_data=send_data` since the client transmits the received data as is.  
`rx1` extraction: Conversion of 10th - 14th bytes to an integer  
`rx2` extraction: Conversion of 14th - 18th bytes to an integer  
The final result is `res=18`, `rx1=1`, and `rx2=2`.

### ▪ Example 3: Reconnection

```
def fn_reopen():
    global g_sock
    server_socket_close(g_sock)
    g_sock = server_socket_open(20002)
    return

g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

server_socket_write(g_sock, b"abcd")
wait(0.1)

while 1:
    res, rx_data = server_socket_read(g_sock)
    if res < 0:
        fn_reopen()
    else:
        tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
        wait(0.1)
```

The example checks the return value of the `server_socket_read()` command.  
A negative value is returned if the connection to the client is terminated or there is a communication problem.  
The function `reopen()` is called to wait for the client connection if a negative value is returned.  
Note that the opened socket is closed when a reconnection is attempted.

## 8.4 Modbus

### 8.4.1 `add_modbus_signal (ip, port, name, reg_type, index, value=0, slaveid=255)`

#### ▪ Features

This function registers the ModbusTCP signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| ip             | string    | -             | IP address of the ModbusTCP module   |
| port           | int       | -             | Port number of the ModbusTCP module  |
| name           | string    | -             | Modbus signal name   |
| reg_type       | int       | -             | Modbus signal type<br><ul style="list-style-type: none"> <li>• DR_MODBUS_DIG_INPUT</li> <li>• DR_MODBUS_DIG_OUTPUT</li> <li>• DR_MODBUS_REG_INPUT</li> <li>• DR_MODBUS_REG_OUTPUT</li> </ul> |
| 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           | <ul style="list-style-type: none"> <li>• Slave ID of the ModbusTCP module (0 or 1-247 or 255)</li> </ul> 0 : Broadcast address<br>255 : Default value for ModbusTCP                          |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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



| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| 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)
```

8.4.2 **add\_modbus\_rtu\_signal (slaveid=1, port=None, baudrate=115200, bytesize=DR\_EIGHTBITS, parity=DR\_PARITY\_NONE, stopbits=DR\_STOPBITS\_ONE, name, reg\_type, index, value=0)**

#### ▪ Features

This function registers the ModbusRTU signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| slaveid        | int       | 1             | <ul style="list-style-type: none"> <li>Slave ID of the ModbusRTU module (0 or 1-247)</li> <li>0 : Broadcast address</li> </ul>  |
| port           | string    | None          | E.g.) "USB_COM"   |
| baudrate       | int       | 115200        | Baud rate<br>9600, 19200, 57600, 115200, etc  |
| bytesize       | int       | 8             | Number of data bits <ul style="list-style-type: none"> <li>DR_FIVEBITS: 5</li> <li>DR_SIXBITS: 6</li> <li>DR_SEVENBITS: 7</li> <li>DR_EIGHTBITS: 8</li> </ul>                     |
| parity         | string    | "N"           | Parity checking <ul style="list-style-type: none"> <li>DR_PARITY_NONE: "N"</li> <li>DR_PARITY_EVEN: "E"</li> <li>DR_PARITY_ODD: "O"</li> </ul>                                    |
| stopbits       | int       | 1             | Number of stop bits <ul style="list-style-type: none"> <li>DR_STOPBITS_ONE = 1</li> <li>DR_STOPBITS_TWO = 2</li> </ul>  |
| name           | string    | -             | Modbus signal name  |
| reg_type       | int       | -             | Modbus signal type <ul style="list-style-type: none"> <li>DR_MODBUS_DIG_INPUT</li> <li>DR_MODBUS_DIG_OUTPUT</li> <li>DR_MODBUS_REG_INPUT</li> <li>DR_MODBUS_REG_OUTPUT</li> </ul> |
| index          | int       | -             | Modbus signal index   |
| value          | int       | 0             | Output when the type is DR_MODBUS_DIG_OUTPUT or DR_MODBUS_REG_OUTPUT (ignored otherwise)  |

- **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

- **Exception**

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

- **Example**

```
add_modbus_rtu_signal(slaveid=1, port=port_info, baudrate=115200,
bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE,
name='di1', reg_type=DR_MODBUS_REG_INPUT, index=0)
add_modbus_rtu_signal(slaveid=1, port=port_info, baudrate=115200,
bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE,
name='do1', reg_type=DR_MODBUS_REG_OUTPUT, index=0, value=12345)
```

### 8.4.3 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
```

### 8.4.4 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 <ul style="list-style-type: none"> <li>• ON : 1</li> <li>• OFF : 0</li> </ul> |
|                |           |               | 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)
```

### 8.4.5 `set_modbus_outputs(iobus_list, val_list)`

#### ▪ Features

This function sends multiple signals to the Modbus Slave unit.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                 |
|----------------|-----------|---------------|-----------------------------|
| iobus          | string    | -             | Modbus name (set in the TP) |
| value          | int       | -             | I/O output value list       |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failed      |

#### ▪ Exception

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

#### ▪ Example

```
# Modbus digital I/O output contact "d1" OFF, "d2" ON, and "d3" ON
set_modbus_outputs(iobus_list=[ "d1", "d2", "d3",], val_list=[0,1,1])
```

```
# Modbus digital I/O output contact "d3" OFF and "d4" ON
set_modbus_outputs(["d3", "d4"], [0,1])
```

### 8.4.6 get\_modbus\_input(iobus)

▪ **Features**

This function reads the signal from the Modbus Slave unit. Parameters

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

▪ **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")
```



### 8.4.7 `get_modbus_inputs(iobus_list)`

#### ▪ Features

This function reads multiple signals from the Modbus Slave unit.

#### ▪ Parameters

| Parameter Name          | Data Type                 | Default Value | Description   |
|-------------------------|---------------------------|---------------|---|
| <code>iobus_list</code> | <code>list(string)</code> | -             | Modbus input name list (set in the TP)<br>signal type can be used only in the following cases <ul style="list-style-type: none"> <li>• <code>DR_MODBUS_DIG_INPUT</code></li> <li>• <code>DR_MODBUS_DIG_OUTPUT</code></li> </ul> |

#### ▪ Return

| Value                     | Description   |
|---------------------------|---|
| <code>int (&gt;=0)</code> | Multiple signals to be read at once<br>(the value of the combination of <code>iobus_list</code> where the first value is LSB and the last value is MSB) |
| Negative number           | Failed  |

#### ▪ Exception

| Exception                                | Description                        |
|--|------------------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code>    | Parameter data type error occurred |
| <code>DR_Error (DR_ERROR_VALUE)</code>   | Parameter value is invalid         |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C extension module error occurred  |
| <code>DR_Error (DR_ERROR_STOP)</code>    | Program terminated forcefully      |

#### ▪ Example

```
# Modbus digital I/O input signal "di1" =OFF, "di2"=ON, and "di3" =ON
res = get_modbus_inputs(iobus_list=["di1", "di2", "di3"])
#res expected value = 0b110 (binary number), 6 (decimal number), or 0x06
(hexadecimal number)

# Modbus digital I I/O input signal "di4" OFF and "di5" ON
res = get_modbus_inputs(["di4", "di5"])
#res expected value = 0b10 (binary number), 2 (decimal number), or 0x02
(hexadecimal number)
```

### 8.4.8 get\_modbus\_inputs\_list(iobus\_list)

#### ▪ Features

It is the command for reading multiple register type open signals from an external Modbus Slave unit.

#### ▪ Parameter

| Parameter Name | Data Type    | Default Value | Description   |
|----------------|--------------|---------------|---|
| iobus_list     | list(string) | -             | Modbus input name list (configured at TP)<br>Available only with the following signal types <ul style="list-style-type: none"> <li>• DR_MODBUS_REG_INPUT</li> <li>• DR_MODBUS_REG_OUTPUT</li> </ul> |

#### ▪ Return

| Value    | Description  |
|----------|--|
| res      | Number values read                                 |
| val_list | List of multiple signal values read simultaneously |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully     |

#### ▪ Example

```
# Modbus Register I/O "Holding1" is 1234, "Input1" is 567,
# and "Holding2" is 9876
res, val_list = get_modbus_inputs_list(iobus_list=[ "Holding1", "Input1", "Holding2"])
#res expected value = 3
#val_list expected value = [1234, 567, 9876]
```

### 8.4.9 wait\_modbus\_input(iobus, val, timeout=None)

#### ▪ Features

This function waits until the specified signal value of the Modbus digital I/O becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

#### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| iobus          | string    | -             | Modbus name (set in the TP)   |
| value          | int       | -             | Modbus digital I/O  |
|                |           |               | <ul style="list-style-type: none"> <li>• ON : 1</li> <li>• OFF : 0</li> </ul>     |
|                |           |               | Value for Modbus analog I/O   |
| timeout        | float     | -             | Waiting time (sec)<br>This function waits indefinitely if the timeout is not set. |

#### ▪ Return

| Value | Description       |
|-------|-------------------|
| 0     | Success           |
| -1    | Failed (time-out) |

#### ▪ Exception

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

#### ▪ Example

```
wait_modbus_input("CIN0", ON) # Indefinite wait until the "CIN0" signal becomes ON
wait_modbus_input("CIN0", OFF) # Indefinite wait until the "CIN0" signal becomes OFF
```

```
res = wait_modbus_input("CIN0", ON, 3) # Wait for up to 3 seconds until the "CIN0" signal becomes ON
```

```
# res = 0 if the "CIN0" signal becomes ON within 3 seconds.
```

```
# res = -1 if the "CIN0" signal does not become ON within 3 seconds.
```

### 8.4.10 set\_modbus\_slave(address, val)

- **Features**

It is used to export values to the General Purpose Register area of the Modbus TCP Slave.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description                         |
|----------------|-----------|---------------|-------------------------------------|
| address        | int       | -             | Address value of GPR area (128~255) |
| val            | int       | -             | 2byte value (0~65535)               |

- **Return**

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failure     |

- **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

- **Example**

```
set_modbus_slave(128, 0)
set_modbus_slave(255, 65535)
```

8.4.11 `get_modbus_slave(address)`

- **Features**

It is used to import values by approaching the General Purpose Register area of the Modbus TCP Slave.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description                                     |
|----------------|-----------|---------------|---|
| address        | int       | -             | Address value of the GPR area to read (128~255) |

- **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully     |

- **Example**

```
value1 = get_modbus_slave(128)
value2 = get_modbus_slave(255)
```

### 8.4.12 modbus\_crc16(data)

- **Features**

When using the Modbus protocol, this command reduces the load on Modbus CRC16 calculations.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description                        |
|----------------|-----------|---------------|------------------------------------|
| data           | byte      | -             | Modbus data for CRC16 calculations |

- **Return**

| Value   | Description                           |
|---------|---------------------------------------|
| crchigh | High byte of CRC16 calculation result |
| crclow  | Low byte of CRC16 calculation result  |

- **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

- **Example**

```
data = b"\x01\x02\x03\x04\x05\x06"
crchigh, crclow = modbus_crc16(data)
#crchigh = 186(DEC), BA(HEX)
#crclow = 221(DEC), DD(HEX)
```

## 8.4.13 modbus\_send\_make(send\_data)

- **Features**

When using the Modbus protocol, this command provides the result data including the Modbus CRC16 result for the send data.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description                             |
|----------------|-----------|---------------|---|
| send_data      | byte      | -             | Transfer data requiring CRC calculation |

- **Return**

| Value       | Description  |
|-------------|--|
| result_data | Data in which transmission data and CRC16 value are combined |

- **Exception**

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully     |

- **Example**

```
senddata = b"\x01\x02\x03\x04\x05\x06"
resultdata = modbus_send_make(senddata)
#resultdata = b'\x01\x02\x03\x04\x05\x06\xba\xdd'
```

### 8.4.14 modbus\_recv\_check(recv\_data)

- **Features**

When using Modbus protocol, this command to check data integrity using CRC16 value for receive data.

- **Parameter**

| Parameter Name | Data Type | Default Value | Description     |
|----------------|-----------|---------------|-----------------|
| recv_data      | byte      | -             | raw modbus data |

- **Return**

| Value | Description |
|-------|-------------|
| res   | True/False  |

- **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

- **Example**

```
#recvdata = b"\x01\x02\x03\x04\x05\x06\xba\xdd"
res = modbus_recv_check(recvdata)
#recv = True
```



## 8.5 Industrial Ethernet (EtherNet/IP,PROFINET)

### 8.5.1 set\_output\_register\_bit(address, val)

#### ▪ Features

It is used to export values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

#### ▪ Parameter

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Bit GPR area in Industrial Ethernet Slave(0-63) |
| val            | int            | -             | ON : 1<br>OFF : 0   |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failure     |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully     |

#### ▪ Example

```
set_output_register_bit (0, ON)
set_output_register_bit (63, OFF)
```

### 8.5.2 set\_output\_register\_int(address, val)

#### ▪ Features

It is used to export values to the Output Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

#### ▪ Parameter

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Int GPR area in Industrial Ethernet Slave(0-23) |
| val            | int            | -             | Int value(4bytes)   |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failure     |

#### ▪ Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

#### ▪ Example

```
set_output_register_int (0, 0x00FF00FF)
set_output_register_int (23, 65535)
```

### 8.5.3 set\_output\_register\_float(address, val)

#### ▪ Features

It is used to export values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

#### ▪ Parameter

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Float GPR area in Industrial Ethernet Slave(0-23) |
| val            | int            | -             | float value(4bytes)   |

#### ▪ Return

| Value          | Description |
|----------------|-------------|
| 0              | Success     |
| Negative value | Failure     |

#### ▪ Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

#### ▪ Example

```
set_output_register_float(0, 4.5)
set_output_register_float(23, 2.3)
```

### 8.5.4 `get_output_register_bit(address)`

▪ **Features**

It is used to import values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

▪ **Parameter**

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Bit GPR area in Industrial Ethernet Slave(0-63) |

▪ **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

▪ **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

▪ **Example**

```
a = get_output_register_bit(0)
b = get_output_register_bit(63)
```

### 8.5.5 `get_output_register_int(address)`

#### ▪ Features

It is used to import values to the Output Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

#### ▪ Parameter

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Int GPR area in Industrial Ethernet Slave(0-23) |

#### ▪ Return

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

#### ▪ Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

#### ▪ Example

```
a = get_output_register_int(0)
b = get_output_register_int(23)
```

### 8.5.6 `get_output_register_float(address)`

▪ **Features**

It is used to import values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

▪ **Parameter**

| Parameter Name | Data Type      | Default Value | Description   |
|----------------|----------------|---------------|---|
| address        | unsigned short | -             | Address value of Output Float GPR area in Industrial Ethernet Slave(0-23) |

▪ **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

▪ **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

▪ **Example**

```
a = get_output_register_float(0)
b = get_output_register_float(63)
```

### 8.5.7 `get_input_register_bit(address)`

- **Features**

It is used to import values to the Input Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

- **Parameter**

| Parameter Name | Data Type      | Default Value | Description  |
|----------------|----------------|---------------|--|
| address        | unsigned short | -             | Address value of Input Bit GPR area in Industrial Ethernet Slave(0-63) |

- **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

- **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

- **Example**

```
a = get_input_register_bit(0)
b = get_input_register_bit(63)
```

### 8.5.8 `get_input_register_int(address)`

▪ **Features**

It is used to import values to the Input Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

▪ **Parameter**

| Parameter Name | Data Type      | Default Value | Description  |
|----------------|----------------|---------------|--|
| address        | unsigned short | -             | Address value of Input Int GPR area in Industrial Ethernet Slave(0-23) |

▪ **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

▪ **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

▪ **Example**

```
a = get_input_register_int(0)
b = get_input_register_int(23)
```



### 8.5.9 `get_input_register_float(address)`

- **Features**

It is used to import values to the Input Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

- **Parameter**

| Parameter Name | Data Type      | Default Value | Description  |
|----------------|----------------|---------------|--|
| address        | unsigned short | -             | Address value of Input Float GPR area in Industrial Ethernet Slave(0-23) |

- **Return**

| Value | Description                  |
|-------|------------------------------|
| value | Corresponding register value |

- **Exception**

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

- **Example**

```
a = get_input_register_float(0)
b = get_input_register_float(63)
```

## 9. External Vision Commands

### Overview

Doosan Robotics provides the commands to guide robots with vision by connecting the robots to an external vision system. It enables connecting a robot to a 2D vision system which can measure the object position (Tx, Ty) data and the rotation (Rz) data (offset information) to guide the inputted robot task, and the commands can receive the measurement data inputs of multiple objects. The installation and tasks of the robot application using the 2D vision system need to calibrate the visual coordinate system of the vision system to the physical coordinate system of the robot system (coordinate system correction). When using an external vision system, the vision system must calibrate the coordinate system and transfer the corrected coordinate data to the robot.

The vision system can be installed in the Eye-in-hand mode connected to the robot or in-line mode separated from the robot. It must be fixed so that the relative position of the robot and vision system does not change during a task. The vision system and the robot controller communicate through the TCP/IP protocol, and communication is established when the cable of the vision system is connected to the wired hub port of the robot controller.

## 9.1 vs\_set\_info(type)

### ▪ Features

This function set the type of vision system to use.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| type           | int       | DR_VS_CUSTOM  | DR_VS_CUSTOM(0)<br>DR_VS_COGNEX(1)<br>DR_VS_SICK(2) |

### ▪ Return

| Value      | Description       |
|------------|-------------------|
| ID of Type | ID of type to set |

### ▪ Example

```
vs_set_info(DR_VS_COGNEX) #Vision type information setting
vs_connect("192.168.137.10") #Connect to vision - Vision IP, Default port
# Enter your task
vs_disconnect() #Disconnect to vision
```

### \* Supported Model

| Type         | Model  |
|--------------|--|
| DR_VS_COGNEX | COGNEX IS2000M-23M Series<br>COGNEX IS5600/5705 Series<br>COGNEX IS7000Series<br>COGNEX IS8000Series |
| DR_VS_SICK   | SICK Inspector PIM60 Series<br>SICK Inspector PI50 Series  |
| DR_VS_CUSTOM |  |

## 9.2 vs\_connect(ip\_addr, port\_num=9999)

- **Features**

This function establishes communication with the vision system.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                                      |
|----------------|-----------|---------------|--|
| ip_addr        | str       | -             | Server IP of vision module (ex. 192.168.137.200) |
| port_num       | int       | 9999          | Port 번호 (예, 9999)                                |

- **Return**

| Value | Description        |
|-------|--------------------|
| 0     | Connection success |
| -1    | Connection failed  |

- **Example**

```
vs_set_info(DR_VS_COGNEX) #Vision type information setting
vs_connect("192.168.137.10") #Connect to vision - Vision IP, Default port
# Enter your task
vs_disconnect() #Disconnect to vision
```

### 9.3 vs\_disconnect()

- **Features**

This function terminates the connection to the vision system.

- **Return**

| Value | Description |
|-------|-------------|
| N/A   | N/A         |

- **Example**

```
vs_set_info(DR_VS_COGNEX) #Vision type information setting
vs_connect("192.168.137.10") #Connect to vision - Vision IP, Default port
# Enter your task
vs_disconnect() #Disconnect to vision
```

## 9.4 vs\_get\_job()

### ▪ Features

This function loaded the task name, currently loaded in the vision system. (\*VS\_TYPE: DR\_VS\_COGNEX, DR\_VS\_SICK)

### ▪ Return

| Value    | Data Type | Description        |
|----------|-----------|--------------------|
| job_name | string    | Connection success |
| -1       | int       | Connection failed  |

### ▪ Example

```
vs_set_info(DR_VS_COGNEX) #Vision type information setting
vs_connect("192.168.137.10") #Connect to vision - Vision IP, Default port

vs_set_job("test.job") # Set (load) the current vision job
job_name=vs_get_job() # Get the current setting vision job
tp_popup("{0}".format(job_name))

vs_disconnect() # Disconnect to vision
```

## 9.5 vs\_set\_job(job\_name)

### ▪ Features

This function loaded the entered task into the vision system.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description            |
|----------------|-----------|---------------|------------------------|
| job_name       | string    |               | Task name to be loaded |

### ▪ Return

| Value | Data Type | Description |
|-------|-----------|-------------|
| 0     | int       | Success     |
| -1    | int       | Failed      |

### ▪ Example

```
vs_set_info(DR_VS_COGNEX) #Vision type information setting
vs_connect("192.168.137.10") #Connect to vision - Vision IP, Default port

vs_set_job("test.job") # Set (load) the current vision job
job_name=vs_get_job() # Get the current setting vision job
tp_popup("{0}".format(job_name))

vs_disconnect() # Disconnect to vision
```

## 9.6 vs\_trigger()

### ▪ Features

This function transmits the measurement command to the vision system. If the measurement is successful, the result is returned. (\*VS\_TYPE: DR\_VS\_COGNEX, DR\_VS\_SICK)

### ▪ Return

| Value    | Data Type   | Description   |
|----------|-------------|---|
| pos      | posx        | pos: measuring position of an object (posx parameter type)  |
| var_list | list[float] | var_list: Additional measurement results entered by Users<br>ex) Check pass/fail, distance measurement, angle measurement, etc. |
| -1, []   | int         | failed  |

### ▪ Example

```

vs_set_info(DR_VS_COGNEX)           # Select type of vision sensor
vs_connect("192.168.137.10")       # Vision IP, Default port

vis_posx = posx (410,310,300,0,0,0) # Define the initial posx data - vision
rob_posx = posx (400,300,300,0,180,0) # Define the initial posx data - robot

vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial posx data to Vision

for i in range(10):
    pos, var_list = vs_trigger()     # Execute the vision measurement
    if var_list[0] == 1:             # Check the inspection result
        robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
        move1(robot_posx_meas)      # move the robot pose
    else:
        tp_popup("Inspection Fail")

vs_disconnect()

```



## 9.7 vs\_set\_init\_pos(vision\_posx\_init, robot\_posx\_init, vs\_pos=1)

### ▪ Features

Enter the initial position information of the object to perform the vision guidance operation.  
(\*VS\_TYPE: DR\_VS\_COGNEX, DR\_VS\_SICK)

### ▪ Parameters

| Parameter Name   | Data Type | Default Value | Description                                 |
|------------------|-----------|---------------|---|
| vision_posx_init | posx      | -             | Vision measurement coordinate initial value |
| robot_posx_init  | posx      | -             | Coordinate initial value for robot work     |
| vs_pos           | int       | 1             | The pos number of the initial value entered |

### ▪ Return

| Value  | Data Type | Description                                 |
|--------|-----------|---|
| vs_pos | int       | Success – The entered pos number 입력된 pos 번호 |
| -1     | int       | Failed                                      |

### ▪ Example

```
vs_set_info(DR_VS_COGNEX)           # Select type of vision sensor
vs_connect("192.168.137.10")        # Vision IP, Default port
vis_posx = posx (410,310,300,0,0,0) # Define the initial posx data - vision
rob_posx = posx (400,300,300,0,180,0) # Define the initial posx data - robot
vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial posx data to Vision

for i in range(10):
    pos, var_list = vs_trigger()      # Execute the vision measurement
    if var_list[0] == 1:              # Check the inspection result
        robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
        move1(robot_posx_meas)        # move the robot pose
    else:
        tp_popup("Inspection Fail")

vs_disconnect()
```

## 9.8 vs\_get\_offset\_pos(vision\_posx \_meas, vs\_pos=1)

### ▪ Features

The coordinate of the robot is calculated using the coordinate values, measured in the vision system. The initial value should be entered in advance through vs\_set\_init\_pos.

### ▪ Parameters

| Parameter Name   | Data Type | Default Value | Description  |
|------------------|-----------|---------------|--|
| vision_posx_meas | posx      | -             | Vision measurement coordinate values, caculated using vs_trigger     |
| vs_pos           | int       | 1             | Pos number of the robot initial value to calculate offset coordinate |

### ▪ Return

| Value           | Data Type | Description |
|-----------------|-----------|-------------|
| robot_posx_meas | posx      | Success     |
| -1              | int       | Failed      |

### ▪ Example

```

vs_set_info(DR_VS_COGNEX)           # Select type of vision sensor
vs_connect("192.168.137.10")       # Vision IP, Default port

vis_posx = posx (410,310,300,0,0,0) # Define the initial posx data - vision
rob_posx = posx (400,300,300,0,180,0) # Define the initial posx data - robot

vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial posx data to Vision

for i in range(10):
    pos, var_list = vs_trigger()     # Execute the vision meausrement
    if var_list[0] == 1:             # Check the inspection result
        robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
        movel(robot_posx_meas)      # move the robot pose
    else:
        tp_popup("Inspection Fail")

vs_disconnect()

```

## 9.9 vs\_request(cmd)

### Features

This function sets the feature for the vision system to request  
(\*VS\_TYPE: DR\_VS\_CUSTOM)

### Parameters

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| cmd            | int       | -             | The number of objects to be detected by the vision system |

### Return

| Value | Description                      |
|-------|----------------------------------|
| 0     | Success                          |
| -1    | Failed                           |
| -2    | Communication timed out (3 sec.) |

### Example

```
vs_request(1)           # request the vision measurement on the "1" job
```

## 9.10 vs\_result()

### Features

This function retrieves the processing result of the vision system.

(\*VS\_TYPE: DR\_VS\_CUSTOM)

### Return

| Value        | Description   |
|--------------|---|
| cnt<br>(>=1) | Success<br>The number of objects detected by the vision system                              |
| result       | Position list as a result of the vision system (x coordinate, y coordinate, rotation value) |
| -            | cnt ==-2 and res = empty list if failed   |

### 예제

```

vs_set_info(DR_VS_CUSTOM)
res = vs_connect("192.168.137.200", 9999)           #Vision and communication
connection attempt
if res !=0:                                       #Check the result of communication
connection
    tp_popup("connection fail",DR_PM_MESSAGE) #End program if connection fails
    exit()

ret = vs_request(1)                               #Request vision measurement information of object 1

cnt, result = vs_result()                         #Load object measurement result information

for i in range(cnt):
    x = result[i][0]
    y = result[i][1]
    t = result[i][2]
    tp_popup("x={0},y={1}, t={2}".format(result[i][0], result[i][1],
result[i][2]),DR_PM_MESSAGE)

```

## 9.11 Integrated example 1 (DR\_VS\_COGNEX, DR\_VS\_SICK)

### Example

```

vs_set_info(DR_VS_COGNEX)                # Select type of vision sensor
if ( vs_connect("192.168.137.10") != 0 ): # Vision IP, Default port
    tp_popup("connection fail",DR_PM_MESSAGE)
    exit()

vis_posx_init = posx (410,310,300,0,0,0) # Define the initial posx data - vision
rob_posx_init1 = posx (400,300,300,0,180,0) # Define the initial posx data - robot
rob_posx_init2 = posx (420,320,300,0,180,0) # Define the initial posx data - robot

vs_set_init_pos(vis_posx_init, rob_posx_init1, VS_POS1) # Enter the initial posx data to Vision
vs_set_init_pos(vis_posx_init, rob_posx_init2, VS_POS2)

for i in range(10):
    pos_meas, var_list = vs_trigger()      # Execute the vision measurement
    if pos_meas== -1:                     # Vision Fail to measure the object
        tp_popup("Vision measure fail")
        continue
    if var_list[0] == 1:                   # Check the inspection result
        # Get guided posx data
        rob_posx1_meas = vs_get_offset_pos(pos_meas, VS_POS1) # offset the robot pose
        rob_posx2_meas = vs_get_offset_pos(pos_meas, VS_POS2) # offset the robot pose
        movel(rob_posx1_meas)
        movel(rob_posx2_meas)
    else:
        tp_popup("Inspection Fail")
        continue

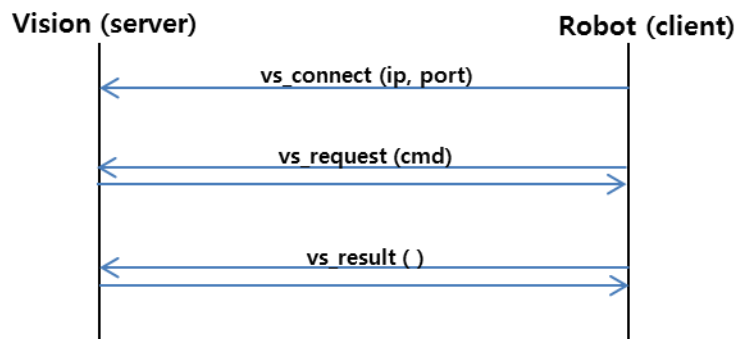
vs_disconnect()

```

## 9.12 Integrated Example 2

### Communication Protocol

The vision system must conform to the following protocol to ensure that vision commands run properly.



#### vs\_request (cmd)

1) Robot controller → Vision system

**"MEAS\_START" +cmd[4byte]**

- cmd refers to the number of detected objects: Conversion of the integer to 4 bytes. ex) cmd=1 → 00000001

ex) In case of **cmd= 1** : "MEAS\_START"+**00000001**

Actual packet : 4D4541535F5354415254**00000001**

2) Vision system → Robot controller

**"MEAS\_OK"** is transmitted if the vision system is normal, and **"MEAS\_NG"** is transmitted otherwise.

#### VS\_result()

1) Robot controller → Vision system

**"MEAS\_REQUEST"**

2) Vision system → Robot controller

**"MEAS\_INFO" +cnt[4byte] +[(x[4byte] + y[4byte] + t[4byte]) x cnt]**

- cnt refers to the number of detected objects.

- The transmitted x (x coordinate), y (y coordinate), and t (rotation value) must be scaled up 100 times.

ex) **cnt = 1** , (x=1.1 , **y=2.2**, t=3.3)

**"MEAS\_INFO"+1[4byte] +110[4byte] +220[4byte] +330[4byte]**

Actual packet: 4D4541535F494E464F**00000001**00000006E**000000DC**0000014A

ex) **cnt = 2** , (x=1.1 , **y=2.2**, t=3.3) (**x=1.1** , y=-2.2, **t=-3.3**)

**"MEAS\_INFO"+2[4byte] +110[4byte] +220[4byte] +330[4byte]**

**+110[4byte] -220[4byte] -330[4byte]**

Actual packet: 4D4541535F494E464F**00000002**00000006E**000000DC**0000014A

**0000006EFFFFFF24FFFFFFEB6**

**Example**

```
vs_set_info(DR_VS_CUSTOM)
res = vs_connect("192.168.137.200", 9999)    #Vision and communication connection
attempt
if res !=0:                                #Check the result of communication
connection
    tp_popup("connection fail",DR_PM_MESSAGE) #Upon connection failure, program
termination
    exit()

ret = vs_request(1)                        #Request of Vision Measurement
Information for No. 1 Object

cnt, result = vs_result()                  #Get object measurement result information

for i in range(cnt):
    x = result[i][0]
    y = result[i][1]
    t = result[i][2]
    tp_popup("x={0},y={1}, t={2}".format(result[i][0], result[i][1],
result[i][2]),DR_PM_MESSAGE)
```

## 10. Doosan Vision(SVM) Command

### 10.1 svm\_connect (ip= "192.168.137.5", port=20)

- **Features**

This function establishes communication with the SVM.

- **Parameters**

| Parameter Name | Data Type | Default Value   | Description                        |
|----------------|-----------|-----------------|------------------------------------|
| ip             | str       | "192.168.137.5" | Server IP address of vision module |
| port           | int       | 20              | Port number                        |

- **Return**

| Value | Description        |
|-------|--------------------|
| 0     | Connection success |
| -1    | Connection failed  |

- **Example**

```
svm_connect()      #Connect to vision - Default IP address and port number
# Enter the vision task
svm_disconnect()   #Disconnect to vision
```



## 10.2 svm\_disconnect ()

- **Features**

This function terminates the connection to the SVM.

- **Return**

| Value | Description |
|-------|-------------|
| N/A   | N/A         |

- **Example**

```
svm_connect() #Connect to vision – Default IP address and port
# Enter the vision task
svm_disconnect() #Disconnect to vision
```

## 10.3 svm\_set\_job (job\_id)

- **Features**

This function loads the Vision task corresponding to the input id into the SVM..

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                          |
|----------------|-----------|---------------|--------------------------------------|
| job_id         | int       | -             | Vision Task id (ex. 1000, 2000, ...) |

- **Return**

| Value | Description         |
|-------|---------------------|
| 0     | Job Loading success |
| -1    | Job Loading fail    |

- **Example**

```
svm_connect()           #Connect to vision - Vision IP, Default port
vision_test=1000       # Define vision job ID
svm_set_job(vision_test) # Load the vision_test (1000)
svm_disconnect()       #Disconnect to vision
```

## 10.4 svm\_get\_robot\_pose (job\_id)

### ▪ Features

The robot pose information(joint coordinate system) set in the vision task is loaded. Robot pose information is used as shoot\_pose for vision task.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                          |
|----------------|-----------|---------------|--------------------------------------|
| job_id         | int       | -             | Vision Task id (ex. 1000, 2000, ...) |

### ▪ Return

| Value     | Description                                    |
|-----------|--|
| float [6] | Robot joint coordinate information (posj type) |
| -1        | failed   |

### ▪ Example

```

svm_connect()                # Connect to vision
vision_test=1000             # Define vision job ID
svm_set_job(vision_test)     # Load the vision_test (1000)
shoot_pos=svm_get_robot_pose (vision_test) # Load the robot pose of vision_test
tp_popup("{0}".format(shoot_pos))
svm_disconnect()             # Disconnect to vision

```

## 10.5 svm\_get\_vision\_info (job\_id)

### ▪ Features

Performs the measurement command corresponding to the input vision task. The detailed information of the measurement command of the vision work should be entered in advance through the Workcell manager(WCM).

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                          |
|----------------|-----------|---------------|--------------------------------------|
| job_id         | int       | -             | Vision Task id (ex. 1000, 2000, ...) |

### ▪ Return

| Value | Data Type   |
|-------|---|
| 1     | Measurement success – One object was detected / measured successfully.      |
| 0     | Measurement failed – Failed to detect the corresponding vision work object. |
| -1    | Measurement failed – Communication error (timeout)                          |

### ▪ Example

```

svm_connect()                # Connect to vision
vision_test=1000             # Define vision job ID
svm_set_job(vision_test)     # Load the vision_test (1000)
shoot_pos=svm_get_robot_pose(vision_test) # Load the robot pose of vision_test
count=svm_get_vision_info(vision_test)   # Execute the vision measurement
tp_popup("{0}".format(count)) # Check the result
svm_disconnect()             # Disconnect to vision
    
```

## 10.6 svm\_get\_variable (tool\_id, var\_type)

### ▪ Features

If the object detection/measurement is successful(1) by executing `svm_get_vision_info`, the detection/measurement data is loaded. Enter the tool id and variable type for the data to be loaded.

- Position tool: POSX\_TYPE (Object location), VALUE\_TYPE (Detection similarity)
- Presence tool: INSP\_TYPE (Presence inspection result), VALUE\_TYPE (Pixel count)
- Distance tool: INSP\_TYPE (Distance inspection result), VALUE\_TYPE (Distance measure)
- Angle tool: INSP\_TYPE (Angle inspection result), VALUE\_TYPE (Angle measure)
- Diameter tool: INSP\_TYPE (Diameter inspection result), VALUE\_TYPE (Diameter measure), POSX\_TYPE (Circle center position)

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| tool_id        | int       | -             | Vision tool id (ex. 1000, 1001, 1002, ...)   |
| var_type       | int       | -             | POSX_TYPE: Vision measurement coordinate variable (posx)<br>INSP_TYPE: Inspection result variable (int)<br>VALEU_TYPE: Measurement result (int or float) |

### ▪ Return

| Value    | Description   |
|----------|---|
| variable | POSX_TYPE – Coordinate information variable, ex.<br>Posx(x,y,z,rx,ry,rz)<br>INSP_TYPE: Inspection result variable - int (Returns 1 if successful)<br>VALEU_TYPE: Measurement result variable (int of float) |
| -1       | Failed – No measurement data or input variable error.   |

- **Example**

```
svm_connect()                # Connect to vision
vision_test=1000             # Define vision job ID
print_insp=1001             # Define inspection tool ID
box_size=1002               # Define measurement tool ID
count=svm_get_vision_info(vision_test) # Execute the vision measurement
if (count==1):              # Check the result
    # Get the position information (posx) of vision_test tool
    pos_result=svm_get_variable(vision_test, POSX_TYPE)
    tp_popup("{0}".format(pos_result))

    # Get the inspection information (PASS or Fail) of print_insp tool
    inspection_result=svm_get_variable(print_insp, INSP_TYPE)
    tp_popup("{0}".format(inspection_result))

    # Get the distance information (distance) of box_size tool
    measurement_result= svm_get_variable(box_size, VALUE_TYPE)
    tp_popup("{0}".format(measurement_result))

svm_disconnect()            # Disconnect to vision
```

## 10.7 svm\_get\_offset\_pos (posx\_robot\_init, job\_id, tool\_id)

### ▪ Features

The robot task coordinate information reflecting the vision measurement result is loaded into the robot work coordinate input by the user.

\*Procedure: Input posx\_robot\_init → Vision measurement → Call svm\_get\_offset\_pos  
→ Changed robot work coordinates (posx\_robot\_offset) output

### ▪ Parameters

| Parameter Name  | Data Type | Default Value | Description  |
|-----------------|-----------|---------------|--|
| posx_robot_init | posx      | -             | Robot task coordinate information)<br>(Input by direct teaching method.) |
| job_id          | Int       | -             | Vision job id (ex. 1000, 2000, 3000, ...)                                |
| tool_id         | int       | -             | Vision tool id (ex. 1000, 1001, 1002, ...)                               |

### ▪ Return

| Value | Description  |
|-------|--|
| posx  | Robot work coordinate information reflecting vision measurement result |
| -1    | Failed – No measurement data or input variable error.                  |

### ▪ Example

```

svm_connect()                # Connect to vision
vision_test=1000             # Define vision job ID
count=svm_get_vision_info(vision_test) # Execute the vision measurement
if (count == 1):             # Check the result
    # Get the position information (posx) of vision_test tool
    pos_result=svm_get_variable(vision_test, POSX_TYPE)
    tp_popup("{0}".format(pos_result))
    # Get the vision guided robot pose
    Id_list =[vision_test]
    pos_list =[pos_result]
    svm_set_init_pos_data(Id_list,pos_list)

    rob_posx=svm_get_offset_pos(posx(200,200,100,0,180,0), vision_test)
    tp_popup("{0}".format(rob_posx))
    # move to the rob_posx
    move!(rob_posx, vel=30, acc=100)
svm_disconnect()             # Disconnect to vision

```

## 10.8 svm\_set\_init\_pos\_data(Id\_list, Pos\_list)

### ▪ Features

Enter the initial id\_list and posx\_list information of the object to perform the vision guidance operation.

### Caution

\* Be sure to set the settings before calling the function svm\_get\_offset\_pos (posx\_robot\_init, job\_id, tool\_id).

\* Note : id\_list and pos\_list should be matched with posx corresponding to each id.

### ▪ Parameters

| Parameter Name | Data Type  | Default Value | Description                            |
|----------------|------------|---------------|--|
| Id_list        | List(int)  | -             | Id list ([id, id, id, ...])            |
| Pos_list       | List(Posx) | -             | Posx list (ex.[posx, posx, posx, ...]) |

### ▪ Return

| Value | Description |
|-------|-------------|
| None  | -           |

### ▪ Example

```
svm_connect()                # Connect to vision
vision_test=1000             # Define vision job ID
count=svm_get_vision_info(vision_test) # Execute the vision measurement
if (count == 1):            # Check the result
    # Get the position information (posx) of vision_test tool
    pos_result=svm_get_variable(vision_test, POSX_TYPE)
    tp_popup("{0}".format(pos_result))
    # Get the vision guided robot pose
    Id_list =[vision_test]
    pos_list =[pos_result]
    svm_set_init_pos_data(Id_list,pos_list)
    rob_posx=svm_get_offset_pos(posx(200,200,100,0,180,0), vision_test)
    tp_popup("{0}".format(rob_posx))
    # move to the rob_posx
    movel(rob_posx, vel=30, acc=100)
svm_disconnect()            # Disconnect to vision
```



## 10.9 svm\_set\_tp\_popup (svm\_flag)

### ▪ Features

Set whether (tp\_popup) should be displayed when SVM error occurs.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description                    |
|----------------|-----------|---------------|--------------------------------|
| svm_flag       | int       | 1             | 1(Activation), 0(Deactivation) |

### ▪ Return

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| None           | -         | -             | -           |

### ▪ Example

```
svm_set_tp_popup(0)           # Hide tp_popup
svm_connect()                 # Connect to vision
svm_disconnect()              # Disconnect to vision
```

### 10.10 svm\_set\_led\_brightness(value)

- **Features**

Set SVM LED brightness value.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                    |
|----------------|-----------|---------------|--------------------------------|
| value          | int       | -             | LED brightness value (0-1000). |

- **Return**

| Value | Data Type | Description   |
|-------|-----------|---|
| -1    | int       | Fail - No measurement data or input variable error. |

#### Example

```
svm_connect()                # Connect to vision
svm_set_led_brightness(500)
svm_disconnect()             # Disconnect to vision
```

## 10.11 svm\_get\_led\_brightness()

- **Features**

Return the LED brightness value set in the SVM.

- **Return**

| Value | Description   |
|-------|---|
| int   | SVM brightness value (0-1000)                       |
| -1    | Fail - No measurement data or input variable error. |

### Example

```
svm_connect()           # Connect to vision
svm_get_led_brightness()
svm_disconnect()        # Disconnect to vision
```

## 10.12 svm\_set\_camera\_exp\_val(value)

- **Features**

Set exposure value of the SVM.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                                 |
|----------------|-----------|---------------|---|
| value          | int       | -             | SVM exposure value (2,660,000 – 29,260,000) |

- **Return**

| Value | Description   |
|-------|---|
| -1    | Fail - No measurement data or input variable error. |

### Example

```
svm_connect()           # Connect to vision
svm_set_camera_exp_val(2,660,000)
svm_disconnect()        # Disconnect to vision
```

## 10.13 svm\_set\_camera\_gain\_val(value)

- **Features**

Set SVM gain value..

- **Parameters**

| Parameter Name | Data Type | Default Value | Description              |
|----------------|-----------|---------------|--------------------------|
| value          | int       | 1             | SVM gain value (0-1600). |

- **Return**

| Value | Description   |
|-------|---|
| -1    | Fail - No measurement data or input variable error. |

### Example

```
svm_connect()           # Connect to vision
svm_set_camera_gain_val(500)
svm_disconnect()       # Disconnect to vision
```

## 10.14 svm\_set\_camera\_load(job\_id)

- **Features**

Load LED brightness, exp, gain and focus setting saved in the Job numbered job\_id.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description                    |
|----------------|-----------|---------------|--------------------------------|
| job_id         | int       |               | job id (ex - 1000, 2000, 3000) |

- **Return**

| Value | Description   |
|-------|---|
| -1    | Fail - No measurement data or input variable error. |

### Example

```
svm_connect()                # Connect to vision
svm_set_camera_load(job_id)
svm_disconnect()             # Disconnect to vision
```

## 10.15 Intergrated example (SVM)

### ▪ Example

Vision Job setting status

- After deleting all the jobs saved in WCM, create vision task / tool as below.
- Create vision task : vision\_test (position tool, 1000)
- Add vision tools: print\_insp (presence tool, 1001)  
                  box\_size (distance tool, 1002)
- Select the "vision\_test" task in TW vision command set the variable information.
- Add the following example to your custom code to test.

```

svm_connect()                # Connect to vision
vision_test=1000            # Define vision job ID
print_insp=1001            # Define inspection tool ID
box_size=1002              # Define measurement tool ID
svm_set_job(vision_test)    # Load the vision_test (1000)
movej(svm_get_robot_pose(vision_test), vel=10, acc=20) # Move to shoot pose (movej)

if (svm_get_vision_info(vision_test)== 1): # Execute the vision measurement
  # Load the vision variables
  # Get the position information (posx) of vision_test tool
  pos_result=svm_get_variable(vision_test, POSX_TYPE)
  tp_popup("pos_result {0}".format(pos_result))

  # Get the inspection information (PASS or Fail) of print_insp tool
  inspection_result=svm_get_variable(print_insp, INSP_TYPE)
  tp_popup("inspection_result {0}".format(inspection_result))

  # Get the distance information (distance) of box_size tool
  measurement_result= svm_get_variable(box_size, VALUE_TYPE)
  tp_popup("measurement_result {0}".format(measurement_result))

  # Move to the vision guided robot pose
  # Get the vision guided robot pose
  ld_list =[vision_test]
  pos_list =[pos_result]
  svm_set_init_pos_data(ld_list,pos_list)
  rob_posx=svm_get_offset_pos(posx(200,200,100,0,180,0), vision_test)
  tp_popup("rob_posx {0}".format(rob_posx))

  # move to the rob_posx
  movel(rob_posx, vel=30, acc=100)

svm_disconnect()            # Disconnect to vision

```

# 11. Application Commands

## 11.1 External Encoder Setting Commands

### 11.1.1 `set_extenc_polarity(channel, polarity_A, polarity_B, polarity_Z, polarity_S)`

#### Features

It configures the polarity of phase A, B and the trigger method of phase S, Z of the corresponding encoder channel.

#### Parameter

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| channel        | int       | 1             | Encoder channel (1, 2)<br>1: Channel 1<br>2: Channel 2      |
| polarity_A     | int       | 0             | Polarity of phase A (0: Phase A, 1: /Phase A)               |
| polarity_B     | int       | 0             | Polarity of phase B (0: Phase B, 1: /Phase B)               |
| polarity_Z     | int       | 0             | Trigger method of Phase Z (0: Falling edge, 1: Rising edge) |
| polarity_S     | int       | 0             | Trigger method of Phase S (0: Falling edge, 1: Rising edge) |

#### Return

| Value | Description |
|-------|-------------|
| N/A   | Not Used    |

#### Exception

| Exception | Description |
|-----------|-------------|
| N/A       | Not Used    |

#### Example

```
set_extenc_polarity(1, 0, 1, 0, 1)
# External Encoder channel 1 is set to phase A, /phase B, phase Z (falling edge),
phase S (rising edge)
```



**Related Commands****set\_extenc\_mode**

11.1.2 **set\_extenc\_mode(channel, mode\_AB, pulse\_AZ, mode\_Z, mode\_S, inverse\_cnt)**

**Features**

It configures the operation mode of phase A, B, Z and S of the corresponding encoder channel.

**Parameter**

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---|
| channel        | int       | 1             | Encoder channel (1, 2)<br>1: Channel 1<br>2: Channel 2  |
| mode_AB        | int       | 0             | Use of phase AB Mode (0 ~ 4)<br>0: Not Used<br>1: Phase A Quadrature use<br>Phase B Quadrature use<br>2: Phase A Count<br>Phase B Direction use<br>3: Phase A Up Count use<br>Phase B Not Used<br>4: Phase A Down Count use<br>Phase B Not Used |
| pulse_AZ       | int       | 0             | Pulse A Count per Pulse Z<br>(0 ~ 100000)   |
| mode_Z         | int       | 0             | Phase Z Use Mode (0 ~ 1)<br>0: Not Used<br>1: A/B Count Error Compensation<br>2: Encoder Count Clear  |
| mode_S         | int       | 0             | Phase S Use Mode (0 ~ 1)<br>0: Not Used<br>1: Strobe Signal<br>2: Encoder Count Clear   |
| inverse_cnt    | int       | 0             | Encoder Count Direction Reverse   |

## External Encoder Setting Commands

---

| Parameter Name | Data Type | Default Value | Description                        |
|----------------|-----------|---------------|------------------------------------|
|                |           |               | Status<br>0: Forward<br>1: Reverse |

### Return

| Value | Description |
|-------|-------------|
| N/A   | Not Used    |

### Exception

| Exception | Description |
|-----------|-------------|
| N/A       | Not Used    |

### Example

```
set_extenc_mode(1, 2, 20000, 1, 1, 0)
# External Encoder channel 1 operation mode is set as follows
# Phase A Count, Phase B Direction Use
# Pulse A Count per Z Pulse is 20000
# Phase Z error count accumulate compensation mode use, phase S use
# Encoder Count direction is set to forward
```

### Related Commands

`set_extenc_polarity`

### 11.1.3 `get_extenc_count(channel)`

#### Features

Get the count value of the corresponding encoder channel.

#### Parameter

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| channel        | int       | 1             | Encoder channel (1, 2)<br>1: Channel 1<br>2: Channel 2 |

#### Return

| Value | Description  |
|-------|--|
| count | Current encoder count value of corresponding channel |

#### Exception

| Exception | Description |
|-----------|-------------|
| N/A       | Not Used    |

#### Example

```
enc_cnt = get_extenc_count(1)
# External Encoder channel 1 current count value calculation
```

#### Related Commands

`set_extenc_polarity`

`set_extenc_mode`

`clear_extenc_count`

### 11.1.4 clear\_extenc\_count(channel)

#### Features

Reset counter value of the corresponding encoder channel to 0.

#### Parameter

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| channel        | int       | 1             | Encoder channel (1, 2)<br>1: Channel 1<br>2: Channel 2 |

#### Return

| Value | Description |
|-------|-------------|
| N/A   | Not Used    |

#### Exception

| Exception | Description |
|-----------|-------------|
| N/A       | Not Used    |

#### Example

```
clear_extenc_count(1)  
# External Encoder channel 1 count value reset to 0
```

#### Related Commands

`get_extenc_count`

## 11.2 Conveyor Tracking

### 11.2.1 set\_conveyor(name)

#### ▪ Features

If conveyor information is configured in the UI, obtain ID with the conveyor name to start the Conveyor Tracking Application from the program and execute the command for workpiece monitoring. Workpiece monitoring is performed on workpieces triggered in the conveyor, and monitoring continues until the program ends.

#### ▪ Parameter

| Parameter Name | Data Type | Default Value | Description   |
|----------------|-----------|---------------|---------------|
| name           | string    | -             | Conveyor name |

#### ▪ Return

| Value       | Description   |
|-------------|---|
| Conveyor ID | Returns Conveyor ID if conveyor setting is successful |
| None        | Conveyor setting failure                              |

#### ▪ Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

#### ▪ Example

```
CONV1 = set_conveyor("conveyor1")
```

#### ▪ Related Commands

**get\_conveyor\_obj(), tracking\_conveyor(), untracking\_conveyor()**

### 11.2.2 set\_conveyor\_ex

#### ▪ Features

Configures the conveyor and obtains Conveyor ID to allow the Conveyor Tracking Application to start. After the command is executed, it monitors workpieces triggered in the configured conveyor until the program ends. It can be used when you need to set parameters manually if is unavailable to configure conveyor information through UI.

- 1) Added default value for all arguments compared to versions prior to M2.4.0
- 2) Added 'ref' argument compared to versions prior to M2.4.0 (world coordinates available)
- 3) Removed 'obj\_offset\_coord' argument compared to versions prior to M2.4.0, The 'obj\_offset\_coord' argument is changed to input only in get\_conveyor\_obj() function.

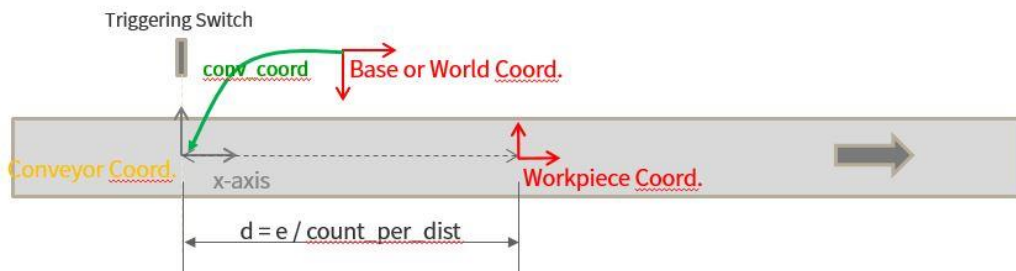
#### ▪ Parameter

| Parameter Name       | Data Type       | Default Value     | Description   |
|----------------------|-----------------|-------------------|---|
| name                 | string          | ""                | Conveyor name   |
| conv_type            | int             | 0                 | Conveyor type (0: Linear, 1: Circular)  |
| encoder_channel      | int             | 1                 | External encoder channel (1, 2)   |
| triggering_mute_time | float           | 0.0               | It is the time (s) triggering (encoder reset, start workpiece tracking) is not performed when a triggering signal is received immediately after triggering. |
| count_per_dist       | int             | 5000              | Encoder count converted value per length (Linear: count/m, Circular: count/rad)   |
| conv_coord           | posx            | posx(0,0,0,0,0,0) | Fixed conveyor coordinates (based on Base/World coordinates, mm, °)   |
|                      | list (float[6]) |                   |   |
| ref                  | int             | DR_BASE           | Reference coordinates of conveyor coordinates (DR_BASE: Base, DR_WORLD: World)  |
| conv_speed           | float           | 100.0             | Conveyor nominal velocity (Linear: mm/s, Circular: °/s)   |
| speed_filter_size    | int             | 500               | Moving Average Filter Size during conveyor velocity filtering   |
| min_dist             | float           | 0.0               | Minimum conveyor work length (based on Triggering Switch, Linear: mm, Circular: °)  |

| Parameter Name    | Data Type | Default Value | Description   |
|-------------------|-----------|---------------|---|
| max_dist          | float     | 1000.0        | Maximum conveyor work length (based on Triggering Switch, Linear: mm, Circular: °)                      |
| watch_window      | float     | 100.0         | Conveyor work standby monitoring length (based on minimum work length, Linear: mm, Circular: °)         |
| out_tracking_dist | float     | 10.0          | Conveyor tracking release buffer section length (based on maximum work length, Linear: mm, Circular: °) |

 **Note**

- Currently conv\_type argument does not support Circular Conveyors!
- All workpieces that pass the Triggering Switch are monitored until they reach max\_dist after set\_conveyor() or set\_conveyor\_ex() function execution and before the program ends.
- However, if triggering\_mute\_time is configured, and if the Triggering Switch activates during the corresponding time after the previous workpiece is detected, it is not included on the monitoring list. It is used when noise is present in the Triggering Switch or when the workpiece needs to be removed for a certain amount of time.
- conv\_coord is a coordinate system fixed to the conveyor relative to the base or world coordinate system. **Here, the x-axis of conv\_coord represents the direction the conveyor flows.** From the moment the conveyor workpiece activates the triggering switch, the increased encoder value can be converted to the length of the workpiece travel by using the count\_per\_dist argument, and extending this length in the x-axis direction of the conv\_coord will position the workpiece relative to the reference coordinate system.



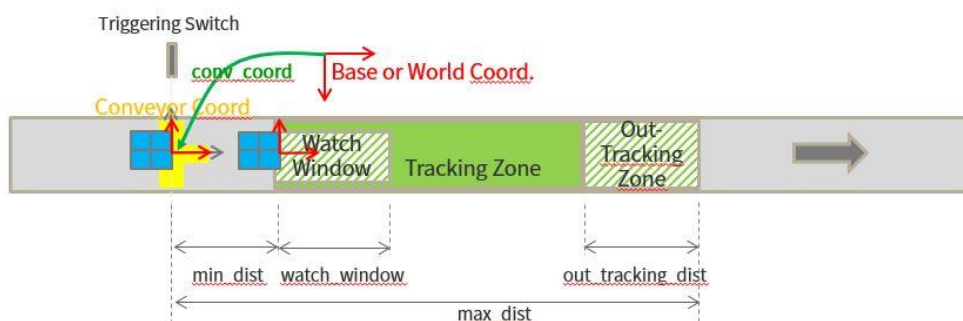
d: distance from conv\_coord to workpiece  
 e: incremental encoder value after triggering switch activated

[Conveyor/Item Coordinate]

- conv\_speed is the moving speed of the conveyor. It is used to give Info only if the conveyor speed sensed by the encoder exceeds 200% of this speed. Therefore, if the measurement is not possible through the TP UI, enter the approximate value.
- Speed\_filter\_size is the size of the moving-average filter used to estimate the conveyor speed from the encoder. The larger the size, the more the noise can be canceled, but the tracking accuracy may deteriorate during acceleration and deceleration.

## Conveyor Tracking

- The area on top of the conveyor is categorized into Watch Window, Tracking Zone and Out-Tracking Zone.
- Watch Window is the area that determines whether workpieces within the area are available for the job when obtaining workpiece coordinates for tracking. When the `get_conveyor_obj()` function is loaded, if a workpiece is not present within this area, the function is not returned, and if a workpiece is present within this area, it returns workpiece coordinates according to `get_conveyor_obj()` function options (FIFO, LIFO).
- The Tracking Zone is the area that performs Conveyor Tracking.
- The Out-Tracking Zone is the area where the robot automatically ends tracking after it determines that the robot has exited the work space of the robot or the work space specified by the user during continuous tracking.
- These three areas are defined with the four lengths (`min_dist`, `max_dist`, `watch_window`, `out_tracking_dist`) as shown below.



[Conveyor Area and Length]

### Return

| Value       | Description   |
|-------------|---|
| Conveyor ID | Returns Conveyor ID if conveyor setting is successful |
| None        | Conveyor setting failure                              |

### Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |



- **Example**

```
CONV1 = set_conveyor_ex(name='conveyor_1',
    conv_type=0, # linear
    encoder_channel=1, triggering_mute_time=0.0,
    count_per_dist=5000, # 5000 count/mm)
    conv_coord=posx(500, 100, 500, 0, -90, 0), ref=DR_BASE,
    conv_speed=100.0, # conveyor speed: 100 mm/s,
    speed_filter_size=500, # moving avg. filter size: 500 ms
    min_dist=100, max_dist=1000, watch_window=200, out_tracking_dist=10)
```

- **Related Commands**

**get\_conveyor\_obj(), tracking\_conveyor(), untracking\_conveyor()**

## 11.2.3 `get_conveyor_obj(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None)`

### ▪ Features

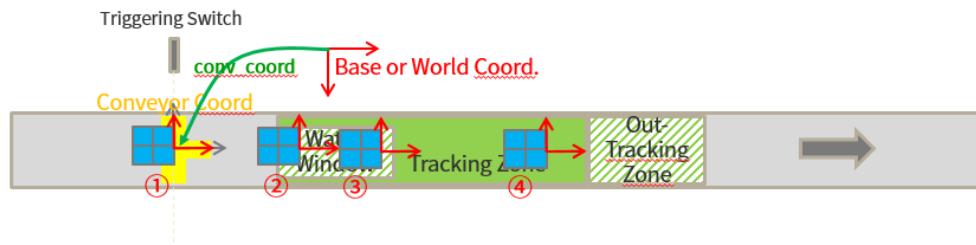
It returns the workpiece coordinate ID available for the job from the corresponding conveyor. When a function is called, it returns the workpiece present in the Watch Zone one by one according to the container rule.

### ▪ Parameter

| Parameter Name   | Data Type       | Default Value | Description  |
|------------------|-----------------|---------------|--|
| conv_id          | int             | -             | Conveyor ID  |
| timeout          | float           | None          | If no workpiece to return is present, it ends the standby and returns the function |
| container_type   | int             | DR_FIFO       | Workpiece container type (DR_FIFO: first-in/first-out, DR_LIFO: last-in/last-out)  |
| obj_offset_coord | posx            | None          | Workpiece coordinates (mm, °) based on conveyor lock coordinates                   |
|                  | list (float[6]) |               |  |

### Note

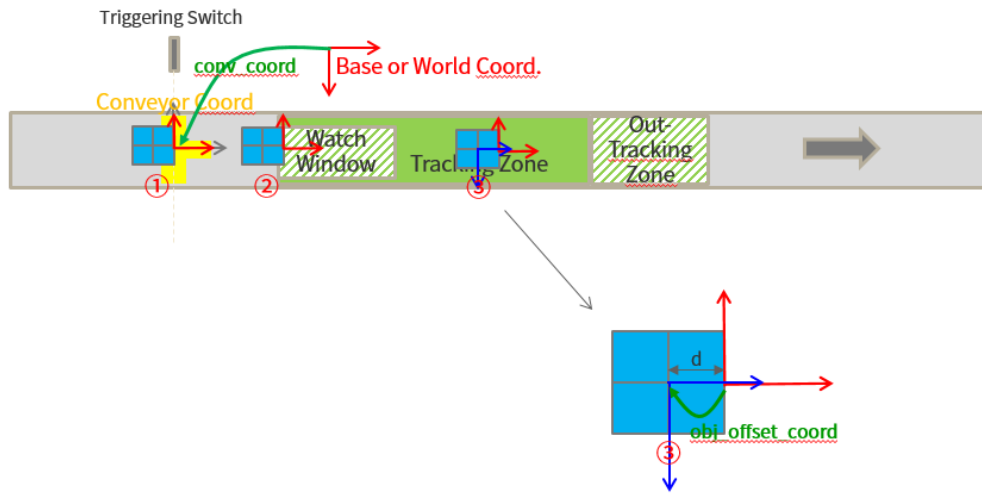
- When calling this function, it returns the coordinates ID of each workpiece in the Watch Window according to the container rule. For example, if you call `get_conveyor_obj()` function when the workpieces are placed as shown below, the workpiece ② and ③ in the Watch Window will be candidates. At this time, if the `container_type` is set to `DR_FIFO` the coordinates ID of ③ that entered the Watch Window first. If it is set to `DR_LIFO`, it returns the coordinates ID of ② that entered the Watch Window later. If there is no workpieces in the Watch Window at the time of the function call, it waits until the time set in the `timeout` parameter and return the id if the workpiece comes in.



[Workpiece Coordinate ID Return Rule Description]

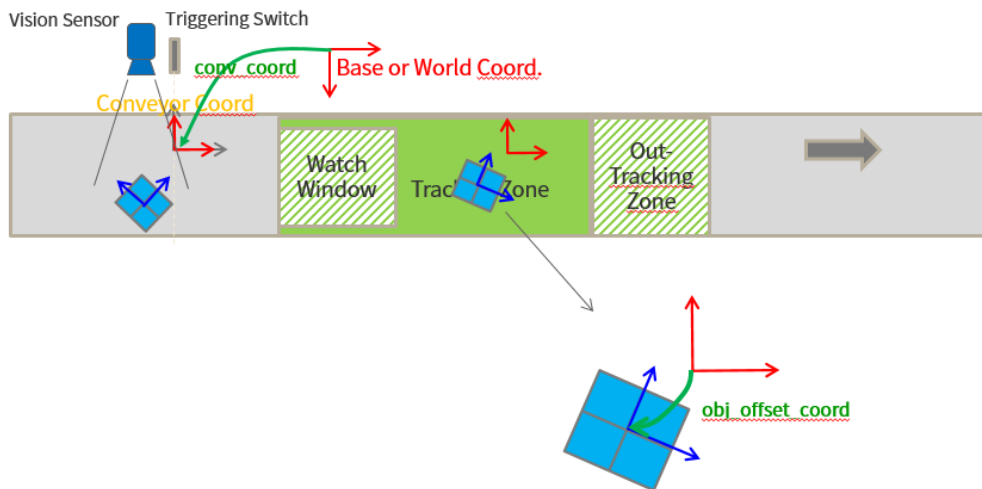
- `obj_offset_coord` is used when you want to apply offset to the workpiece coordinates. It is usually used for easy input of a teaching point or when you want to dynamically change the position and orientation of the workpiece coordinates in conjunction with an external sensor (ex. Vision sensor).

In the case shown below, the workpiece coordinates are created on the right side of the workpiece and the orientation is different from the base or world coord. At this time, if you want to position the workpiece coordinates at the center of workpiece and make the orientation to be same with the ones of base or world coordinates, you can apply it as  $\text{obj\_offset\_coord} = \text{posx}(-d, 0, 0, -90, 0, 0)$ . It is not necessary to acquire a teaching point through this TP UI, but it could utilize this method if you need to use  $\text{drl}$  only or enter the teaching point directly.



[obj\_offset\_coord use case 1]

Next, if the workpiece changes its position in a direction that is independent of the conveying direction, or the orientation of the workpiece changes as shown below, the encoder signal alone cannot determine the position / orientation of the workpiece. In this case, you need to detect them using external vision sensor. After detecting this value, you can input the position/orientation change detected as  $\text{obj\_offset\_coord}$  dynamically and the workpiece coordinates are created accordingly.



[obj\_offset\_coord use case 2: using vision sensor]

### Return

| Value            | Description   |
|------------------|---|
| int              | CONV_COORD. Conveyor user coordinate ID (121~150)         |
| Negative integer | If no workpiece is present even after the timeout expires |

### Note

If no workpiece to return is present, no function is returned until the timeout time expires. If the timeout time expires but no workpiece is present, it returns -1. However, if a timeout time is not entered, it doesn't return continuously.

### Exception

| Exception                   | Description                            |
|-----------------------------|--|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred occurred |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred       |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred      |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully          |

### Example

```
## One object in a cycle
CONV1 = set_conveyor('conveyor1')

movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE) # waiting position
while True:
    CONV_COORD_1 = get_conveyor_obj(CONV1)
    tracking_conveyor(CONV1)

    # synched motion
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
    movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
    set_digital_output(DO_GRIPPER, 1)
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

    untracking_conveyor(CONV1)

    movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE) # waiting position

## Multi objects in a cycle
CONV1 = set_conveyor('conveyor1')

while True:
    CONV_COORD_1 = get_conveyor_obj(CONV1)
```

```
tracking_conveyor(CONV1)

# fist object
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

# second object
CONV_COORD_2 = get_conveyor_obj(CONV1, time_out=10)
if CONV_COORD_2 > 0: # -1 if no objects available during time_out
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_2)
    movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_2)
    set_digital_output(DO_GRIPPER, 1)
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_2)

# first object if you need
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1)

movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE)
```

- **Related Commands**

**tracking\_conveyor(), untracking\_conveyor()**

## 11.2.4 tracking\_conveyor(conv\_id)

### ▪ Features

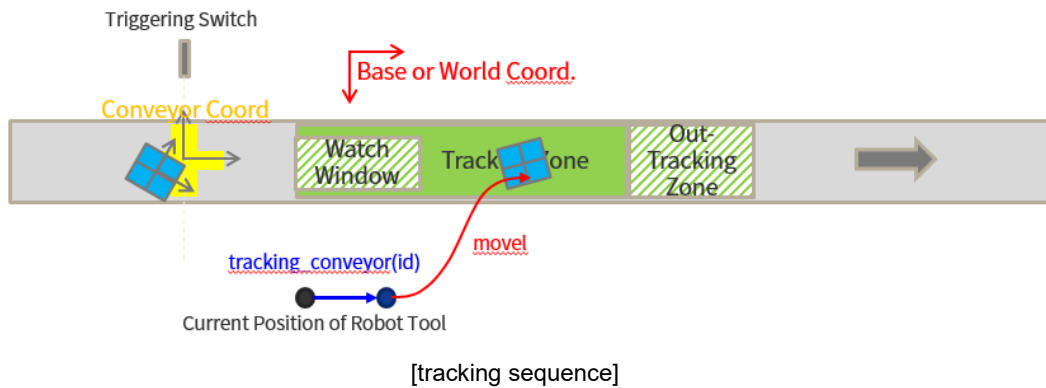
The robot starts Conveyor Tracking. It returns a function when it reaches conveyor velocity by accelerating from stop.

### ▪ Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| conv_id        | int       | -             | Conveyor ID |

### Note

If the tracking\_conveyor command is given, the conveyor starts tracking, and as it accelerates from stop and reaches the conveyor velocity, it returns a function. After a function is returned, it is possible to perform task motion.



### ▪ Return

| Value            | Description   |
|------------------|---|
| 0                | Velocity synchronization success  |
| Negative integer | If the robot is expected to exit the robot work space during velocity synchronization |

### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |

| Exception                | Description                   |
|--------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

### ▪ Example

```
CONV1 = set_conveyor('conveyor1')

while True:
    CONV_COORD_1 = get_conveyor_obj(CONV1)

    tracking_conveyor(CONV1)    # start moving to track conveyor

    # task on conveyor
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
    movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
    set_digital_output(DO_GRIPPER, 1)
    movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

    untracking_conveyor(CONV1)
    obj_count = obj_count + 1
```

### ▪ Related Commands

**untracking\_conveyor()**

### 11.2.5 untracking\_conveyor(conv\_id, time=None)

#### ▪ Features

The robot ends Conveyor Tracking. Return is made when end motion is complete and velocity reaches 0.

#### ▪ Parameter

| Parameter Name | Data Type | Default Value | Description                             |
|----------------|-----------|---------------|---|
| conv_id        | int       | -             | Conveyor ID                             |
| time           | float     | None          | Deceleration time (sec) to end Tracking |

#### Note

- If the time is set to None, it decelerates with the velocity set to Default. If cycle time reduction is necessary, adjust the deceleration time. If a time value is shorter than the robot's maximum deceleration speed, the robot ignores the entered value and decelerates using the maximum deceleration speed.

#### ▪ Return

| Value            | Description   |
|------------------|---|
| 0                | End velocity synchronization success  |
| Negative integer | If the robot is expected to exit the robot work space during end velocity synchronization |

#### ▪ Exception

| Exception                   | Description                       |
|-----------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data error occurred     |
| DR_Error (DR_ERROR_VALUE)   | Invalid parameter value occurred  |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully     |

#### ▪ Example

```
CONV1 = set_conveyor('conveyor1')

while True:
    CONV_COORD_1 = get_conveyor_obj(CONV1)
    tracking_conveyor(CONV1)

    # task on conveyor
    movel(posx(0,0, 50, 0, 0), ref=CONV_COORD_1)
```



```
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1, 0.1)
```

### ▪ Related Commands

tracking\_conveyor()

## 12. A-Series Command

### 12.1 get\_function\_input(index)

- **Features**

This function reads a state of the function button from the process button device.

- **Parameters**

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| index          | int       | -             | It is the index of the function button mounted on the process button to be read. It is available 1 to 4. |

- **Return**

| Value          | Description |
|----------------|-------------|
| 1              | ON          |
| 0              | OFF         |
| Negative value | Failed      |

- **Exception**

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

- **Example**

```
in1 = get_function_input(1)    # Reads the no. 1 function button input
in8 = get_function_input(4)    # Reads the no. 4 function button input
```

## 12.2 flange\_serial\_open(baudrate=115200, bytesize=DR\_EIGHTBITS, parity=DR\_PARITY\_NONE, stopbits=DR\_STOPBITS\_ONE)

### ▪ Features

This command is for opening the pseudo flange serial communication port.

The characteristics of pseudo flange serial communication are different from normal serial communication. Therefore, handshaking communication is recommended. (e.g., modbus RTU) Due to the internal buffer size limit (255bytes) and internal delay, overflow may occur when used in sensors, etc.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| baudrate       | int       | 115200        | Baud rate<br>2400, 4800, 9600, 19200, 38400, 57600, 115200 etc   |
| bytesize       | int       | 8             | Number of data bits<br>• DR_FIVEBITS: 5<br>• DR_SIXBITS: 6<br>• DR_SEVENBITS: 7<br>• DR_EIGHTBITS: 8   |
| parity         | str       | "N"           | Parity checking<br>• DR_PARITY_NONE: "N"<br>• DR_PARITY_EVEN: "E"<br>• DR_PARITY_ODD: "O"<br>• DR_PARITY_MARK: "M"<br>• DR_PARITY_SPACE: "S" |
| stopbits       | int       | 1             | Number of stop bits<br>• DR_STOPBITS_ONE = 1<br>• DR_STOPBITS_TWO = 2  |

### ▪ Return

| Value          | Description           |
|----------------|-----------------------|
| 0              | Successful connection |
| Negative value | Fail to connection    |

### ▪ Exception

| Exception                | Description                        |
|--------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |

**flange\_serial\_open(baudrate=115200, bytesize=DR\_EIGHTBITS,  
parity=DR\_PARITY\_NONE, stopbits=DR\_STOPBITS\_ONE)**

---

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

## 12.3 flange\_serial\_close()

### ▪ Features

This function closes a flange serial communication port.

### ▪ Return

| Value | Description |
|-------|-------------|
| 0     | Success     |

### ▪ Exception

| Exception                   | Description                        |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred |
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred  |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully      |

## 12.4 flange\_serial\_write(tx\_data)

### ▪ Features

This function records the data (data) to a flange serial port.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description  |
|----------------|-----------|---------------|--|
| tx_data        | byte      | -             | Data to be transmitted (max 32byte) <ul style="list-style-type: none"> <li>• The data type must be a byte.</li> <li>• Refer to the example below.</li> </ul> |

### ▪ Return

| Value | Description |
|-------|-------------|
| 0     | Success     |

### ▪ Exception

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

## 12.5 flange\_serial\_read()

### ▪ Features

This function reads the data from a flange serial port.

### ▪ Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| -              | -         | -             | -           |

### ▪ Return

| Value (res, rx_data) | Description                          |
|----------------------|--------------------------------------|
| res                  | Number of bytes of the received data |
| rx_data              | Number of bytes read (byte type)     |

### ▪ Exception

| Exception                   | Description                        |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE)    | Parameter data type error occurred |
| DR_Error (DR_ERROR_RUNTIME) | C extension module error occurred  |
| DR_Error (DR_ERROR_STOP)    | Program terminated forcefully      |



**Doosan Robotics**

[www.doosanrobotics.com](http://www.doosanrobotics.com)