

Software version : V2.9.3
Document version : V2.6
Original instructions(en_US)

Programming manual(V2.9.3)



M0609 | M0617 | M1013 | M1509 | H2017 | H2515
A0509 | A0509S | A0912 | A0912S

DOOSAN

© 2022 Doosan Robotics Inc.

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Preface | 15 |
| 1.1 | Copyright | 15 |
| 2 | DRL Basic Syntex | 16 |
| 2.1 | Basic Syntax | 16 |
| 2.1.1 | Indent..... | 16 |
| 2.1.2 | Comment | 17 |
| 2.2 | Variable | 17 |
| 2.2.1 | Variable name | 17 |
| 2.2.2 | Numeric value | 18 |
| 2.2.3 | String | 19 |
| 2.2.4 | list | 20 |
| 2.2.5 | tuple | 20 |
| 2.2.6 | dictionary | 21 |
| 2.3 | Function | 21 |
| 2.3.1 | Function Syntax..... | 21 |
| 2.3.2 | Scoping rule | 23 |
| 2.3.3 | Parameter mode..... | 23 |
| 2.4 | Control Statement | 24 |
| 2.4.1 | pass | 24 |
| 2.4.2 | if | 25 |
| 2.4.3 | while | 26 |
| 2.4.4 | for | 26 |
| 2.4.5 | break | 27 |
| 2.4.6 | continue | 27 |
| 2.4.7 | Else in a loop | 28 |
| 3 | Motion-related Commands | 29 |
| 3.1 | Pos Creation | 29 |
| 3.1.1 | posj(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0) | 29 |
| 3.1.2 | posx(X=0, Y=0, Z=0, A=0, B=0, C=0) | 30 |

| | | |
|------------|--|-----------|
| 3.1.3 | trans(pos, delta, ref, ref_out)..... | 32 |
| 3.1.4 | posb(seg_type, posx1, posx2=None, radius=0) | 34 |
| 3.1.5 | fkin(pos, ref) | 36 |
| 3.1.6 | ikin(pos, sol_space, ref, ref_pos_opt, iter_threshold) | 38 |
| 3.1.7 | addto(pos, add_val=None) | 40 |
| 3.2 | Motion settings..... | 42 |
| 3.2.1 | set_velj(vel) | 42 |
| 3.2.2 | set_accj(acc) | 43 |
| 3.2.3 | set_velx(vel1, vel2) | 45 |
| 3.2.4 | set_velx(vel) | 46 |
| 3.2.5 | set_accx(acc1, acc2)..... | 48 |
| 3.2.6 | set_accx(acc)..... | 49 |
| 3.2.7 | set_tcp(name) | 51 |
| 3.2.8 | set_ref_coord(coord) | 52 |
| 3.3 | Synchronous Motion | 54 |
| 3.3.1 | movej() | 54 |
| 3.3.2 | movel() | 59 |
| 3.3.3 | movejx() | 64 |
| 3.3.4 | movec() | 68 |
| 3.3.5 | movesj()..... | 74 |
| 3.3.6 | movesx() | 77 |
| 3.3.7 | moveb() | 81 |
| 3.3.8 | move_spiral() | 85 |
| 3.3.9 | move_periodic() | 89 |
| 3.3.10 | move_home() | 94 |
| 3.4 | Asynchronous Motion..... | 95 |
| 3.4.1 | amovej() | 95 |
| 3.4.2 | amovel() | 98 |
| 3.4.3 | amovejx()..... | 101 |
| 3.4.4 | amovec()..... | 104 |
| 3.4.5 | amovesj() | 108 |
| 3.4.6 | amovesx() | 111 |
| 3.4.7 | amoveb() | 114 |
| 3.4.8 | amove_spiral() | 118 |
| 3.4.9 | amove_periodic() | 122 |

| | | |
|------------|---|------------|
| 3.5 | Additional Functions | 125 |
| 3.5.1 | mwait(time=0)..... | 125 |
| 3.5.2 | begin_blend(radius=0) | 127 |
| 3.5.3 | end_blend()..... | 129 |
| 3.5.4 | check_motion() | 130 |
| 3.5.5 | stop(st_mode) | 131 |
| 3.5.6 | change_operation_speed(speed) | 133 |
| 3.5.7 | wait_manual_guide() | 135 |
| 3.5.8 | wait_nudge() | 137 |
| 3.5.9 | enable.Alter_motion(n,mode,ref,limit_dPOS,limit_dPOS_per) | 139 |
| 3.5.10 | alter_motion(pos)..... | 141 |
| 3.5.11 | disable.Alter_motion()..... | 143 |
| 3.6 | Servo Motion | 144 |
| 3.6.1 | servoj() | 144 |
| 3.6.2 | servol() | 146 |
| 3.6.3 | speedj()..... | 149 |
| 3.6.4 | speedl()..... | 151 |

4 Auxiliary Control Commands **154**

| | | |
|------------|---|------------|
| 4.1 | Robot Current Value..... | 154 |
| 4.1.1 | get_current_posj() | 154 |
| 4.1.2 | get_current_velj() | 154 |
| 4.1.3 | get_current_posx(ref) | 155 |
| 4.1.4 | get_current_tool_flange_posx(ref) | 157 |
| 4.1.5 | get_current_velx(ref) | 158 |
| 4.1.6 | get_current_rotm(ref)..... | 159 |
| 4.1.7 | get_joint_torque() | 160 |
| 4.1.8 | get_external_torque()..... | 160 |
| 4.1.9 | get_tool_force(ref) | 161 |
| 4.2 | Robot Target Value | 162 |
| 4.2.1 | get_desired_posj() | 162 |
| 4.2.2 | get_desired_velj() | 163 |
| 4.2.3 | get_desired_posx(ref) | 164 |
| 4.2.4 | get_desired_velx(ref) | 165 |
| 4.3 | Control State Value | 166 |

| | | |
|----------|---|------------|
| 4.3.1 | get_control_mode() | 166 |
| 4.3.2 | get_control_space() | 166 |
| 4.3.3 | get_current_solution_space() | 167 |
| 4.3.4 | get_solution_space(pos) | 168 |
| 4.3.5 | get_orientation_error(xd, xc, axis)..... | 169 |
| 5 | Other Settings Command | 171 |
| 5.1 | Tool/Workpiece Settings | 171 |
| 5.1.1 | get_workpiece_weight() | 171 |
| 5.1.2 | reset_workpiece_weight()..... | 172 |
| 5.1.3 | set_workpiece_weight(weight=0.0, cog=[0.0,0.0,0.0], cog_ref=DR_CUR_TCP, add_up=DR_REPLACE, start_time=None, transition_time=None)..... | 172 |
| 5.1.4 | set_tool_shape(name) | 175 |
| 5.1.5 | set_tool(name, start_time, transition_time) | 176 |
| 5.2 | Control Mode Settings..... | 177 |
| 5.2.1 | set_singularity_handling(mode) | 177 |
| 5.2.2 | set_singular_handling_force(mode)..... | 179 |
| 5.2.3 | set_palletizing_mode(mode) | 181 |
| 5.2.4 | set_motion_end(mode) | 182 |
| 6 | Force/Stiffness Control and Other User-Friendly Features..... | 185 |
| 6.1 | Force/Compliance Control | 185 |
| 6.1.1 | release_compliance_ctrl() | 185 |
| 6.1.2 | task_compliance_ctrl(stx, time)..... | 186 |
| 6.1.3 | set_stiffnessx(stx, time) | 187 |
| 6.1.4 | set_desired_force(fd, dir, time, mod) | 189 |
| 6.1.5 | release_force(time=0) | 192 |
| 6.1.6 | get_force_control_state()..... | 193 |
| 6.2 | User-friendly Functions..... | 195 |
| 6.2.1 | parallel_axis(x1, x2, x3, axis, ref) | 195 |
| 6.2.2 | parallel_axis(vect, axis, ref)..... | 197 |
| 6.2.3 | align_axis(x1, x2, x3, pos, axis, ref) | 198 |
| 6.2.4 | align_axis(vect, pos, axis, ref) | 200 |
| 6.2.5 | is_done_bolt_tightening(m=0, timeout=0, axis=None)..... | 202 |
| 6.2.6 | calc_coord(x1, x2, x3, x4, ref, mod) | 203 |

| | | |
|--------|--|-----|
| 6.2.7 | set_user_cart_coord(pos, ref) | 205 |
| 6.2.8 | set_user_cart_coord(x1, x2, x3, pos, ref) | 207 |
| 6.2.9 | set_user_cart_coord(u1, v1, pos, ref) | 208 |
| 6.2.10 | overwrite_user_cart_coord(id, pos, ref) | 210 |
| 6.2.11 | get_user_cart_coord(id) | 211 |
| 6.2.12 | check_position_condition(axis, min, max, ref, mod, pos) | 213 |
| 6.2.13 | check_force_condition(axis, min, max, ref) | 215 |
| 6.2.14 | check_orientation_condition(axis, min, max, ref, mod) | 217 |
| 6.2.15 | check_orientation_condition(axis, min, max, ref, mod, pos) | 219 |
| 6.2.16 | coord_transform(pose_in, ref_in, ref_out) | 222 |

7 System Commands 225

| | | |
|--------|--|-----|
| 7.1 | IO Related | 225 |
| 7.1.1 | set_digital_output(index, val=None) | 225 |
| 7.1.2 | set_digital_outputs(bit_list) | 226 |
| 7.1.3 | set_digital_outputs(bit_start, bit_end, val) | 227 |
| 7.1.4 | set_digital_output(index, val=None, time=None, val2=None) | 229 |
| 7.1.5 | get_digital_input(index) | 231 |
| 7.1.6 | get_digital_inputs(bit_list) | 232 |
| 7.1.7 | get_digital_inputs(bit_start, bit_end) | 233 |
| 7.1.8 | wait_digital_input(index, val, timeout=None) | 234 |
| 7.1.9 | set_tool_digital_output(index, val=None) | 236 |
| 7.1.10 | set_tool_digital_outputs(bit_list) | 237 |
| 7.1.11 | set_tool_digital_outputs(bit_start, bit_end, val) | 239 |
| 7.1.12 | set_tool_digital_output(index, val=None, time=None, val2=None) | 240 |
| 7.1.13 | get_tool_digital_input(index) | 242 |
| 7.1.14 | get_tool_digital_inputs(bit_list) | 243 |
| 7.1.15 | get_tool_digital_inputs(bit_start, bit_end) | 245 |
| 7.1.16 | wait_tool_digital_input(index, val, timeout=None) | 246 |
| 7.1.17 | set_mode_analog_output(ch, mod) | 248 |
| 7.1.18 | set_mode_analog_input(ch, mod) | 249 |
| 7.1.19 | set_analog_output(ch, val) | 250 |
| 7.1.20 | get_analog_input(ch) | 251 |
| 7.2 | TP Interface..... 253 | |
| 7.2.1 | tp_popup(message, pm_type=DR_PM_MESSAGE, button_type=0) | 253 |

| | | |
|--------|---|------------|
| 7.2.2 | <code>tp_log(message)</code> | 254 |
| 7.2.3 | <code>tp_get_user_input(message, input_type)</code> | 255 |
| 7.3 | Thread | 257 |
| 7.3.1 | <code>thread_run(th_func_name, loop=False)</code> | 257 |
| 7.3.2 | <code>thread_stop(th_id)</code> | 259 |
| 7.3.3 | <code>thread_pause(th_id)</code> | 260 |
| 7.3.4 | <code>thread_resume(th_id)</code> | 261 |
| 7.3.5 | <code>thread_state(th_id)</code> | 262 |
| 7.3.6 | <code>Integrated example - Thread</code> | 263 |
| 7.4 | Others | 265 |
| 7.4.1 | <code>wait(time)</code> | 265 |
| 7.4.2 | <code>exit()</code> | 266 |
| 7.4.3 | <code>sub_program_run(name)</code> | 267 |
| 7.4.4 | <code>drl_report_line(option)</code> | 268 |
| 7.4.5 | <code>set_fm(key, value)</code> | 269 |
| 7.4.6 | <code>get_robot_model()</code> | 270 |
| 7.4.7 | <code>get_robot_serial_num()</code> | 271 |
| 7.4.8 | <code>check_robot_jts()</code> | 272 |
| 7.4.9 | <code>check_robot_fts()</code> | 272 |
| 7.4.10 | <code>start_timer()</code> | 273 |
| 7.4.11 | <code>end_timer()</code> | 273 |

| | | |
|----------|--------------------------------------|------------|
| 8 | Mathematical Function | 275 |
| 8.1 | Basic Function | 275 |
| 8.1.1 | <code>ceil(x)</code> | 275 |
| 8.1.2 | <code>floor(x)</code> | 275 |
| 8.1.3 | <code>pow(x, y)</code> | 276 |
| 8.1.4 | <code>sqrt(x)</code> | 277 |
| 8.1.5 | <code>log(x, b)</code> | 277 |
| 8.1.6 | <code>d2r(x)</code> | 278 |
| 8.1.7 | <code>r2d(x)</code> | 279 |
| 8.1.8 | <code>random()</code> | 279 |
| 8.2 | Trigonometric functions | 280 |
| 8.2.1 | <code>sin(x)</code> | 280 |
| 8.2.2 | <code>cos(x)</code> | 281 |

| | | |
|------------|---|------------|
| 8.2.3 | <code>tan(x)</code> | 281 |
| 8.2.4 | <code>asin(x)</code> | 282 |
| 8.2.5 | <code>acos(x)</code> | 283 |
| 8.2.6 | <code>atan(x)</code> | 283 |
| 8.2.7 | <code>atan2(y, x)</code> | 284 |
| 8.3 | Linear algebra | 285 |
| 8.3.1 | <code>norm(x)</code> | 285 |
| 8.3.2 | <code>rotx(angle)</code> | 285 |
| 8.3.3 | <code>roty(angle)</code> | 286 |
| 8.3.4 | <code>rotz(angle)</code> | 287 |
| 8.3.5 | <code>rotm2eul(rotm)</code> | 288 |
| 8.3.6 | <code>rotm2rotvec(rotm)</code> | 289 |
| 8.3.7 | <code>eul2rotm([alpha,beta,gamma])</code> | 289 |
| 8.3.8 | <code>eul2rotvec([alpha,beta,gamma])</code> | 290 |
| 8.3.9 | <code>rotvec2eul([rx,ry,rz])</code> | 291 |
| 8.3.10 | <code>rotvec2rotm([rx,ry,rz])</code> | 292 |
| 8.3.11 | <code>htrans(posx1,posx2)</code> | 293 |
| 8.3.12 | <code>get_intermediate_pose(posx1,posx2,alpha)</code> | 294 |
| 8.3.13 | <code>get_distance(posx1, posx2)</code> | 295 |
| 8.3.14 | <code>get_normal(x1, x2, x3)</code> | 296 |
| 8.3.15 | <code>add_pose(posx1,posx2)</code> | 297 |
| 8.3.16 | <code>subtract_pose(posx1,posx2)</code> | 298 |
| 8.3.17 | <code>inverse_pose(posx1)</code> | 299 |
| 8.3.18 | <code>dot_pose(posx1, posx2)</code> | 300 |
| 8.3.19 | <code>cross_pose(posx1, posx2)</code> | 301 |
| 8.3.20 | <code>unit_pose(posx1)</code> | 302 |

9 External Communication Commands..... 304

| | | |
|------------|--|------------|
| 9.1 | Serial | 304 |
| 9.1.1 | <code>serial_open(port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)</code> | 304 |
| 9.1.2 | <code>serial_close(ser)</code> | 306 |
| 9.1.3 | <code>serial_state(ser)</code> | 306 |
| 9.1.4 | <code>serial_set_inter_byte_timeout(ser, timeout=None)</code> | 307 |
| 9.1.5 | <code>serial_write(ser, tx_data)</code> | 308 |

| | | |
|--------|---|-----|
| 9.1.6 | serial_read(ser, length=-1, timeout=-1) | 310 |
| 9.1.7 | serial_get_count() | 311 |
| 9.1.8 | serial_get_info(id) | 312 |
| 9.1.9 | Combined Example - Serial | 313 |
| 9.2 | Tcp/Client | 315 |
| 9.2.1 | client_socket_open(ip, port)..... | 315 |
| 9.2.2 | client_socket_close(sock)..... | 316 |
| 9.2.3 | client_socket_state(sock)..... | 317 |
| 9.2.4 | client_socket_write(sock, tx_data) | 319 |
| 9.2.5 | client_socket_read(sock, length=-1, timeout=-1) | 320 |
| 9.2.6 | Integrated example (Tcp/Client) | 321 |
| 9.3 | Tcp/Server | 324 |
| 9.3.1 | server_socket_open(port) | 324 |
| 9.3.2 | server_socket_close(sock)..... | 325 |
| 9.3.3 | server_socket_state(sock)..... | 326 |
| 9.3.4 | server_socket_write(sock, tx_data) | 327 |
| 9.3.5 | server_socket_read(sock, length=-1, timeout=-1) | 328 |
| 9.3.6 | Integrated example - Tcp/Server | 330 |
| 9.4 | Modbus | 332 |
| 9.4.1 | add_modbus_signal(ip, port, name, reg_type, index, value=0, slaveid=255)..... | 332 |
| 9.4.2 | 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) | 335 |
| 9.4.3 | add_modbus_signal_multi(ip, port, slaveid=255, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1) | 338 |
| 9.4.4 | add_modbus_rtu_signal_multi(slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1) | 339 |
| 9.4.5 | del_modbus_signal(name) | 341 |
| 9.4.6 | del_modbus_signal_multi(name) | 343 |
| 9.4.7 | set_modbus_output(iobus, value)..... | 344 |
| 9.4.8 | set_modbus_outputs(iobus_list, val_list) | 345 |
| 9.4.9 | set_modbus_output_multi(iobus, val_list) | 347 |
| 9.4.10 | get_modbus_input(iobus) | 348 |
| 9.4.11 | get_modbus_inputs(iobus_list) | 350 |
| 9.4.12 | get_modbus_inputs_list(iobus_list) | 351 |
| 9.4.13 | get_modbus_input_multi (iobus) | 352 |

| | | |
|------------|---|------------|
| 9.4.14 | wait_modbus_input(iobus, val, timeout=None) | 353 |
| 9.4.15 | set_modbus_slave(address, val) | 355 |
| 9.4.16 | get_modbus_slave(address) | 356 |
| 9.4.17 | modbus_crc16(data) | 357 |
| 9.4.18 | modbus_send_make(send_data) | 358 |
| 9.4.19 | modbus_recv_check(recv_data) | 359 |
| 9.4.20 | modbus_unsigned_to_signed(unsigned_data) | 360 |
| 9.5 | Industrial Ethernet (EtherNet/IP,PROFINET) | 361 |
| 9.5.1 | set_output_register_bit(address, val) | 361 |
| 9.5.2 | set_output_register_int(address, val) | 363 |
| 9.5.3 | set_output_register_float(address, val) | 364 |
| 9.5.4 | get_output_register_bit(address) | 365 |
| 9.5.5 | get_output_register_int(address) | 366 |
| 9.5.6 | get_output_register_float(address) | 367 |
| 9.5.7 | get_input_register_bit(address) | 368 |
| 9.5.8 | get_input_register_int(address) | 370 |
| 9.5.9 | get_input_register_float(address) | 371 |
| 9.6 | FOCAS | 372 |
| 9.6.1 | focas_connect(ip, port, timeout) | 372 |
| 9.6.2 | focas_disconnect(handle) | 373 |
| 9.6.3 | focas_pmc_read_bit(handle, addr_type, start_num, bit_offset) | 374 |
| 9.6.4 | focas_pmc_read_char(handle, addr_type, start_num, read_count) | 376 |
| 9.6.5 | focas_pmc_read_word(handle, addr_type, start_num, read_count) | 378 |
| 9.6.6 | focas_pmc_read_long(handle, addr_type, start_num, read_count) | 380 |
| 9.6.7 | focas_pmc_read_float(handle, addr_type, start_num, read_count) | 382 |
| 9.6.8 | focas_pmc_read_double(handle, addr_type, start_num, read_count) | 384 |
| 9.6.9 | focas_pmc_write_bit(handle, addr_type, start_num, bit_offset, write_data) | 386 |
| 9.6.10 | focas_pmc_write_char(handle, addr_type, start_num, write_data, write_count) | 388 |
| 9.6.11 | focas_pmc_write_word(handle, addr_type, start_num, write_data, write_count) | 390 |
| 9.6.12 | focas_pmc_write_long(handle, addr_type, start_num, write_data, write_count) | 392 |
| 9.6.13 | focas_pmc_write_float(handle, addr_type, start_num, write_data, write_count) | 394 |
| 9.6.14 | focas_pmc_write_double(handle, addr_type, start_num, write_data, write_count) | 396 |
| 9.6.15 | focas_get_error_str(handle, errorCode) | 398 |

10 External Vision Commands **403**

| | | |
|-------------|---|------------|
| 10.1 | Overview | 403 |
| 10.2 | 2D Vision - COGNEX / SICK / VISOR..... | 403 |
| 10.2.1 | vs_set_info(type)..... | 404 |
| 10.2.2 | vs_connect(ip_addr, port_num=9999) | 405 |
| 10.2.3 | vs_disconnect() | 406 |
| 10.2.4 | vs_get_job() | 406 |
| 10.2.5 | vs_set_job(job_name) | 407 |
| 10.2.6 | cognex_set_integer (job_name, integer_number) | 408 |
| 10.2.7 | visor_job_change(index) | 408 |
| 10.2.8 | vs_trigger() | 410 |
| 10.2.9 | vs_set_init_pos(vision_posx_init, robot_posx_init, vs_pos=1)..... | 411 |
| 10.2.10 | vs_get_offset_pos(vision_posx_meas, vs_pos=1) | 412 |
| 10.2.11 | Integrated example - DR_VS_COGNEX, DR_VS_SICK..... | 413 |
| 10.2.12 | vs_request(cmd)..... | 414 |
| 10.2.13 | vs_result()..... | 415 |
| 10.2.14 | Integrated example - DR_VS_CUSTOM | 416 |
| 10.3 | Pickit 3D | 417 |
| 10.3.1 | pickit_connect(ip)..... | 418 |
| 10.3.2 | pickit_disconnect() | 419 |
| 10.3.3 | pickit_change_configuration(setup_id, product_id)) | 419 |
| 10.3.4 | pickit_request_calibration() | 420 |
| 10.3.5 | pickit_detection(offset_z) | 420 |
| 10.3.6 | pickit_next_object(offset_z)..... | 422 |
| 10.3.7 | pickit_save_snapshot() | 424 |
| 10.3.8 | Integrated example - Pickit 3D | 425 |
| 11 | Doosan Vision(SVM) Command..... | 428 |
| 11.1 | Camera Connection..... | 429 |
| 11.1.1 | svm_connect(ip="192.168.137.5", port=20)..... | 429 |
| 11.1.2 | svm_disconnect() | 430 |
| 11.2 | Check Image Quality | 430 |
| 11.2.1 | svm_set_led_brightness(value) | 430 |
| 11.2.2 | svm_get_led_brightness() | 431 |
| 11.2.3 | svm_set_camera_exp_val(value) | 431 |
| 11.2.4 | svm_set_camera_gain_val(value) | 432 |

| | | |
|--------|--|------------|
| 11.2.5 | svm_set_camera_load(job_id) | 433 |
| 11.3 | Vision Job Manage | 433 |
| 11.3.1 | svm_set_job(job_id) | 433 |
| 11.4 | Camera Calibration | 434 |
| 11.4.1 | svm_get_robot_pose(job_id) | 434 |
| 11.5 | Object Recognition/Detection | 435 |
| 11.5.1 | svm_get_vision_info(job_id)..... | 435 |
| 11.5.2 | svm_get_variable(tool_id, var_type)..... | 436 |
| 11.5.3 | svm_detect_landmark(job_id)..... | 437 |
| 11.5.4 | svm_get_marker_offset_pose(cpos, offset, euler_mode) | 438 |
| 11.5.5 | svm_get_detect_barcode()..... | 440 |
| 11.5.6 | svm_get_barcode_db_data(index) | 440 |
| 11.5.7 | svm_compare_barcode_db_data(dbdata) | 441 |
| 11.6 | Robot Task | 442 |
| 11.6.1 | svm_set_init_pos_data(Id_list, Pos_list)..... | 442 |
| 11.6.2 | svm_get_offset_pos(posx_robot_init, job_id, tool_id) | 443 |
| 11.7 | Other SVM commands | 445 |
| 11.7.1 | svm_set_tp_popup (svm_flag) | 445 |
| 11.8 | Integrated example (SVM) | 445 |
| 11.8.1 | Example..... | 445 |

| | | |
|-----------|--|------------|
| 12 | Application Commands | 447 |
| 12.1 | External Encoder Setting Commands | 447 |
| 12.1.1 | set_extenc_polarity(channel, polarity_A, polarity_B, polarity_Z, polarity_S) | 447 |
| 12.1.2 | set_extenc_mode(channel, mode_AB, pulse_AZ, mode_Z, mode_S, inverse_cnt) | 448 |
| 12.1.3 | get_extenc_count(channel) | 450 |
| 12.1.4 | clear_extenc_count(channel) | 451 |
| 12.2 | Conveyor Tracking | 452 |
| 12.2.1 | set_conveyor(name) | 452 |
| 12.2.2 | set_conveyor_ex(name="", conv_type=0, encoder_channel=1, triggering_mute_time=0.0, count_per_dist=5000, conv_coord=posx(0,0,0,0,0), ref=DR_BASE, conv_speed=100.0, speed_filter_size=500, min_dist=0.0, max_dist=1000.0, watch_window=100.0...)..... | 453 |
| 12.2.3 | get_conveyor_obj(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None)..... | 457 |
| 12.2.4 | tracking_conveyor(conv_id, time=0.3) | 461 |

| | | |
|-------------|--|------------|
| 12.2.5 | untracking_conveyor(conv_id, time=0.3) | 463 |
| 12.3 | Welding..... | 465 |
| 12.3.1 | app_weld_enable_digital() | 465 |
| 12.3.2 | app_weld_disable_digital() | 467 |
| 12.3.3 | app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0,0,0,0], robot_ready=[0,0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0,0]) | 469 |
| 12.3.4 | app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0,0],wm_opt1=[0,0,0,0,0,0,0,0]) .474 | |
| 12.3.5 | app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)..... | 477 |
| 12.3.6 | app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])..... | 480 |
| 12.3.7 | app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0],...)..... | 483 |
| 12.3.8 | app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0]) | 487 |
| 12.3.9 | app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...) | 491 |
| 12.3.10 | app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0],...) | 494 |
| 12.3.11 | app_weld_set_weld_cond_digital(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...)..... | 498 |
| 12.3.12 | app_weld_adj_welding_cond_digital(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None) | 503 |
| 12.3.13 | app_weld_get_welding_cond_digital() | 507 |
| 12.3.14 | app_weld_enable_analog(ch_v_out=[1,0], spec_v_out=[0,0,0,0], ch_f_out=[2,0], spec_f_out=[0,0,0,0], ch_v_in=[1,0], spec_v_in=[0,0,0,0], ch_c_in=[2,0], spec_c_in=[0,0,0,0],ch_arc_on=1,ch_gas_on=2,ch_inching_fwd=3,ch_inching_bwd=4, ...) .512 | |
| 12.3.15 | app_weld_disable_analog() | 517 |
| 12.3.16 | app_weld_set_weld_cond_analog(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5]) | 519 |
| 12.3.17 | app_weld_adj_welding_cond_analog(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None) | 523 |

| | | |
|---------|--|-----|
| 12.3.18 | app_weld_get_welding_cond_analog() | 526 |
| 12.3.19 | app_weld_weave_cond_trapezoidal(wv_offset=[0,0], wv_ang=0, wv_param=[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3]) | 530 |
| 12.3.20 | app_weld_weave_cond_zigzag(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6]) | 532 |
| 12.3.21 | app_weld_weave_cond_circular(wv_offset=[0,0], wv_ang=0, wv_param=[3,3,0.3,0.3]) | 535 |
| 12.3.22 | app_weld_weave_cond_sinusoidal(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6])..... | 537 |

13 A-Series Command.....540

| | | |
|--------|---|-----|
| 13.1 | Controller | 540 |
| 13.1.1 | get_function_input(index)..... | 540 |
| 13.2 | Flange I/O..... | 541 |
| 13.2.1 | flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)..... | 541 |
| 13.2.2 | flange_serial_close() | 542 |
| 13.2.3 | flange_serial_write(tx_data) | 543 |
| 13.2.4 | flange_serial_read(timeout=None) | 544 |
| 13.2.5 | Integrated example - A-Series Command | 545 |

1 Preface

This manual is composed of thirteen chapters. Chapter 2 through 12 describe DRL commands common to M-series robot, H-series robot and A-series robot, chapter 13 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/>)

1.1 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.

© 2022 Doosan Robotics Inc., All rights reserved

2 DRL Basic Syntex

2.1 Basic Syntax

⚠ Caution

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

DRL only supports the information described in this manual.

2.1.1 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

```

1  Code block 1
2  [TAB] Code block 2
3
4  Example)
5  if x == 3 :
6    y += 1
7
8  # No Error
9  def fnSum(x,y):
10   [space4] sum = x + y
11   [space4] return sum
12
13 # No Error
14 def fnSubtract(x,y):
15   [TAB]diff = x - y
16   [TAB] return diff
17
18
19 # Indentation error
20 def fnProduct(x,y):
21   [space4]product = x * y
22   [TAB] return product

```

2.1.2 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

```

1 # Comment example 1
2   ''
3     Comment example 2

```

2.2 Variable

2.2.1 Variable name

Features

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

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

⚠ Caution

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

| | | | | |
|-------------|----------------|--------------|--------------|-----------------|
| and | assert | break | class | continue |
| def | del | elif | else | except |
| exec | finally | for | from | global |

| | | | | |
|--------------|---------------|---------------|------------|---------------|
| if | import | in | is | lambda |
| list | not | open | or | pass |
| print | raise | return | try | while |
| with | yield | select | | |

Example

```

1 friend = 10
2 Friend = 1
3 _myFriend = None
4
5 Pass = 10 # Syntax error
6 movei = 0 # movej is a DRL instruction name and should not be
    used as a variable.

```

2.2.2 Numeric value

Features

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

Example

```

1 10 , 0x10 , 0o10 , 0b10
2 3.14 , 314e - 2
3 x = 3 - 4j
4
5 int_value = 10
6 hexa_value = 0x10
7 octa_value = 0o10
8 binary_value = 0b10
9 double_value = 3.14
10 double_value = 314e - 2
11 complex_value = 3 - 4j

```

2.2.3 String

String

Features

All character strings are in Unicode.

- Escape characters
 - \n: New line
 - \t: Tab
 - \r: Carriage 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

```

1  "string1"
2  'string2'
3
4  tp_log( "st" + "ring" )
5      #expected result: string
6  tp_log( "str" * 3 )
7      #expected result: strstrstrstr
8  tp_log( "line1\nline2" )
9      #expected result: line1
10     # line2
11  tp_log( "\"string\\\"\"")
12      #expected result: "string"
13  tp_log( "str" [ 0 ] )
14      #expected result: s
15  tp_log( "string" [ 1 : 3 ] )
16      #expected result: tr

```

`+, *`

Example

| | |
|---|---|
| 1 | <code>"Doosan" + "Robotics" → "DoosanRobotics"</code> |
| 2 | <code>"Doo" * 3 → "DooDooDoo"</code> |

Indexing & slicing

Example

| | |
|---|--|
| 1 | <code>" Doosan" [0] → "D"</code> |
| 2 | <code>" Doosan" [1 : 4] → "oos"</code> |

2.2.4 list

Features

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

Example

| | |
|----|--|
| 1 | <code>colors = ["red" , "green" , "blue"]</code> |
| 2 | <code>tp_log(colors[0] + "," + colors[1] + "," + colors[2])</code> |
| 3 | <code>#expected print result: red,green,blue</code> |
| 4 | |
| 5 | <code>numbers = [1 , 3 , 5 , 7 , 9]</code> |
| 6 | <code>sum = 0</code> |
| 7 | <code>for number in numbers:</code> |
| 8 | <code> sum += number</code> |
| 9 | <code>tp_log(str (sum))</code> |
| 10 | <code>#expected result: 25</code> |

2.2.5 tuple

Features

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

Example

```

1 colors = ( "red" , "green" , "blue" )
2 numbers = ( 1 , 3 , 5 , 7 , 9 )
3
4 def fnMinMax(numbers):
5     numbers.sort()
6     return (numbers[ 0 ] , numbers[ - 1 ])
7 minmax = fnMinMax( [ 4 , 1 , 2 , 9 , 5 , 7 ] )
8 tp_log( "Min Value= " + str(minmax[ 0 ]))
9     #expected result: Min Value = 1
10 tp_log( "Max Value= " + str(minmax[ 1 ]))
11     #expected result: Max Value = 9

```

2.2.6 dictionary

Features

Dictionary specifies the keys and values and lists the values.

Example

```

1 d = dict (a = 1 , b = 3 , c = 5 )
2
3 colors = dict ()
4 colors[ "cherry" ] = "red"
5
6 ages = { 'Kim' : 35 , 'Lee' : 38 , 'Chang' : 37 }
7 tp_log( "Ages of Kim = " + str(ages[ 'Kim' ]))
8 #expected print result: Ages of Kim = 35

```

2.3 Function

2.3.1 Function Syntax

Features

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

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

⚠ Caution

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

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

sentence

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

Example

```

1 # Example
2 def fn_Times(a, b):
3     return a * b
4
5 fn_Times ( 10 , 10 )
6
7 def fn_Times(a, b):
8     return a * b
9
10 tp log( str (fn_Times( 10 , 5 )))
11 #expected result: 50
12
13 def movej():
14     return 0          # movej should not be used as a function name
15     as interpreter
16         # internal function name
```

2.3.2 Scoping rule

Features

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

Example

```

1 # Error: can't find simple_pi in circle_area
2 simple pi = 3.14
3 def circle_area(r):
4     return r * r * simple_pi
5
6 # simple pi should be declared as global if it is used in
7 # circle area ok
8 def circle_area_ok(r):
9     global simple pi
10    return r * r * simple_pi
11
12 tp log( str (circle area( 3.0 )))
#expected result: 28.26

```

2.3.3 Parameter mode

Features

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

Example

```

1 def fn Times(a = 10 , b = 20 ):
2     return a * b
3
4 #Example - Default parameter value
5 tp log( str (fn Times( 5 )))
6 #expected result: 100
7

```

```

8 #Example - Keyword parameter
9 tp_log( str (fn_Times(b = 5 )))
10 #expected result: 50
11 tp_log( str (fn_Times(a = 5 , b = 5 )))
12 #expected result: 25
13
14 #Example - arbitrary parameter
15 def fn_myUnion( * args):
16     for arg in args:
17         tp_log( str (arg))
18     fn_myUnion( "red" , 1 )
19         #expected print result: red
20     #                                     1

```

Example - Default parameter value

```

1 def fn_Times(a = 10 , b = 20 )
2     return a * b
3
4 fn_Times( 5 )

```

Example - Keyword parameter

```

1 def fn_Times(a = 10 , b = 20 )
2     return a * b
3
4 fn_Times(a = 5 , b = 5 )

```

Example - Variable parameter

```

1 def fn_myUnion( * ar)
2     .....
3
4 fn_myUnion( "red" , "white" , "black" )

```

2.4 Control Statement

2.4.1 pass

Features

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

Example

```

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

```

2.4.2 if

Features

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

sentense

if <conditional statement>:

<syntax>

if <conditional statement 1>:

<Syntax 1>

elif <conditional statement 2>:

<Syntax 2>

else:

<Syntax 3>

Example - if, elif, else

```

1  numbers = [ 2 , 5 , 7 ]
2  for number in numbers:
3      if number % 2 == 0 :
4          tp_log( str (number) + " is even" )
5      else :
6          tp_log ( str (number) + " is odd" )
7 #expected result:
8 #2 is even
9 #5 is odd
10 #7 is odd

```

2.4.3 while

Features

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

syntax

while <conditional statement>:

<syntax>

Example

```

1 sum = 0
2 cnt = 1
3 while cnt < 10 :
4     sum = sum + cnt
5     cnt = cnt + 1
6 tp_log( "sum = " + str( sum ))
7 #expected result:
8 #sum = 45

```

2.4.4 for

Features

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

syntax

for <item> in <sequential object S>:

<syntax>

Example

```

1 x = 0
2 for i in range( 0 , 3 ): # i is 0 -> 1 -> 2
3     x = x + 1
4
5 sum = 0
6 for i in range( 0 , 10 ):
7     sum = sum + i
8 tp_log( "sum = " + str( sum ))

```

```

9 #expected result:
10 #sum = 45

```

2.4.5 break

Features

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

Example

```

1 x = 0
2 while True :
3     x = x + 1
4     if x > 10 :
5         break
6
7 sum = 0 ;cnt = 0
8 while True :
9     if cnt > 9 :
10        break
11     sum = sum + cnt
12     cnt = cnt + 1
13 tp_log( "sum = " + str( sum ))
14 #expected print result:
15 #sum = 45

```

2.4.6 continue

Features

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

Example

```

1 #<ex> 1
2 x = 0
3 v = 0
4 while True :
5     x = x + 1
6     if x > 10 :
7         continue
8     y += 100
9

```

```

10 #<ex> 2
11 sum = 0
12 for i in range(0, 10):
13     if i % 2 == 0:
14         continue
15     sum = sum + i
16 tp_log("sum of odd numbers = " + str(sum))
17 #sum of odd numbers = 25

```

2.4.7 Else in a loop

Features

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

Example

```

1 L = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
2
3 for i in L:
4     if i % 2 == 0:
5         continue
6 else:
7     tp_log("exit without break")

```

3 Motion-related Commands

3.1 Pos Creation

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

Features

This function designates the joint space angle in coordinate values.

Parameters

| No. | Data Type | Default Value | Description |
|-----|-----------------------|---------------|--|
| J1 | float list posj | 0 | 1-axis angle or angle list or posj |
| J2 | float | 0 | 2-axis angle |
| J3 | float | 0 | 3-axis angle |
| J4 | float | 0 | 4-axis angle |
| J5 | float | 0 | 5-axis angle |
| J6 | float | 0 | 6-axis angle |

Return

Posj

Exception

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

Example

```

1 q1 = posj()                                # q1=posj(0,0,0,0,0,0)
2
3 q2 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
4
5 q3 = posi([ 0 , 30 , 60 , 0 , 90 , 0 ])    #
q3=posj(0,30,60,0,90,0)

```

Related commands

- [movej\(\)](#)(p. 54)
- [amovej\(\)](#)(p. 95)
- [movesj\(\)](#)(p. 74)
- [amovesj\(\)](#)(p. 108)

3.1.2 posx(X=0, Y=0, Z=0, A=0, B=0, C=0)

Features

This function designates the joint space angle in coordinate values.

Parameters

| No. | Data Type | Default Value | Description |
|-----|-----------------------|---------------|--|
| X | float list posx | 0 | X position or position list or posx |
| Y | float | 0 | Y position |
| Z | float | 0 | Z position |
| A | float | 0 | A orientation(z-direction rotation of reference coordinate system) |
| B | float | 0 | B orientation(y-direction rotation of A rotated coordinate system) |

| No. | Data Type | Default Value | Description |
|-----|-----------|---------------|--|
| C | float | 0 | C orientation(z-direction rotation of A and B rotated coordinate system) |

Return

Posx

Exception

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

Example

```

1 movej([ 0 , 0 , 90 , 0 , 90 , 0 ], v = 10 , a = 20 )
2
3 x2 = posx( 400 , 300 , 500 , 0 , 180 , 0 )
4
5 x3 = posx([ 350 , 350 , 450 , 0 , 180 , 0 ])
#x3=posx(350, 350, 450, 0, 180, 0)
6
7 x4 = posx(x2)                                     #x4=posx(400, 300,
8 500, 0, 180, 0)
9 movel(x2, v = 100 , a = 200 )

```

Related commands

- [movel\(\)](#)(p. 59)
- [movec\(\)](#)(p. 68)
- [movejx\(\)](#)(p. 64)
- [amovel\(\)](#)(p. 98)
- [movec\(\)](#)(p. 68)
- [movejx\(\)](#)(p. 64)

3.1.3 trans(pos, delta, ref, ref_out)

Features

- pos (pose) defined based on the ref coordinate is moved/rotated by the amount equal to delta, and then a value converted based on the ref_out coordinate is returned.
- In case that the ref coordinate is the tool coordinate, this function 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"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: user defined |
| ref_out | int | DR_BASE | reference coordinate <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: user defined |

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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  p0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2
3  movej(p0, v = 30 , a = 30 )
4
5
6
7  x1 = posx( 200 , 200 , 200 , 0 , 180 , 0 )
8
9  delta = [ 100 , 100 , 100 , 0 , 0 , 0 ]
10
11 x2 = trans(x1, delta, DR_BASE, DR_BASE)
12
13
14 x1_base = posx( 500 , 45 , 700 , 0 , 180 , 0 )
15
16 x4 = trans(x1_base, [ 10 , 0 , 0 , 0 , 0 , 0 ], DR_TOOL)
17
18 movel(x4, v = 100 , a = 100 , ref = DR_BASE)
19
20
21
22 uu1 = [ 1 , 1 , 0 ]
23
24 vv1 = [ -1 , 1 , 0 ]
25
26 pos = posx( 559 , 34.5 , 651.5 , 0 , 180.0 , 0 )
27
28

```

```

29 DR_userTC1 = set_user_cart_coord(uu1, vv1, pos) #user defined
30 coordinate system
31 x1_userTC1 = posx( 30 , 20 , 100 , 0 , 180 , 0 )
32 #posx on user coordinate system
33 x9 = trans(x1_userTC1, [ 0 , 0 , 50 , 0 , 0 , 0 ],
34 DR_userTC1, DR_BASE)
35 movel(x9, v = 100 , a = 100 , ref = DR_BASE)

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [addto\(pos, add_val=None\)\(p. 40\)](#)

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

Features

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

Parameters

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

Return

Posb

Exception

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

Example

```

1  q0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2
3  movej(q0,vel = 30 ,acc = 60 )
4
5  x0 = posx( 564 , 34 , 690 , 0 , 180 , 0 )
6
7  movel(x0, vel = 200 , acc = 400 )      # Moves to the start
position.
8
9
10
11 x1 = posx( 564 , 200 , 690 , 0 , 180 , 0 )
12
13 seg1 = posb(DR_LINE, x1, radius = 40 )
14
15 x2 = posx( 564 , 100 , 590 , 0 , 180 , 0 )
16
17 x2c = posx( 564 , 200 , 490 , 0 , 180 , 0 )
18
19 seg2 = posb(DR_CIRCLE, x2, x2c, radius = 40 )
20
21 x3 = posx( 564 , 300 , 490 , 0 , 180 , 0 )
22
23 seg3 = posb(DR_LINE, x3, radius = 40 )
24
25 x4 = posx( 564 , 400 , 590 , 0 , 180 , 0 )
26
27 x4c = posx( 564 , 300 , 690 , 0 , 180 , 0 )
28
29 seg4 = posb(DR_CIRCLE, x4, x4c, radius = 40 )
30
31 x5 = posx( 664 , 300 , 690 , 0 , 180 , 0 )
32
33 seg5 = posb(DR_LINE, x5, radius = 40 )
34
35 x6 = posx( 564 , 400 , 690 , 0 , 180 , 0 )
36
37 x6c = posx( 664 , 500 , 690 , 0 , 180 , 0 )
38
39 seg6 = posb(DR_CIRCLE, x6, x6c, radius = 40 )

```

```

40
41   x7 = posx( 664 , 400 , 690 , 0 , 180 , 0 )
42
43   seg7 = posb(DR_LINE, x7, radius = 40 )
44
45   x8 = posx( 664 , 400 , 590 , 0 , 180 , 0 )
46
47   x8c = posx( 564 , 400 , 490 , 0 , 180 , 0 )
48
49   seg8 = posb(DR_CIRCLE, x8, x8c, radius = 0 )           # The last
radius must be 0.
      # If not 0, it is processed as 0.
50
51
52
53
54   b_list = [seg1, seg2, seg3, seg4, seg5, seg6, seg7, seg8]
55
56
57
58   moveb(b_list, vel = 200 , acc = 400 )

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [moveb\(\)\(p. 81\)](#)
- [amoveb\(\)\(p. 114\)](#)

3.1.5 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 position list |
| | list (float[6]) | | |

| Parameter Name | Data type | Default Value | Description |
|----------------|-----------|---------------|--|
| ref | int | DR_BASE | <p>reference coordinate</p> <ul style="list-style-type: none"> • DR_BASE : base coordinate • 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 |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |

Example

```

1 q1 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movej(q1,v = 10 ,a = 20 )
3
4 q2 = posj( 30 , 0 , 90 , 0 , 90 , 0 )
5 x2 = fkin(q2, DR_WORLD)
6 # x2: Space coordinate at the edge of the robot (TCP)
7 # corresponding to joint value q2
8 movel(x2,v = 100 ,a = 200 ,ref = DR_WORLD) # Linear motion to x2
9

```

Related commands

- [set_tcp\(name\)\(p. 51\)](#)
 - The tcp information of the name registered in the teach pendant is reflected during fkin operation.
- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)

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

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. Joint position is returned according to closest joint position depend on option(ref_pos_opt). Through status of return value, you can check whether current robot pose is in wrist singularity or out of operating region.

i Note

After SW version V2.9, if the accuracy of robot is corrected when using this function, the accuracy correction algorithm operates. The level of accuracy is adjustable according to the option (iter_threshold). Also, depending on the robot posture, there may be a delay of up to 0.1 seconds.

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 <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate |
| ref_pos_opt | int | 0 | Determine closest joint position depend on option among mu <ul style="list-style-type: none"> • 0 : posj(0,0,0,0,0,0) is reference • 1 : current joint position is reference |
| iter_threshold | list (float[2]) | 0.005 | If the accuracy has been corrected, the level of the accuracy correction The norm value of TCP position (X, Y, Z) [mm] |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| | | 0.01 | If the accuracy has been corrected, the level of the accuracy correction The norm value of TCP orientation (A, B, C) [deg] |

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 |

Return

| Value | Description |
|--------|--|
| posj | Joint space point |
| status | <ul style="list-style-type: none"> • 0 : Return joint position • 1 : Out of workspace • 2 : In wrist singularity region |

Exception

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

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

Example

```

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

```

Related commands

- [set_tcp\(name\)\(p.51\)](#)
- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)

3.1.7 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 position list |
| | list (float[6]) | | |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|---|
| add_val | list (float[6]) | None | List of add values to be added to the position <ul style="list-style-type: none"> 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

```

1  q1 = posj( 10 , 20 , 30 , 40 , 50 , 60 )
2
3  movej (q1, v = 10 , a = 20 )
4
5  q2 = addto(q1, [ 0 , 0 , 0 , 0 , 45 , 0 ])
6
7  movei (q2, v = 10 , a = 20 ) # The robot moves to the joint (10,
8  20, 30, 40, 95, 60).
9  q3 = addto(q2, [])
10 q4 = addto(q3)

```

Related commands

- posj(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0)(p. 29)

3.2 Motion settings

3.2.1 set_velj(vel)

Features

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

Parameters

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

Return

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

Exception

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

Example

```

1 #1
2 Q1 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
3 Q2 = posi( 0 , 0 , 0 , 0 , 90 , 0 )
4 movej(Q1, vel = 10 , acc = 20 )
5 set_velj( 30 ) # The global joint velocity is set to 30 (deg/
sec).

```

```

6   set_accj( 60 ) # The global joint acceleration is set to 60 (deg/
7   sec2). [See set_accj().]
8   movei(Q2)      # The joint motion velocity to Q2 is 30 (deg/sec)
9   which is the global velocity.
10  movei(Q1, vel = 20 , acc = 40 ) # The joint motion velocity to
11  Q1 is 20 (deg/sec) which is the specified velocity.
#2
10  set_veli( 20.5 ) # Decimal point input is possible.
11  set_veli([ 10 , 10 , 20 , 20 , 30 , 10 ]) # The global
velocity can be specified to each axis.

```

Related commands

- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [movej\(\)\(p. 54\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movesj\(\)\(p. 74\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovesj\(\)\(p. 108\)](#)

3.2.2 set_accj(acc)

Features

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

Parameters

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

Return

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

Exception

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

Example

```

1      #1
2      Q1 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
3      Q2 = posi( 0 , 0 , 0 , 0 , 90 , 0 )
4      movei(Q1, vel = 10 , acc = 20 )
5      set veli( 30 ) # The global joint velocity is set to 30 (deg/
sec). [See set veli().]
6      set accj( 60 ) # The global joint acceleration is set to 60 (deg/
sec2).
7      movei(Q2)      # The joint motion acceleration to Q2 is 60(deg/
sec2) which is the global acceleration.
8      movei(Q1, vel = 20 , acc = 40 ) # The joint motion acceleration
to Q1 is 40(deg/sec2) which is the specified acceleration.
9      #2
10     set acci( 30.55 )
11     set_accj([ 30 , 40 , 30 , 30 , 30 , 10 ])

```

Related commands

- [set_velj\(vel\)\(p. 42\)](#)
- [movej\(\)\(p. 54\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movesj\(\)\(p. 74\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovesj\(\)\(p. 108\)](#)

3.2.3 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 movei(), amovei(), 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 #1
2 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
3 movei(P0)
4 P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
5 P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
6 movel(P1, vel = 10 , acc = 20 )
7 set_velx( 30 , 20 )    # The global task velocity is set to 30(mm/
sec) and 20(deg/sec).
8 set_accx( 60 , 40 )    # The global task acceleration is set to
60(mm/sec2) and 40(deg/sec2).

```

```

9      movel(P2)          # The task motion velocity to P2 is 30(mm/sec)
10     and 20(deg/sec) which are the global velocity.
11     movel(P1, vel = 20 , acc = 40 )    # The task motion velocity to
12     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\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [movel\(\)\(p. 59\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [amovel\(\)\(p. 98\)](#)
- [amovec\(\)\(p. 104\)](#)
- [amovesx\(\)\(p. 111\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)

3.2.4 set_velx(vel)

Features

This function sets the linear velocity of the task space motion globally. The globally set velocity vel is applied as the default velocity if the task motion such as movel(), amovel(), movec(), movesx() is called without the explicit input of the velocity value. The set value vel defines the linear velocity of the TCP while the rotating velocity of the TCP is determined proportionally to the linear velocity.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| vel | float | - | velocity |

Return

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

Exception

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

Example

```

1 #1
2 p0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
3 movej(p0)
4 P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
5 P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
6 movel(P1, vel = 10 , acc = 20 )
7 set_velx( 30 ) # The global task velocity is set to 30 (mm/sec).
8 The global task angular velocity is automatically determined.
9 set_accx( 60 ) # The global task acceleration is set to 60 (mm/
sec2). The global task angular acceleration is automatically
determined.
10 movel(P2) # The task motion linear velocity to P2 is 30(mm/sec)
which is the global velocity.
11 movel(P1, vel = 20 , acc = 40 ) # The task motion linear velocity
to P1 is 20(mm/sec) which is the specified velocity.
12 #2
13 set_velx( 10.5 ) # Decimal point input is possible.

```

Related commands

- [set_accx\(acc1,acc2\)\(p. 48\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [movel\(\)\(p. 59\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [movej\(\)\(p. 59\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesx\(\)\(p. 77\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)

3.2.5 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 movei(), amovei(), 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

```

1  P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2  movei(P0)
3  P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
4  P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
5  movel(P1, vel = 10 , acc = 20 )
6  set_velx( 30 , 20 ) # The global task velocity is set to 30(mm/
sec) and 20(deg/sec).
7  set_accx( 60 , 40 ) # The global task acceleration is set to
60(mm/sec2) and 40(deg/sec2).
```

```

8      movel(P2)          # The task motion acceleration to P2 is 60(mm/
sec2) and 40(deg/sec2) which is the global acceleration.
9      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\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [movel\(\)\(p. 59\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [movel\(\)\(p. 59\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesx\(\)\(p. 77\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)

3.2.6 set_accx(acc)

Features

This function sets the linear acceleration of the task space motion globally. The globally set acceleration acc is applied as the default acceleration if the task motion such as movel(), amovel(), movec(), movesx() is called without the explicit input of the acceleration value. The set value acc defines the linear acceleration of the TCP while the rotating acceleration of the TCP is determined proportionally to the linear acceleration.

Parameters

| Parameter Value | 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

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movej(P0)
3 P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
4 P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
5 movej(P0, vel = 10 , acc = 20 )
6 movel(P1, vel = 10 , acc = 20 )
7 set_velx( 30 )      # The global task velocity is set to 30 (mm/
sec). The global task angular velocity is automatically
determined.
8 set_accx( 60 )      # The global task acceleration is set to 60
(mm/sec2). The global task angular acceleration is automatically
determined.
9 movel(P2)          # The task motion linear acceleration to P2 is
60(mm/sec2) which is the global acceleration.
10 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\(vel1,vel2\)¹](#)
- [set_velx\(vel\)²](#)
- [movel\(\)³](#)
- [movec\(\)⁴](#)
- [movesx\(\)⁵](#)
- [moveb\(\)⁶](#)
- [move_spiral\(\)⁷](#)
- [movel\(\)⁸](#)

¹ http://manual.doosanrobotics.com/display/Programming/.set_velx%28vel1%2C+vel2%29+v2.8reworking

² http://manual.doosanrobotics.com/display/Programming/.set_velx%28vel%29+v2.8reworking

³ <http://manual.doosanrobotics.com/display/Programming/.movej%28%29+v2.8reworking>

⁴ <http://manual.doosanrobotics.com/display/Programming/.movec%28%29+v2.8reworking>

⁵ <http://manual.doosanrobotics.com/display/Programming/.movesx%28%29+v2.8reworking>

⁶ <http://manual.doosanrobotics.com/display/Programming/.moveb%28%29+v2.8reworking>

⁷ http://manual.doosanrobotics.com/display/Programming/.move_spiral%28%29+v2.8reworking

⁸ <http://manual.doosanrobotics.com/display/Programming/.movel%28%29+v2.8reworking>

- [movec\(\)](#)⁹
- [movesx\(\)](#)¹⁰
- [amoveb\(\)](#)¹¹
- [amove_spiral\(\)](#)¹²

3.2.7 set_tcp(name)

Features

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

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---------------------------------------|
| name | string | - | Name of the TCP registered in the TP. |

Return

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

Exception

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

⁹ <http://manual.doosanrobotics.com/display/Programming/.movec%28%29+v2.8reworking>

¹⁰ <http://manual.doosanrobotics.com/display/Programming/.movesx%28%29+v2.8reworking>

¹¹ <http://manual.doosanrobotics.com/display/Programming/.amoveb%28%29+v2.8reworking>

¹² http://manual.doosanrobotics.com/display/Programming/.amove_spiral%28%29+v2.8reworking

Example

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movei(P0)
3 set_tcp( "tcp1" ) # The TCP data registered as tcp1 in the TP is
4 called and set to the current TCP value.
5 P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
6 movel(P1, vel = 10 , acc = 20 ) # Moves the recognized center of
7 the tool to the P1 position.

```

Related commands

- [fkin\(pos, ref\)](#)^(p. 36)
- [ikin\(pos, sol_space, ref, ref_pos_opt, iter_threshold\)](#)^(p. 38)
- [movel\(\)](#)¹³
- [movec\(\)](#)¹⁴
- [movesx\(\)](#)¹⁵
- [moveb\(\)](#)¹⁶
- [move_spiral\(\)](#)¹⁷
- [movel\(\)](#)¹⁸
- [movec\(\)](#)¹⁹
- [movesx\(\)](#)²⁰
- [amoveb\(\)](#)²¹
- [amove_spiral\(\)](#)²²

3.2.8 set_ref_coord(coord)

Features

This function sets the reference coordinate system.

13 <http://manual.doosanrobotics.com/display/Programming/.movel%28%29+v2.8reworking>
 14 <http://manual.doosanrobotics.com/display/Programming/.movec%28%29+v2.8reworking>
 15 <http://manual.doosanrobotics.com/display/Programming/.movesx%28%29+v2.8reworking>
 16 <http://manual.doosanrobotics.com/display/Programming/.moveb%28%29+v2.8reworking>
 17 http://manual.doosanrobotics.com/display/Programming/.move_spiral%28%29+v2.8reworking
 18 <http://manual.doosanrobotics.com/display/Programming/.movel%28%29+v2.8reworking>
 19 <http://manual.doosanrobotics.com/display/Programming/.movec%28%29+v2.8reworking>
 20 <http://manual.doosanrobotics.com/display/Programming/.movesx%28%29+v2.8reworking>
 21 <http://manual.doosanrobotics.com/display/Programming/.amoveb%28%29+v2.8reworking>
 22 http://manual.doosanrobotics.com/display/Programming/.amove_spiral%28%29+v2.8reworking

Parameter

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

Return

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

Exception

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

Example

```

1  p0  =  posi( 0 , 0 , 90 , 0 , 90 , 0 )
2  movei(p0, v = 30 , a = 30 )
3  x1  =  posx( 370.9 , 419.7 , 651.5 , 90 , - 180 , 0 )
4  movel(x1, v = 100 , a = 100 ) # Base Coordinate basis

```

```

5   uu1 = [ -1, 1, 0 ] # x-axis vector of the user coordinate
6   system (base coordinate basis)
7   vv1 = [ 1, 1, 0 ] # y-axis vector of the user coordinate
8   system (base coordinate basis)
9   pos = posx( 370.9, -419.7, 651.5, 0, 0, 0 ) # Origin
10  point of the user coordinate system
11  DR_USER1 = set_user_cart_coord(uu1, vv1, pos) # Sets the user
12  coordinate system.
13  set_ref_coord(DR_USER1) # Sets DR_USER1 of the user
14  coordinate system to the global coordinate system.
15  movel([ 0, 0, 0, 0, 0, 0 ], v = 100, a = 100) # The global
16  coordinate system is used if the reference coordinate system is
17  not specified.
18  # Moves to the origin point and
19  direction of the DR_USER1 coordinate system.
20  movel([ 0, 200, 0, 0, 0, 0 ], v = 100, a = 100) # Moves to
21  the (0,200,0) point of the DR_USER1 coordinate system.

```

Related commands

- [movel\(\)](#)²³
- [movejx\(\)](#)(p. 64)
- [movec\(\)](#)²⁴
- [movesx\(\)](#)²⁵
- [moveb\(\)](#)²⁶
- [move_spiral\(\)](#)²⁷
- [move_periodic\(\)](#)(p. 89)

3.3 Synchronous Motion

3.3.1 movej()

Features

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

²³ <http://manual.doosanrobotics.com/display/Programming/.movel%28%29+v2.8reworking>

²⁴ <http://manual.doosanrobotics.com/display/Programming/.movec%28%29+v2.8reworking>

²⁵ <http://manual.doosanrobotics.com/display/Programming/.movesx%28%29+v2.8reworking>

²⁶ <http://manual.doosanrobotics.com/display/Programming/.moveb%28%29+v2.8reworking>

²⁷ http://manual.doosanrobotics.com/display/Programming/.move_spiral%28%29+v2.8reworking

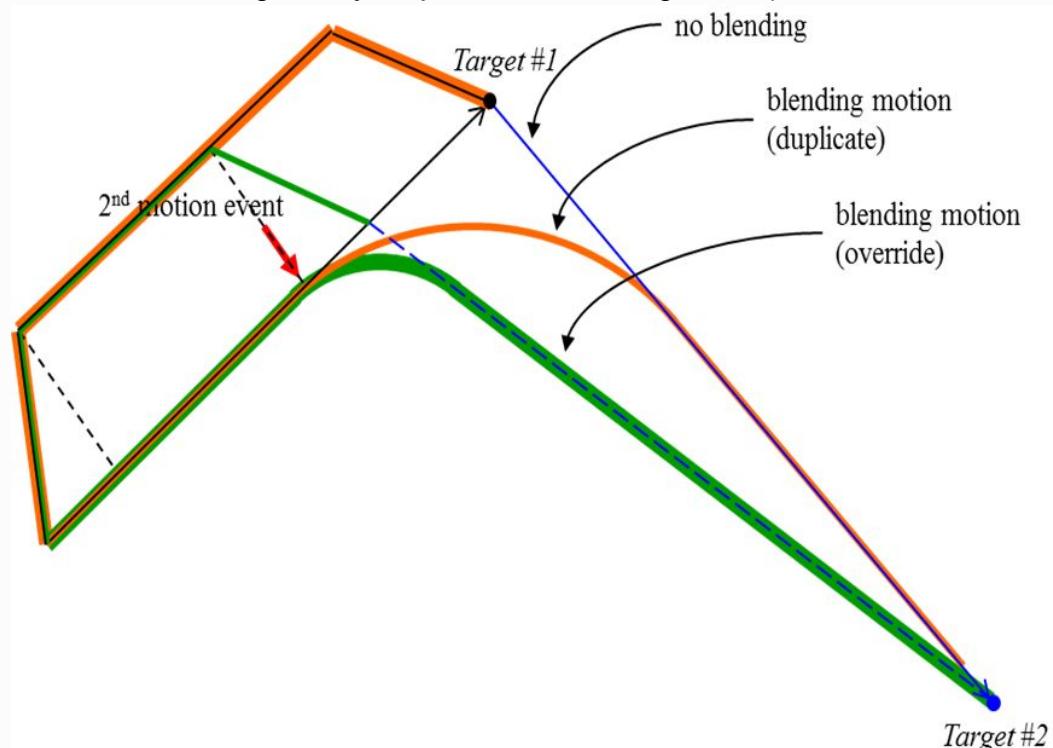
Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|--------------------|---|
| pos | posj | - | posj or joint angle list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity (same to all axes) or velocity (to an axis) |
| | list (float[6]) | None | |
| acc (a) | float | None | acceleration (same to all axes) or acceleration (acceleration to an axis) |
| | list (float[6]) | None | |
| time (t) | float | None | Reach time [sec] |
| radius (r) | float | None | Radius for blending |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative |
| ra | int | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override |

i Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.

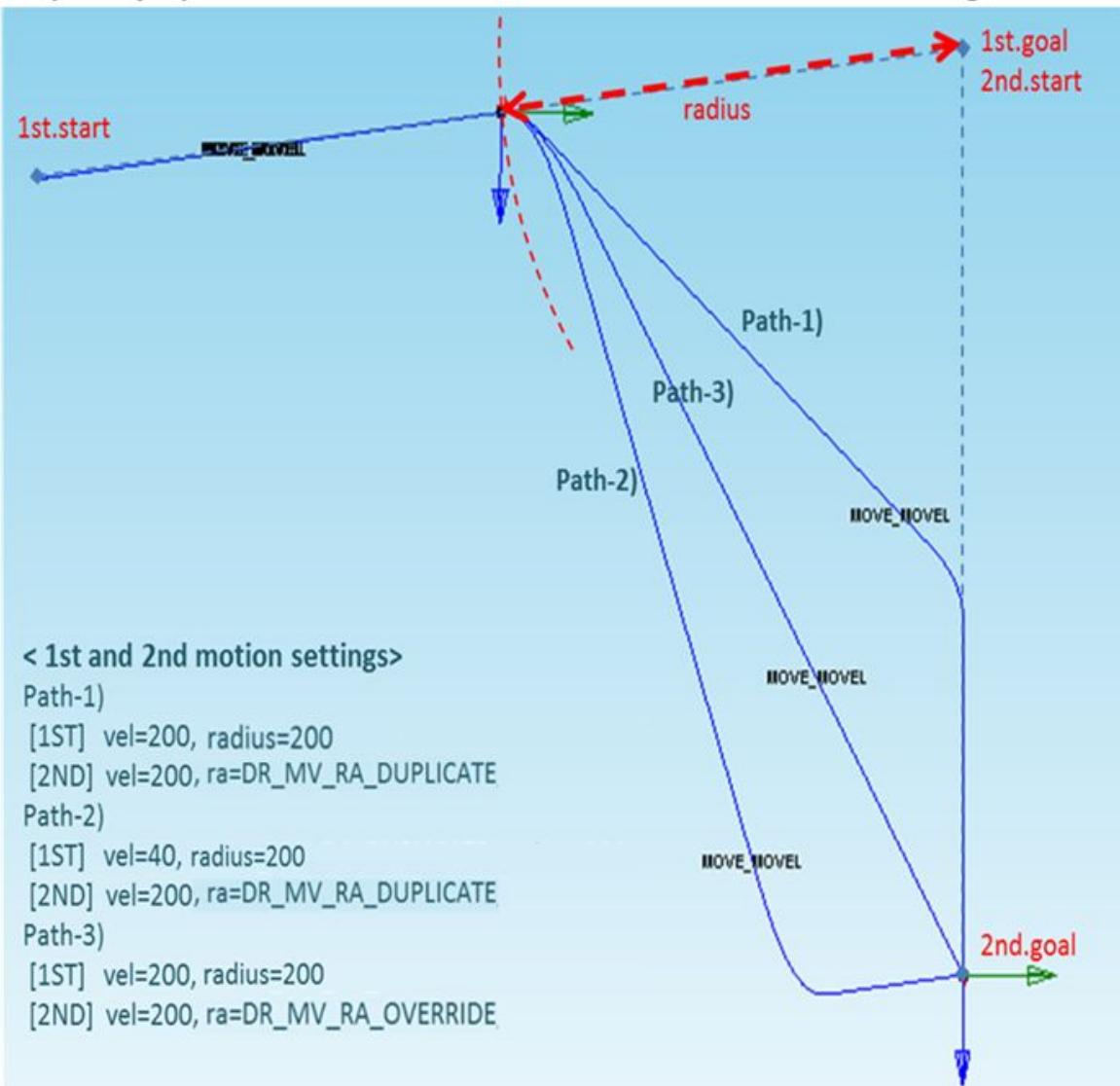
- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of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



⚠ Caution

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

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



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

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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  Q1 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2  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\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [amovej\(\)\(p. 95\)](#)

3.3.2 movel()

Features

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

Parameters

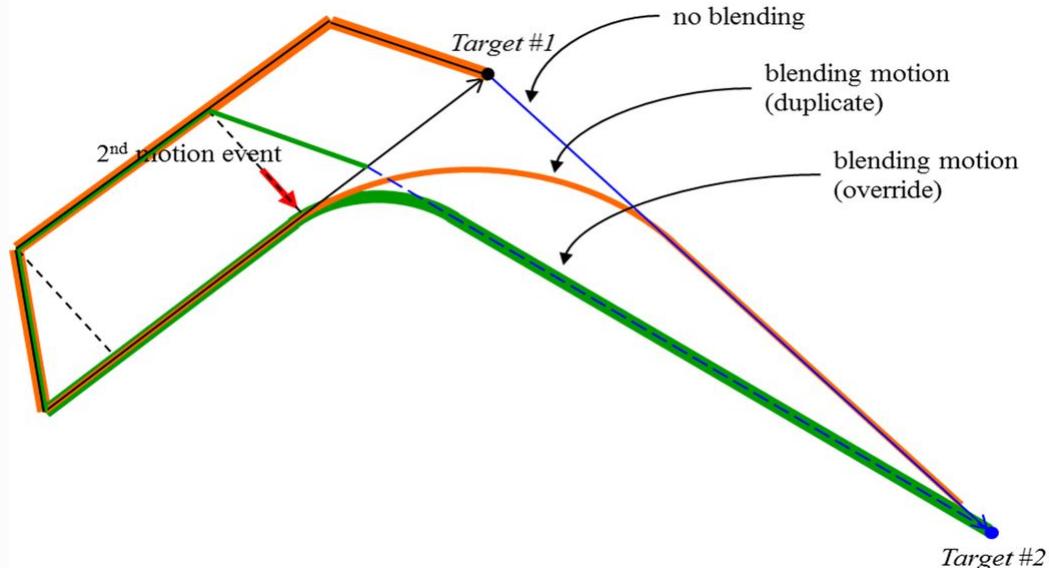
| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|--|
| pos | posx | - | posx or position list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity or velocity1, velocity2 |
| | list (float[2]) | None | |
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | None | |
| time (t) | float | None | Reach time [sec] <ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> DR_BASE: base coordinate DR_WORLD: world coordinate DR_TOOL: tool coordinate user coordinate: user defined |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------------|--|
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative |
| ra | int | DR_MV_RA_DUPLICAT E | Reactive motion mode <ul style="list-style-type: none"> • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override |
| app_type | int | DR_MV_APP_NONE | Application mode <ul style="list-style-type: none"> • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related |

 **Note**

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If ‘app_type’ is ‘DR_MV_APP_WELD’, parameter ‘vel’ is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of ‘vel’.
- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain: ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure

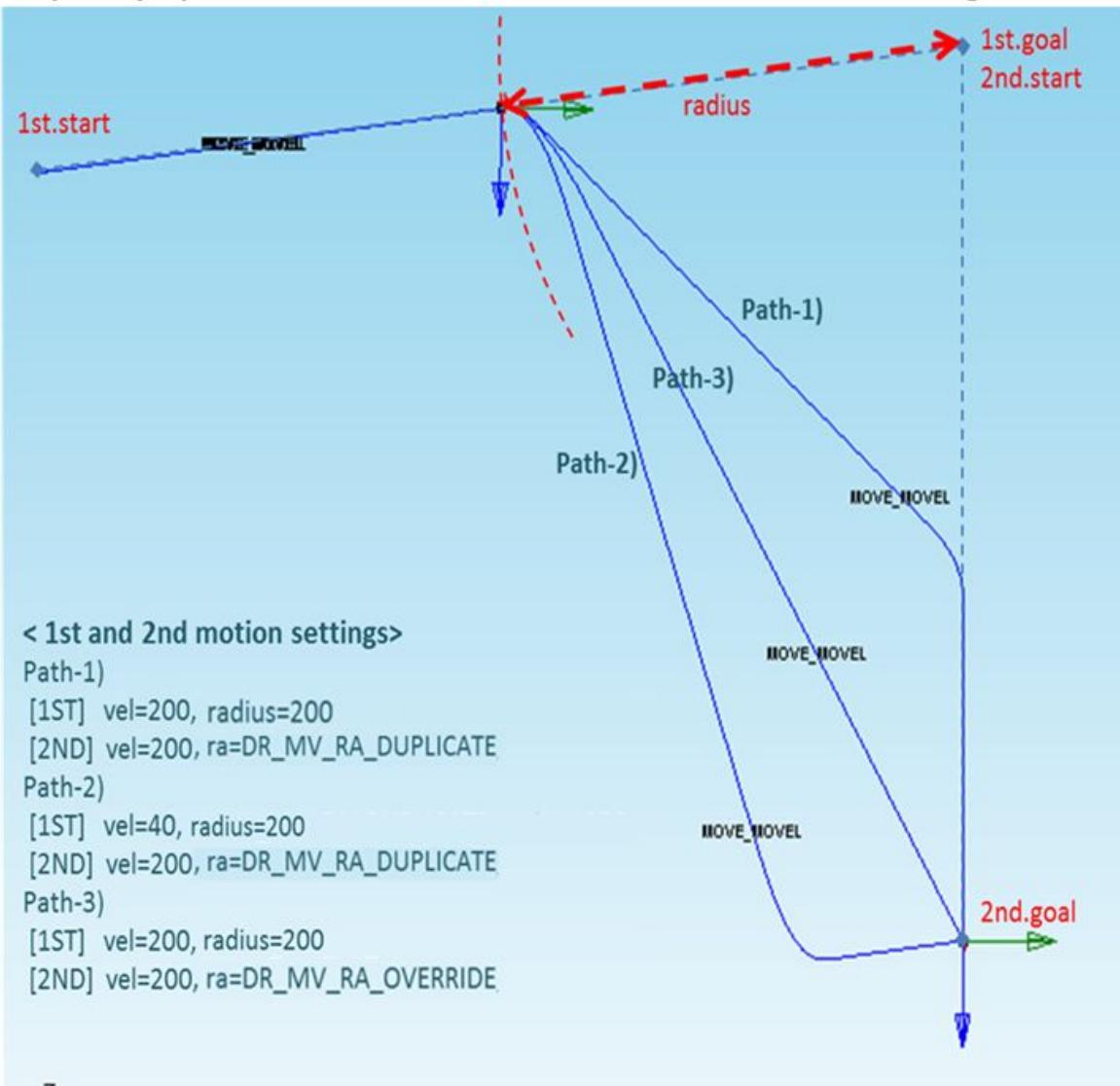
below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option `ra=DR_MV_RA_DUPLICATE` is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option `ra=DR_MV_RA_OVERRIDE` is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



⚠ Caution

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

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



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

[Return](#)

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

| Value | Description |
|----------------|-------------|
| Negative value | Failed |

Exception

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

Example

```

1  P0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
2  movej(P0, v = 30 , a = 30 )
3  P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
4  P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
5  P3 = posx( 30 , 30 , 30 , 0 , 0 , 0 )
6  movel(P1, vel = 30 , acc = 100 )
7      # Moves to the P1 position with a velocity of 30(mm/sec) and
    # acceleration of 100(mm/sec2).
8  movel(P2, time = 5 )
9      # Moves to the P2 position with a reach time of 5 sec.
10 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
11    # with a reach time of 5 sec.
12 movel(P2, time = 5 , r = 10 )
    # Moves the robot to the P2 position with a reach time of 5
    # seconds.
13    # and the next motion is executed when the distance from the
    # P2 position is 10mm.
14
15

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_vely\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accy\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [amovel\(\)\(p. 98\)](#)

3.3.3 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 position list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity (same to all axes) or velocity (to an axis) |
| | list (float[6]) | | |
| acc (a) | float | None | acceleration (same to all axes) or acceleration (acceleration to an axis) |
| | list (float[6]) | | |
| time (t) | float | None | Reach time [sec] |
| radius (r) | float | None | Radius for blending |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|--------------------|---|
| ref | int | None | reference coordinate ž DR_BASE: base coordinate ž DR_WORLD: world coordinate ž DR_TOOL: tool coordinate ž user coordinate: User defined |
| mod | int | DR_MV_MOD_ABS | Movement basis ž DR_MV_MOD_ABS: Absolute ž DR_MV_MOD_REL: Relative |
| ra | int | DR_MV_RA_DUPLICATE | Reactive motion mode ž DR_MV_RA_DUPLICATE: duplicate ž DR_MV_RA_OVERRIDE: override |
| sol | int | 0 | Solution space |

ⓘ Note

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

⚠ Caution

- In versions below SW V2.8, if the blending radius exceeds 1/2 of the total moving distance, the motion is not operated because it affects the motion after blending, and the running task program is terminated when a blending error occurs

- In SW V2.8 or later, if the blending radius exceeds 1/2 of the total moving distance, the blending radius size is automatically changed to 1/2 of the total moving distance, and the change history can be checked in the information log message.

Robot configuration (shape vs. solution space)

| Solution space | Binary | Shoulder | Elbow | Wrist |
|-----------------------|---------------|-----------------|--------------|--------------|
| 0 | 000 | Lefty | Below | No Flip |
| 1 | 001 | Lefty | Below | Flip |
| 2 | 010 | Lefty | Above | No Flip |
| 3 | 011 | Lefty | Above | Flip |
| 4 | 100 | Righty | Below | No Flip |
| 5 | 101 | Righty | Below | Flip |
| 6 | 110 | Righty | Above | No Flip |
| 7 | 111 | Righty | Above | Flip |

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

```

1   P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2
3   movej(P0, v = 30 , a = 30 )
4
5   P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
6
7   P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
8
9   movel(P2, vel = 100 , acc = 200 )           # Linear movement to P2
10
11  X tmp, sol init = get current posx()      # Obtains the current
12    solution space from the P2 position
13
14  movejx(P1, vel = 30 , acc = 60 , sol = sol_init)
15
16    # Moves to the joint angle with a velocity and acceleration of
17    # 30(deg/sec) and 60(deg/sec2), respectively,
18
19  movejx(P2, time = 5 , sol = 2 )
20
21    # Moves to the joint angle with a reach time of 5 sec when the TCP
22    edge is at the P2 position
23
24    # (forcefully sets a solution space to 2)
25
26  moveix(P1, vel = [ 10 , 20 , 30 , 40 , 50 , 60 ] , acc = [ 20 ,
27    20 , 30 , 30 , 40 , 40 ] , radius = 100 , sol = 2 )
28
29    # Moves the robot to the joint angle when the TCP edge is at the
30    # P1 position,
31
32    # and the next motion is executed when the distance from the P2
33    # position is 100mm.
34
35    movejx(P2, v = 30 , a = 60 , ra = DR_MV_RA_OVERRIDE , sol = 2 )
36
37    # Immediately terminates the last motion and blends it to move to
38    # the joint angle
39
40    # when the TCP edge is at the P2 position.

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [get_current_posx\(ref\)\(p. 155\)](#)
- [amovejx\(\)\(p. 101\)](#)

3.3.4 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

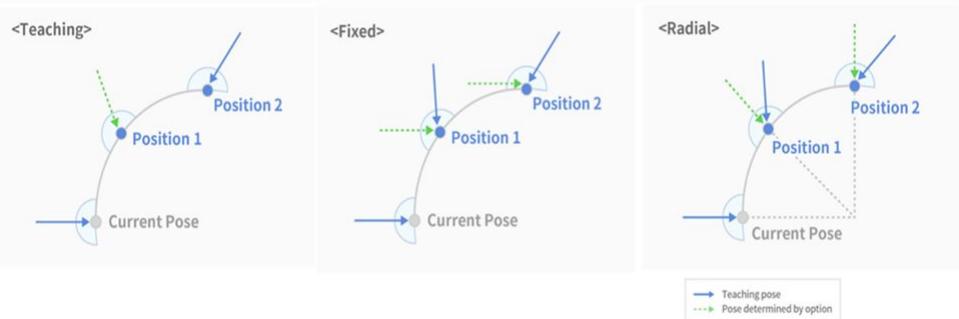
| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|---|
| pos | posj | - | posx or position list |
| | list (float[6]) | | |
| pos2 | posx | | posx or position list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity or velocity1, velocity2 |
| | list (float[2]) | None | |
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | None | |
| time (t) | float | None | Reach time [sec] |
| radius (r) | float | None | Radius for blending |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|--------------------|---|
| ref | int | None | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| angle (an) | float | None | angle or angle1, angle2 |
| | list (float[2]) | | |
| ra | int | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override |
| ori | int | DR_MV_ORI_TEACH | Orientation mode <ul style="list-style-type: none"> • DR_MV_ORI_TEACH: orientation changes continuously from the initial to the final taught value • DR_MV_ORI_FIXED: orientation holds with the initial orientation • DR_MV_ORI_RADIAL: orientation changes radially from the initial. |
| app_type | int | DR_MV_APP_NONE | Application mode <ul style="list-style-type: none"> • DR_MV_APP_NONE: application related • DR_MV_APP_WELD: Welding application related |

(i) Note

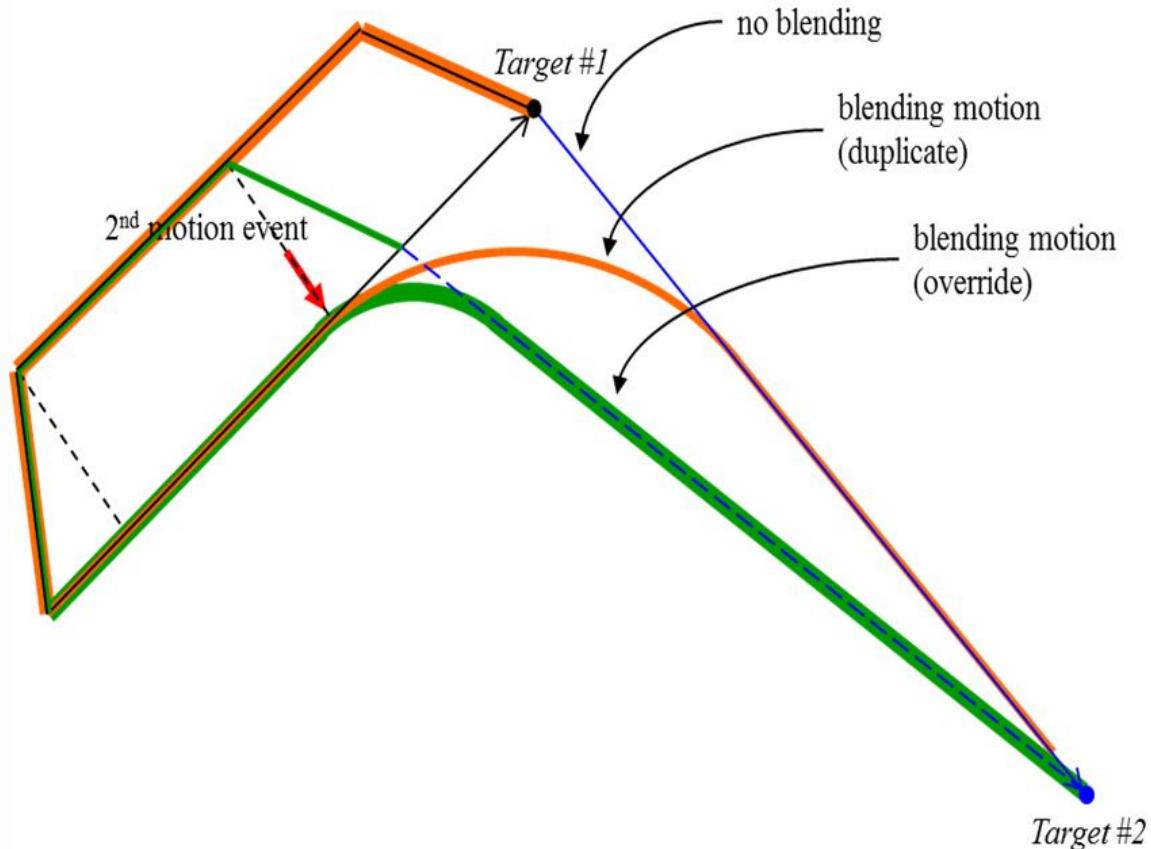
- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius, angle:an)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by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

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



- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain:

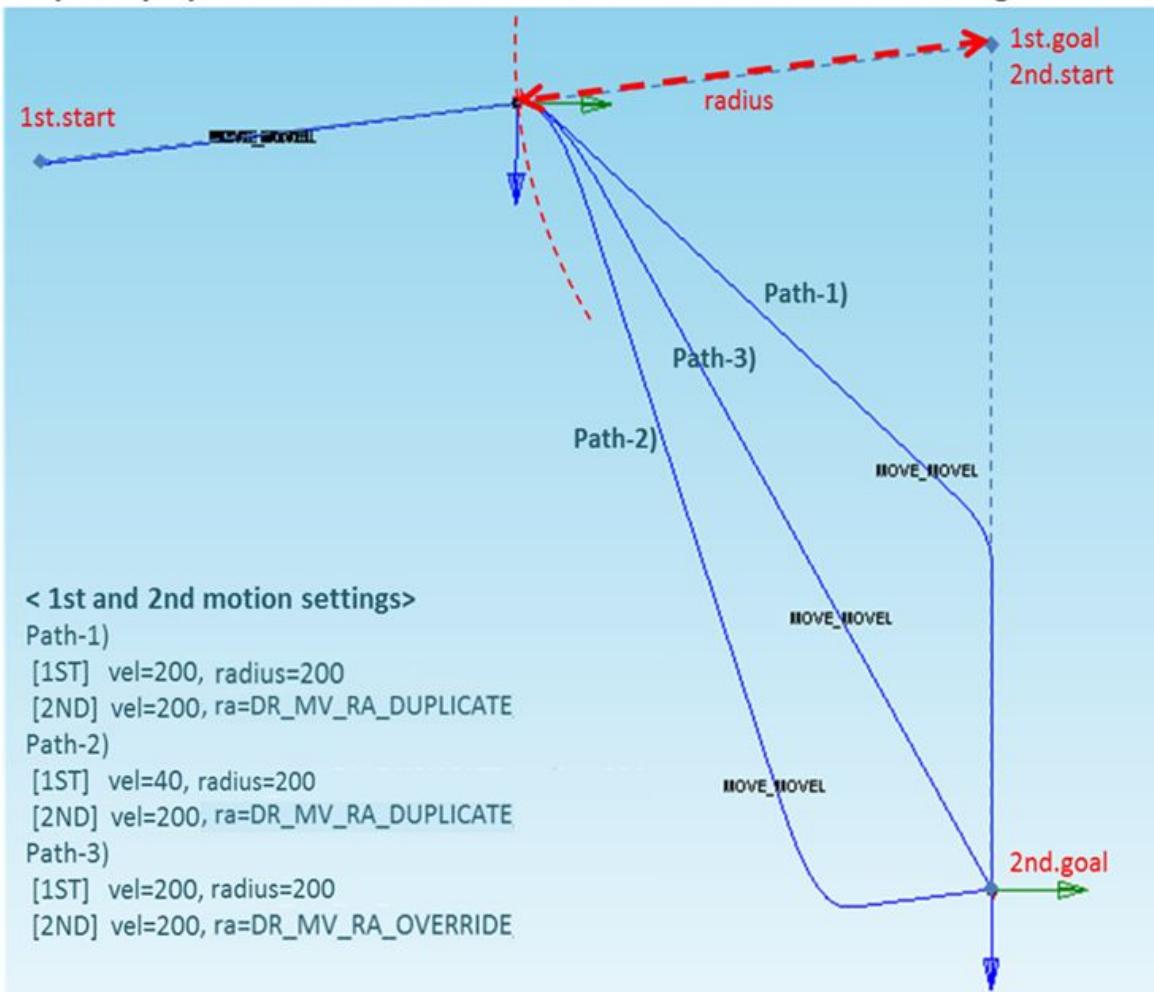
ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option ra=DR_MV_RA_DUPLICATE is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option ra=DR_MV_RA_OVERRIDE is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



⚠ Caution

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

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



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

[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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 #1
2 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
3 movei(P0)
4 set_velx( 30 , 20 ) # Set the global task velocity to 30(mm/sec)
5 and 20(deg/sec).
6 set_accx( 60 , 40 ) # Set the global task acceleration to 60(mm/
sec2) and 40(deg/sec2).
7
8 P1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
9 P2 = posx( 400 , 500 , 500 , 0 , 180 , 0 )
10 P3 = posx( 100 , 300 , 700 , 45 , 0 , 0 )
11 P4 = posx( 500 , 400 , 800 , 45 , 45 , 0 )
12
13 movec(P1, P2, vel = 30 )
14 # Moves to P2 with a velocity of 30(mm/sec) and global
15 acceleration of 60(mm/sec2)
16 # via P1 along the arc trajectory.
17 movei(P0)
18 movec(P3, P4, vel = 30 , acc = 60 )
19 # Moves to P4 with a velocity of 30(mm/sec) and acceleration of
20 60(mm/sec2).
21 # via P3 along the arc trajectory
22 movei(P0).
23 movec(P2, P1, time = 5 )
24 # Moves with a global velocity of 30(mm/sec) and acceleration of
25 60(mm/sec2).
26 # to P1 along the arc trajectory via P2 at the 5-second point.
27 movec(P3, P4, time = 3 , radius = 100 )

```

```

24   # Moves along the arc trajectory to P4 via P3 with a reach time of
25   # 3 seconds
26   # and then executes the next motion at a distance of 100mm from
27   # 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\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [amovec\(\)\(p. 104\)](#)

3.3.5 movesj()

Features

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

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

Parameters

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

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative |

Note

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

Return

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

Exception

| Exception | Description |
|------------------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1   #CASE 1) Absolute angle input (mod= DR_MV_MOD_ABS)
2
3   q0 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
4   movej(q0, vel = 30 , acc = 60 )      # Moves in joint motion to the
initial position (q0).
5   q1 = posi( 10 , - 10 , 20 , - 30 , 10 , 20 ) # Defines the
posi variable (joint angle) q1.
6   q2 = posi( 25 , 0 , 10 , - 50 , 20 , 40 )
7   q3 = posi( 50 , 50 , 50 , 50 , 50 , 50 )
8   q4 = posi( 30 , 10 , 30 , - 20 , 10 , 60 )
9   q5 = posj( 20 , 20 , 40 , 20 , 0 , 90 )
10
11  alist = [q1, q2, q3, q4, q5]      # Defines the list (qlist)
which is a set of q1-q5 as the waypoints.
12
13  movesi(alist, vel = 30 , acc = 100 )
14      # Moves the spline curve that connects the waypoints defined
in the qlist.
15      # with a maximum velocity of 30(mm/sec) and maximum
acceleration of 100(mm/sec2)
16
17  #CASE 2) Relative angle input (mod= DR_MV_MOD_REL)
18  q0 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
19  movei(q0, vel = 30 , acc = 60 )          # Moves in joint motion
to the initial position (q0).
20  dq1 = posi( 10 , - 10 , 20 , - 30 , 10 , 20 )      #
Defines dq1 (q1=q0+dq1) as the relative joint angle of q0
21  dq2 = posi( 15 , 10 , - 10 , - 20 , 10 , 20 )      #
Defines dq2 (q2=q1+dq2) as the relative joint angle of q1
22  dq3 = posi( 25 , 50 , 40 , 100 , 30 , 10 )      #
Defines dq3 (q3=q2+dq3) as the relative joint angle of q2
23  dq4 = posi( - 20 , - 40 , - 20 , - 70 , - 40 , 10 )      #
Defines dq4 (q4=q3+dq4) as the relative joint angle of q3
24  dq5 = posi( - 10 , 10 , 10 , 40 , - 10 , 30 )      #
Defines dq5 (q5=q4+dq5) as the relative joint angle of q4
25
26  dqlist = [dq1, dq2, dq3, dq4, dq5]
27      # Defines the list (dqlist) which is a set of q1-q5 as the
relative waypoints.
28
29  movesi(dqlist, vel = 30 , acc = 100 , mod = DR_MV_MOD_REL )
30      # Moves the spline curve that connects the relative waypoints
defined in the dqlist

```

31

```
# with a maximum velocity of 30(mm/sec) and maximum
acceleration of 100(mm/sec2) (same motion as CASE-1).
```

Related commands

- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [amovesj\(\)\(p. 108\)](#)

3.3.6 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 velocity1, velocity2 |
| | list (float[2]) | | |
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | | |
| time (t) | float | None | Reach time [sec] |
| ref | int | None | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|-----------------|--|
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| vel_opt | int | DR_MVS_VEL_NONE | Velocity option <ul style="list-style-type: none"> • DR_MVS_VEL_NONE: None • DR_MVS_VEL_CONST: Constant velocity |

Note

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

Caution

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

Return

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

Exception

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

Example

```

1 #CASE 1) Absolute coordinate input (mod= DR_MV_MOD_ABS)
2 P0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
3 movei(P0, v = 30 , a = 30 )
4 x0 = posx( 600 , 43 , 500 , 0 , 180 , 0 ) # Defines the
5 posx variable (space coordinate/pose) x0.
6 movel(x0, vel = 100 , acc = 200 ) # Linear movement to the
7 initial position x0
8 x1 = posx( 600 , 600 , 600 , 0 , 175 , 0 ) # Defines the
9 posx variable (space coordinate/pose) x1.
10 x2 = posx( 600 , 750 , 600 , 0 , 175 , 0 )
11 x3 = posx( 150 , 600 , 450 , 0 , 175 , 0 )
12 x4 = posx( - 300 , 300 , 300 , 0 , 175 , 0 )
13 x5 = posx( - 200 , 700 , 500 , 0 , 175 , 0 )
14 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.

```

```

15 movesx(xlist, vel = [ 100 , 30 ], acc = [ 200 , 60 ], vel_opt = DR_MVS_VEL_NONE)
16     # Moves the spline curve that connects the waypoints defined
17     # in the xlist
18     # with a maximum velocity of 100, 30(mm/sec, deg/sec) and
19     # maximum acceleration of 200(mm/sec2) and
20     # 60(deg/sec2).
21 movesx(xlist, vel = [ 100 , 30 ], acc = [ 200 , 60 ], time = 5 ,
22 vel_opt = DR_MVS_VEL_CONST)
23     # Moves the spline curve that connects the waypoints defined
24     # in the xlist
25     # with a constant velocity of 100, 30(mm/sec, deg/sec).
26
27 #CASE 2) Relative coordinate input (mod= DR_MV_MOD_REL)
28 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
29 movej(P0)
30 x0 = posx( 600 , 43 , 500 , 0 , 180 , 0 ) # Defines the
31 posx variable (space coordinate/pose) x0.
32 movel(x0, vel = 100 , acc = 200 ) # Linear movement to the
33 initial position x0
34 dx1 = posx( 0 , 557 , 100 , 0 , - 5 , 0 )
35     # Definition of relative coordinate dx1 to x0 (Homogeneous
36 transformation of dx1 based in x1= x0)
37 dx2 = posx( 0 , 150 , 0 , 0 , 0 , 0 )
38     # Definition of relative coordinate dx2 to x1 (Homogeneous
39 transformation of dx2 based in x2= x1)
40 dx3 = posx( - 450 , - 150 , - 150 , 0 , 0 , 0 )
41     # Definition of relative coordinate dx3 to x2 (Homogeneous
42 transformation of dx3 based in x3= x2)
43 dx4 = posx( - 450 , - 300 , - 150 , 0 , 0 , 0 )
44     # Definition of relative coordinate dx4 to x3 (Homogeneous
45 transformation of dx4 based in x4= x3)
46 dx5 = posx( 100 , 400 , 200 , 0 , 0 , 0 )
47     # Definition of relative coordinate dx5 to x4 (Homogeneous
48 transformation of dx5 based in x5= x4)
49 dx6 = posx( 800 , - 100 , - 100 , 0 , 0 , 0 )
50     # Definition of relative coordinate dx6 to x5 (Homogeneous
51 transformation of dx6 based in x6= x5)
52
53 dxlist = [dx1, dx2, dx3, dx4, dx5, dx6]
54     # Defines the list (dxlist) which is a set of dx1-dx6 as the
55 waypoints.
56
57 movesx(dxlist, vel = [ 100 , 30 ], acc = [ 200 , 60 ], mod =
58 DR_MV_MOD_REL, vel_opt = DR_MVS_VEL_NONE)
59     # Moves the spline curve that connects the waypoints defined
60     # in the dxlist
61     # with a maximum velocity of 100, 30 (mm/sec, deg/sec)
62     # and maximum acceleration of 200(mm/sec2), and 60(deg/sec2)
63     # (same motion as CASE-1).

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [amovesx\(\)\(p. 111\)](#)

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

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|----------------|--|
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| app_type | int | DR_MV_APP_NONE | Application mode <ul style="list-style-type: none"> • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related |

Note

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

Caution

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

Return

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

Exception

| Exception | Description |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| 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 # Init Pose @ Jx1
2 Jx1 = posl( 45 , 0 , 90 , 0 , 90 , 45 )           # initial
3 joint position
X0 = posx( 370 , 420 , 650 , 0 , 180 , 0 )       # initial
task position

```

```

1 # CASE 1) ABSOLUTE
2 # Absolute Goal Poses
3 X1 = posx( 370 , 670 , 650 , 0 , 180 , 0 )
4 X1a = posx( 370 , 670 , 400 , 0 , 180 , 0 )
5 X1a2 = posx( 370 , 545 , 400 , 0 , 180 , 0 )
6 X1b = posx( 370 , 595 , 400 , 0 , 180 , 0 )
7 X1b2 = posx( 370 , 670 , 400 , 0 , 180 , 0 )
8 X1c = posx( 370 , 420 , 150 , 0 , 180 , 0 )
9 X1c2 = posx( 370 , 545 , 150 , 0 , 180 , 0 )
10 X1d = posx( 370 , 670 , 275 , 0 , 180 , 0 )
11 X1d2 = posx( 370 , 795 , 150 , 0 , 180 , 0 )
12
13 seg11 = posb(DR_LINE, X1, radius = 20 )
14 seg12 = posb(DR_CIRCLE, X1a, X1a2, radius = 20 )
15 seg14 = posb(DR_LINE, X1b2, radius = 20 )
16 seg15 = posb(DR_CIRCLE, X1c, X1c2, radius = 20 )

```

```

17 seg16 = posb(DR_CIRCLE, X1d, X1d2, radius = 20 )
18 b_list1 = [seg11, seg12, seg14, seg15, seg16]
19 # The blending radius of the last waypoint (seg16) is ignored.
20
21 movei(Jx1, vel = 30 , acc = 60 , mod = DR_MV_MOD_ABS)
22 # Joint motion to the initial angle (Jx1)
23 movel(X0, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV_MOD_A
BS)
24 # Line motion to the initial position (X0)
25 moveb(b_list1, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV_
MOD_ABS)
26     # Moves the robot from the current position through a
27 trajectory consisting of seg11(LINE), seg12(CIRCLE), seg14(LINE),
28     # seg15(CIRCLE), and seg16(CIRCLE) with a constant velocity
29     # of 150(mm/sec) with the exception of accelerating and decelerating
30 sections.
31     # (The final point is X1d2.) Blending to the next segment
32 begins
33     # when the distance of 20mm from the end point (X1, X1a2,
34 X1b2, X1c2, and X1d2) of each segment
35     # is reached.

```

```

1 # CASE 2) RELATIVE
2 # Relative Goal Poses
3 dX1 = posx( 0 , 250 , 0 , 0 , 0 , 0 )
4 dX1a = posx( 0 , 0 , - 150 , 0 , 0 , 0 )
5 dX1a2 = posx( 0 , - 125 , 0 , 0 , 0 , 0 )
6 dX1b = posx( 0 , 50 , 0 , 0 , 0 , 0 )
7 dX1b2 = posx( 0 , 75 , 0 , 0 , 0 , 0 )
8 dX1c = posx( 0 , - 250 , - 250 , 0 , 0 , 0 )
9 dX1c2 = posx( 0 , 125 , 0 , 0 , 0 , 0 )
10 dX1d = posx( 0 , 125 , 125 , 0 , 0 , 0 )
11 dX1d2 = posx( 0 , 125 , - 125 , 0 , 0 , 0 )
12
13 dseg11 = posb(DR_LINE, dX1, radius = 20 )
14 dseg12 = posb(DR_CIRCLE, dX1a, dX1a2, radius = 20 )
15 dseg14 = posb(DR_LINE, dX1b2, radius = 20 )
16 dseg15 = posb(DR_CIRCLE, dX1c, dX1c2, radius = 20 )
17 dseg16 = posb(DR_CIRCLE, dX1d, dX1d2, radius = 20 )
18 db_list1 = [dseg11, dseg12, dseg14, dseg15, dseg16]
19 # The blending radius of the last waypoint (dseg16) is ignored.
20
21 movei(Jx1, vel = 30 , acc = 60 , mod = DR_MV_MOD_ABS)
22 # Joint motion to the initial angle (Jx1)
23 movel(X0, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV_MOD_A
BS)
24 # Line motion to the initial position (X0)
25 moveb(b_list1, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV_
MOD_ABS)
26     # Moves the robot from the current position through a
27 trajectory consisting of dseg11(LINE), dseg12(CIRCLE),
28 dseg14(LINE),

```

```

27      # dseg15(CIRCLE), and dseg16(CIRCLE) with a constant velocity
28      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\(seg_type, posx1, posx2=None, radius=0\)\(p. 34\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [amoveb\(\)\(p. 114\)](#)

3.3.8 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> |

| Parameter Name | Data Type | Default Value | Range | Description |
|----------------|-----------|---------------|-------|--|
| axis | int | DR_AXIS_Z | - | axis <ul style="list-style-type: none"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |
| ref | Int | DR_TOOL | - | reference coordinate <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined |

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- rev refers to the total number of revolutions of the spiral motion.
- Rmax refers to the maximum radius of the spiral motion.
- Lmax refers to the parallel distance in the axis direction during the motion. A negative value means the parallel distance in the -axis direction.
- Vel refers to the moving velocity of the spiral motion.
- The first value of _global_velx (parallel velocity) is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- acc refers to the moving acceleration of the spiral motion.
- The first value of _global_accx (parallel acceleration) is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- The axis defines the axis that is perpendicular to the surface defined by the spiral motion.
- Ref refers to the reference coordinate system defined by the spiral motion.
- This function does not support online blending of previous and subsequent motions.

Caution

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

Return

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

Exception

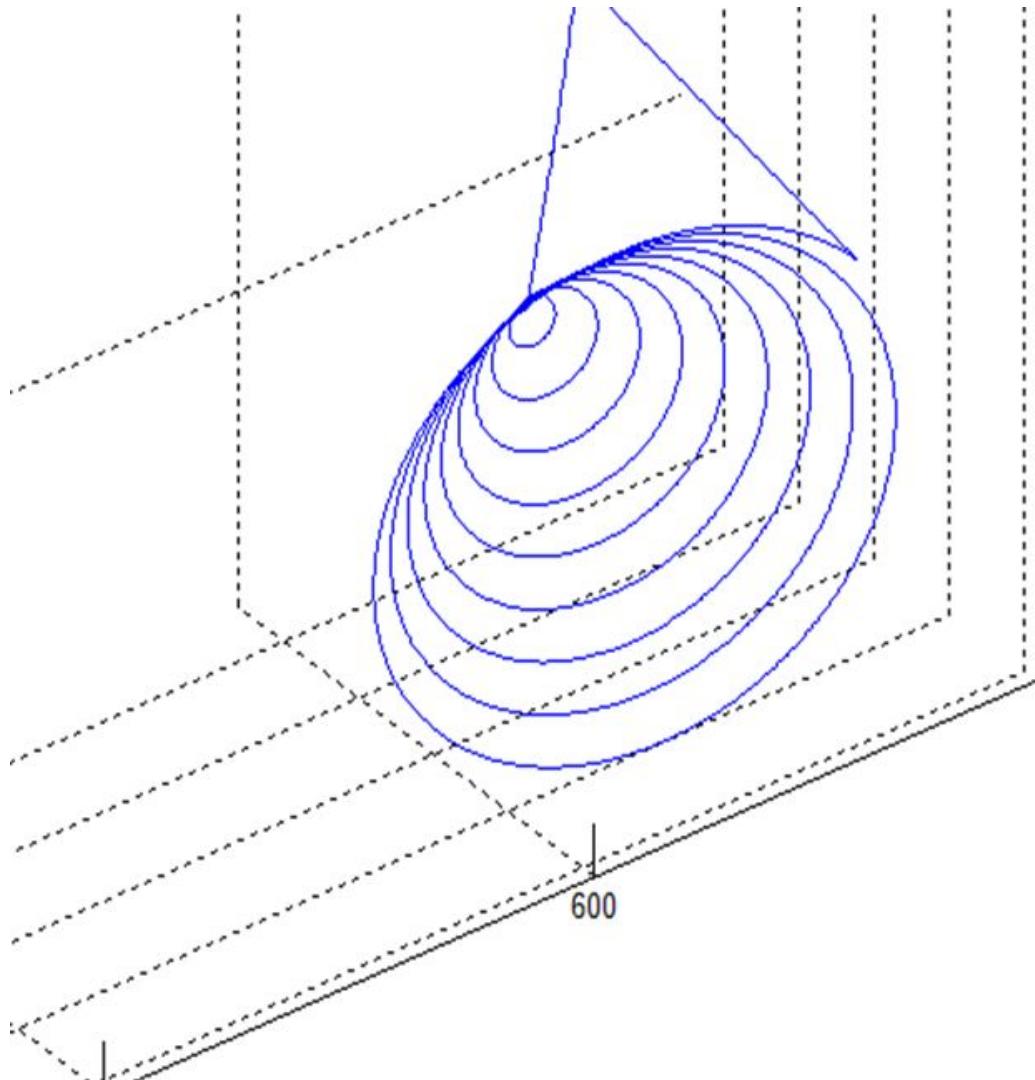
| Exception | Description |
|---------------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |
| DR_Error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

Example

```

1 # hole search
2 # (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)
3
4 J00 = posi( 0 , 0 , 90 , 0 , 60 , 0 )
5 movei(J00,vel = 30 ,acc = 30 )           # Joint movement to the
initial pose
6 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\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [amove_spiral\(\)\(p. 118\)](#)

3.3.9 move_periodic()

Features

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

Parameters

| Parameter Name | Data Type | Default Value | Range | Description |
|----------------|-----------------------------|---------------|----------|--|
| amp | list (float[6]) | - | 0≤amp | Amplitude(motion between -amp and +amp) [mm] or [deg] |
| period | float or list (float[6]) | | 0≤period | period(time for 1 cycle)[sec] |
| atime | float | 0.0 | 0≤atime | Acc-, dec- time [sec] |
| repeat | int | 1 | > 0 | Repetition count |
| ref | int | DR_TOOL | - | reference coordinate <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined |

(i) Note

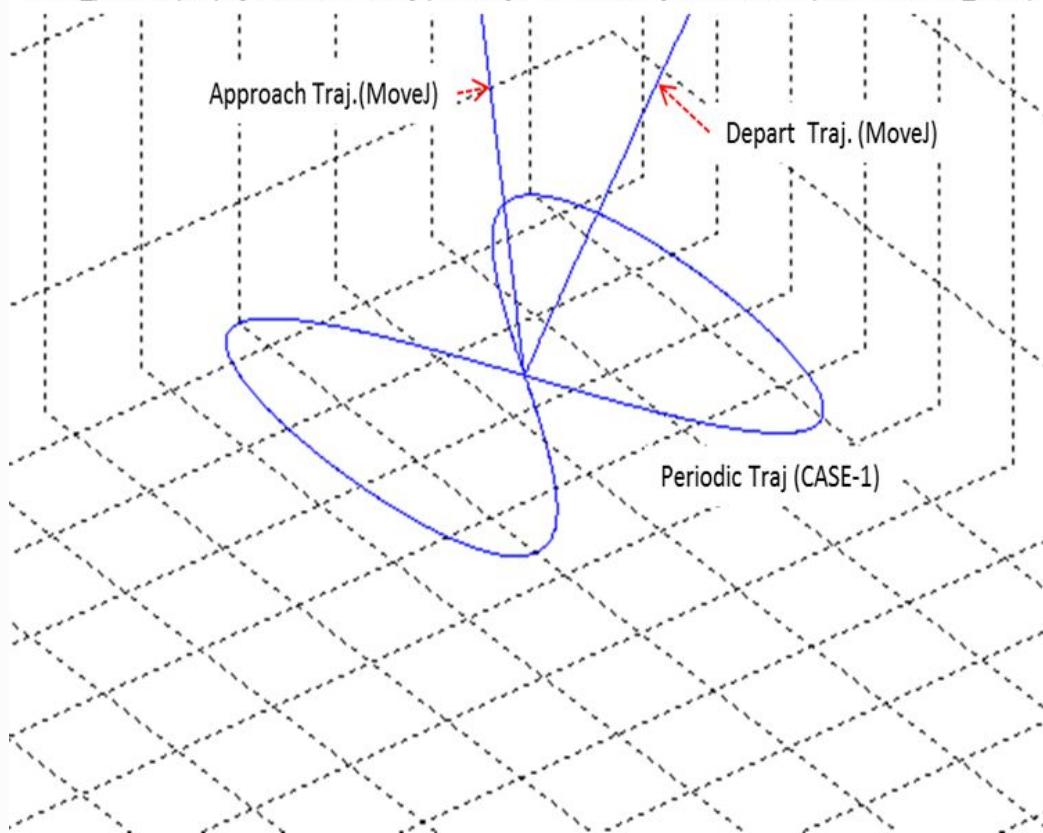
- Amp refers to the amplitude. The input is a list of 6 elements which are the amp values for the axes (x, y, z, rx, ry, and rz). The amp input on the axis that does not have a motion must be 0.
- Period refers to the time needed to complete a motion in the direction, the amplitude. The input is a list 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.

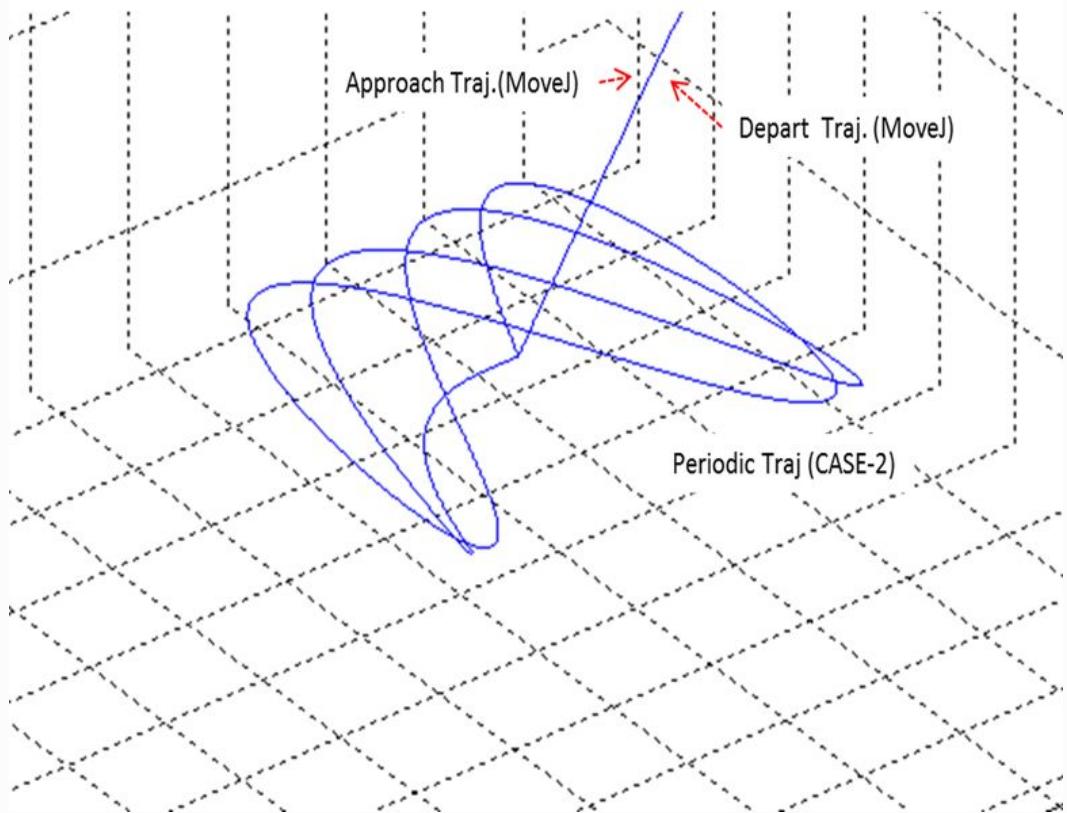
CASE-1) All-axis motions end at the same time

```
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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movej(P0)
3
4 #1
5 move_periodic(amp = [ 10 , 0 , 0 , 0 , 30 , 0 ], period = 1.0 ,
6 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
7     # totally, repeat the motion 5 times.
8
9 #2
10 move_periodic(amp = [ 10 , 0 , 20 , 0 , 0.5 , 0 ], period = [ 1 , 0
11 .5 , 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
12     # 3 times. The rotating y-axis motion is not performed since
    # its period is "0".
13     # 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.
14     # The x-axis motion is repeated 4.5 times.

```

Related commands

- [set_ref_coord\(coord\)\(p. 52\)](#)

- [amove_periodic\(\) \(p. 122\)](#)

3.3.10 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.

Parameter

| Parameter Name | Data Type | Default Value | Range | Description |
|----------------|-----------|---------------|-------|--|
| target | int | - | | <p>Tartget of home position</p> <ul style="list-style-type: none"> • DR_HOME_TARGET_MECHANIC : Mechanical home, joint angle (0,0,0,0,0,0) • DR_HOME_TARGET_USER : user home. |

ⓘ Note

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

Return

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

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 move_home(DR_HOME_TARGET_USER)      # Go to the user home
2
3 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
4 movej(P0)

```

3.4 Asynchronous Motion

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

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- Refer to the description of the movej() motion for the path of blending according to option ra and vel/acc.

Return

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

Exception

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

Example

```

1 #Example 1. The robot moves to q1 and stops the motion 3 seconds
2 #after it begins the motion at q0 and then moves to q99
3 q0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
4 amovei (q0, vel = 10 , acc = 20 )           # Moves to q0 and
5 # performs the next command immediately after
6 wait( 3 )                                # Temporarily suspends the program
7 # execution for 3 seconds (while the motion continues).
8 q1 = posi( 0 , 0 , 0 , 0 , 90 , 0 )
9 amovei (q1, vel = 10 , acc = 20 )
10 # Maintains the q0 motion (DUPLICATE blending if the ra argument
11 # is omitted) and iterates to q1.
12 # Performs the next command immediately after the blending motion.
13 mwait( 0 )                                # Temporarily suspends the program execution
14 until the motion is terminated.
15 q99 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
16 movej (q99, vel = 10 , acc = 20 )          # Joint motion to
q99

```

Related commands

- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [movej\(\)\(p. 54\)](#)

3.4.2 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.

i 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 position list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity or velocity1, velocity2 |
| | list (float[2]) | | |
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | | |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------------|---|
| time (t) | float | None | Reach time [sec] <ul style="list-style-type: none"> If the time is specified, values are processed based on time, ignoring vel and acc. |
| ref | int | None | reference coordinate <ul style="list-style-type: none"> DR_BASE : base coordinate DR_WORLD : world coordinate DR_TOOL : tool coordinate user coordinate: User defined |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> DR_MV_MOD_ABS: Absolute DR_MV_MOD_REL: Relative |
| ra | int | DR_MV_RA_DUPLICAT E | Reactive motion mode <ul style="list-style-type: none"> DR_MV_RA_DUPLICATE: duplicate DR_MV_RA_OVERRIDE: override |
| app_type | int | DR_MV_APP_NONE | Reactive motion mode <ul style="list-style-type: none"> DR_MV_APP_NONE: No application related DR_MV_APP_WELD: Welding application related |

 **Note**

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

- If the time is None, it is set to 0.
- `_g_coord` is applied if the ref is None. (The initial value of `_g_coord` is DR_BASE, and it can be set by the `set_ref_coord` command.)
- Refer to the description of the `movej()` motion for the path of the blending according to option `ra` and `vel/acc`.
- If 'app_type' is 'DR_MV_APP_WELD', parameter 'vel' is internally replaced by the speed setting entered in `app_weld_set_weld_cond()`, not the input value of 'vel'.

Return

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

Exception

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

Example

```

1 #Example 1. D-Out 2 seconds after the motion starts with x1
2 i0 = posi( - 148 , - 33 , - 54 , 180 , 92 , 32 )
3 movei(i0, v = 30 , a = 30 )
4 x1 = posx( 784 , 543 , 570 , 0 , 180 , 0 )
5 amovel (x1, vel = 100 , acc = 200 ) # Performs the next motion
6 immediately after beginning a motion with x1.
7 wait( 2 ) # Temporarily suspends the
program execution for 2 seconds (while the motion continues).
8 set_digital_output( 1 , 1 ) # D-Out (no. 1 channel) ON

```

| | |
|---|--|
| 8 | <code>mwait(0) # Temporarily suspends the program execution until the motion is terminated.</code> |
|---|--|

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [moveL\(\)\(p. 59\)](#)

3.4.3 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.

i Note

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

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|--|
| pos | posx | - | posx or position list |
| | list (float[6]) | | |
| vel (v) | float | None | velocity (same to all axes) or velocity (to an axis) |
| | list (float[6]) | None | |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|--------------------|---|
| acc (a) | float | None | acceleration (same to all axes) or acceleration (acceleration to an axis) |
| | list (float[6]) | None | |
| time (t) | float | None | Reach time [sec] |
| ref | int | None | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD : world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| ra | int | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override |
| sol | int | 0 | Solution space |

(i) 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 velj/acc.

⚠ Caution

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

Return

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

Exception

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

Example

```

1 #Example 1. D-Out 2 seconds after the joint motion starts with x1
2 p0 = posj( - 148 , - 33 , - 54 , 180 , 92 , 32 )
3 movej(p0, v = 30 , a = 30 )
4 x1 = posx( 784 , 443 , 770 , 0 , 180 , 0 )
5 amoveix( x1, vel = 100 , acc = 200 , sol = 1 )      # Performs the
6 next motion immediately after beginning a joint motion with x1.
7 wait( 2 )                                         # Temporarily suspends the
8 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\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [get_current_posx\(ref\)\(p. 155\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [movejx\(\)\(p. 64\)](#)

3.4.4 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

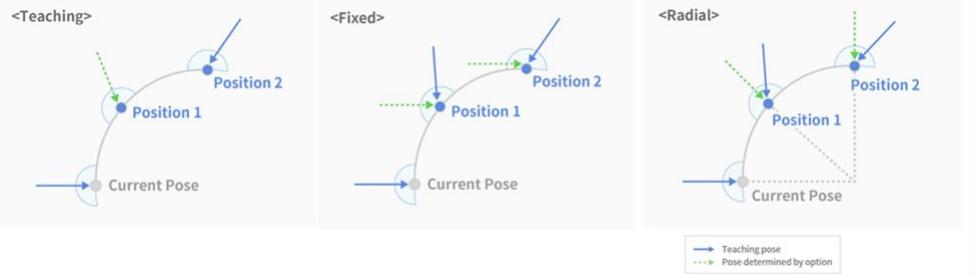
| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|-------------------------------------|
| pos | posx | - | posx or position list |
| | list (float[6]) | | |
| pos2 | posx | - | posx or position list |
| | list (float[6]) | | |
| vel (v)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

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|--------------------|--|
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | | |
| time (t) | float | None | Reach time [sec] |
| ref | int | None | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| angle (an) | float | None | angle or angle1, angle2 |
| | list (float[2]) | | |
| ra | int | DR_MV_RA_DUPLICATE | Reactive motion mode <ul style="list-style-type: none"> • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override |
| ori | int | DR_MV_ORI_TEACH | Orientation mode <ul style="list-style-type: none"> • DR_MV_ORI_TEACH: orientation changes continuously from the initial to the final taught value • DR_MV_ORI_FIXED: orientation holds with the initial orientation • DR_MV_ORI_RADIAL: orientation changes radially from the initial. |
| app_type | int | DR_MV_APP_NONE | Application mode <ul style="list-style-type: none"> • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related |

 **Note**

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

- c. DR_MV_ORI_RADIAL(orientation constrained radially) : Move along the path while maintaining radial orientation at the initial pose to the 'pose 2'



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 #Example 1. D-Out 3 seconds after the arc motion through x1 and x2
2 begins
3 p0 = posi( - 148 , - 33 , - 54 , 180 , 92 , 32 )
4 movej(p0, v = 30 , a = 30 )
5 x1 = posx( 784 , 443 , 770 , 0 , 180 , 0 )

```

```

5   amoveix (x1, vel = 100 , acc = 200 , sol = 2 ) # Performs the
6   next motion immediately after beginning a joint motion with x1.
7   wait( 2 )                                # Temporarily suspends the program
8   execution for 2 seconds (while the motion continues).
    set_digital_output( 1 , 1 )           # D-Out (no. 1 channel) ON
    mwait( 0 )

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [movec\(\)\(p. 68\)](#)

3.4.5 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.

i 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 |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|---|
| vel (v) | float | None | velocity (same to all axes) or velocity (to an axis) |
| | list (float[6]) | | |
| acc (a) | float | None | acceleration (same to all axes) or acceleration (acceleration to an axis) |
| | list (float[6]) | | |
| time (t) | float | None | Reach time [sec] |
| mod | int | DR_MV_MOD_ABS | Movement basis <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |

Note

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

Return

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

Exception

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

Example

```

1 #Example 1. D-Out 3 seconds after the spline motion through q1 - q5 begins
2 q0 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
3 movei(q0, vel = 30 , acc = 60 )      # Moves in joint motion to the
initial position (q0).
4 q1 = posi( 10 , - 10 , 20 , - 30 , 10 , 20 )      #
Defines the posi variable (joint angle) q1.
5 q2 = posi( 25 , 0 , 10 , - 50 , 20 , 40 )
6 q3 = posi( 50 , 50 , 50 , 50 , 50 , 50 )
7 q4 = posi( 30 , 10 , 30 , - 20 , 10 , 60 )
8 q5 = posj( 20 , 20 , 40 , 20 , 0 , 90 )
9
10 qlist = [q1, q2, q3, q4, q5]
11      # Defines the list (qlist) which is a set of waypoints q1-q5.
12
13 amovesi(qlist, vel = 30 , acc = 100 )
14      # Moves the spline curve that connects the waypoints defined
in the qlist.
15      # with a maximum velocity of 30(mm/sec) and maximum
acceleration of 100(mm/sec2).
16      # Executes the next command.
17 wait( 3 )                                # Temporarily suspends
the program execution for 3 seconds (while the motion continues).
18 set digital_output( 1 , 1 )                # D-Out (no. 1
channel) ON
19 mwait( 0 )                                # Temporarily suspends
the program execution until the motion is terminated.

```

Related commands

- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [amovesj\(p. 108\)](#)

3.4.6 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.

(i) 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 velocity1, velocity2 |
| | list (float[2]) | | |
| acc (a) | float | None | acceleration or acceleration1, acceleration2 |
| | list (float[2]) | | |
| time (t) | float | None | Reach time [sec] |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|-----------------|---|
| ref | int | None | <p>reference coordinate</p> <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: User defined |
| mod | int | DR_MV_MOD_ABS | <p>Movement basis</p> <ul style="list-style-type: none"> • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative |
| vel_opt | int | DR_MVS_VEL_NONE | <p>Velocity option</p> <ul style="list-style-type: none"> • DR_MVS_VEL_NONE: None • DR_MVS_VEL_CONST: Constant velocity |

Note

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

Caution

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

Return

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

Exception

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

Example

```

1 #Example 1. D-Out 3 seconds after the spline motion through x1 -
2 x6 begins
3 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
4 movei(P0)
5 x0 = posx( 600 , 43 , 500 , 0 , 180 , 0 )           #
6 Defines the posx variable (space coordinate/pose) x0.
7 movel(x0, vel = 100 , acc = 200 )          # Linear movement to the
8 initial position x0
9 x1 = posx( 600 , 600 , 600 , 0 , 175 , 0 )          #
10 Defines the posx variable (space coordinate/pose) x1.
11 x2 = posx( 600 , 750 , 600 , 0 , 175 , 0 )
12 x3 = posx( 150 , 600 , 450 , 0 , 175 , 0 )
13 x4 = posx( - 300 , 300 , 300 , 0 , 175 , 0 )
14 x5 = posx( - 200 , 700 , 500 , 0 , 175 , 0 )

```

```

11   x6 = posx( 600 , 600 , 400 , 0 , 175 , 0 )
12
13   xlist = [x1, x2, x3, x5, x6]           # Defines the list
(xlist) which is a set of x1-x6 as the waypoints.
14
15   amovesx(xlist, vel = [ 100 , 30 ], acc = [ 200 , 60 ], vel_opt = D
R_MVS_VEL_NONE)
16           # Moves the spline curve that connects the waypoints defined
in the xlist
17           # with a maximum velocity of 100, 30(mm/sec, deg/sec) and
maximum acceleration of 200(mm/sec2) and
18           # 60(deg/sec2). The next command is executed immediately
after the motion starts.
19   wait( 3 )                         # Temporarily suspends the program
execution for 3 seconds (while the motion continues).
20   set_digital_output( 1 , 1 )        # D-Out (no. 1 channel)
ON
21   mwait( 0 )                       # Temporarily suspends the
program execution until the motion is terminated.

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [movesx\(\)\(p. 77\)](#)

3.4.7 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.

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

Note

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

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

Caution

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

Return

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

Exception

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

Example

```

1 #Example 1. D-Out 3 seconds after the motion through the path of
2 seg11 - seg16 begins
3 # Init Pose @ Jx1
4 Jx1 = posj( 45 , 0 , 90 , 0 , 90 , 45 ) # initial
5 joint position
6 X0 = posx( 370 , 420 , 650 , 0 , 180 , 0 ) # initial
7 task position
8
9 # CASE#1) ABSOLUTE
10 # Absolute Goal Poses
11 X1 = posx( 370 , 670 , 650 , 0 , 180 , 0 )
12 X1a = posx( 370 , 670 , 400 , 0 , 180 , 0 )
13 X1a2 = posx( 370 , 545 , 400 , 0 , 180 , 0 )
14 X1b = posx( 370 , 595 , 400 , 0 , 180 , 0 )
15 X1b2 = posx( 370 , 670 , 400 , 0 , 180 , 0 )
16 X1c = posx( 370 , 420 , 150 , 0 , 180 , 0 )
17 X1c2 = posx( 370 , 545 , 150 , 0 , 180 , 0 )
18 X1d = posx( 370 , 670 , 275 , 0 , 180 , 0 )
19 X1d2 = posx( 370 , 795 , 150 , 0 , 180 , 0 )
20
21 seg11 = posb(DR LINE, X1, radius = 20 )
22 seg12 = posb(DR CIRCLE, X1a, X1a2, radius = 20 )
23 seg14 = posb(DR LINE, X1b2, radius = 20 )
24 seg15 = posb(DR CIRCLE, X1c, X1c2, radius = 20 )
25 seg16 = posb(DR CIRCLE, X1d, X1d2, radius = 20 )
26 b_list1 = [seg11, seg12, seg14, seg15, seg16]
27     # The blending radius of the last waypoint (seg16) is
28     # ignored.
29 movej(Jx1, vel = 30 , acc = 60 , mod = DR_MV_MOD_ABS)

```

```

26      # Joint motion to the initial angle (Jx1)
27      movel(X0, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV_MOD_A
28      BS)
29          # Line motion to the initial position (X0)
30          amoveb(b_list1, vel = 150 , acc = 250 , ref = DR_BASE, mod = DR_MV
31          _MOD_ABS)
32              # Moves the robot from the current position through a
33              trajectory consisting of seg11(LINE), seg12(CIRCLE), seg14(LINE),
34              # seg15(CIRCLE), and seg16(CIRCLE) with a constant velocity
35              of 150(mm/sec) with the exception of accelerating and decelerating
36              sections.
37                  # (The final point is X1d2.)
38                  # Blending to the next segment begins when the distance of
39                  20mm from the end point (X1, X1a2, X1b2, X1c2, and X1d2)
40                  # of each segment is reached.
41      wait( 3 )                                # Temporarily suspends the
42      program execution for 3 seconds (while the motion continues).
43      set_digital_output( 1 , 1 )                # D-Out (no. 1 channel)
44      ON
45      mwait( 0 )                                # Temporarily suspends the
46      program execution until the motion is terminated.

```

Related commands

- [posb\(seg_type, posx1, posx2=None, radius=0\)\(p. 34\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [moveb\(\)\(p. 81\)](#)

3.4.8 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

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

(i) 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_RUNTIM E) | C extension module error occurred |

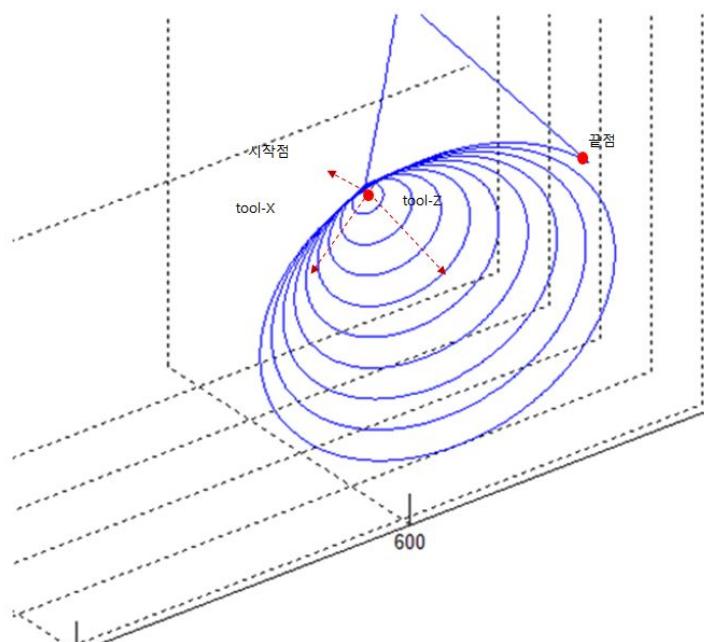
| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  ## hole search
2  # (A motion that completes 9.5 revolutions (rev) to the 30 mm
3  radius (rmax) from 0 on the Tool-X/Y surface as the center of the
4  rotation in the Tool-Z direction
5  # and the spiral trajectory that moves 50 mm (lmax) in the Tool-Z
6  # direction at the same time in 20 seconds
7  # from the initial position.
8  # D-Out (no. 1 channel) 3 seconds after the motion begins.)
9
10 J00 = posi( 0 , 0 , 90 , 0 , 60 , 0 )
11 movel(J00, vel = 30 , acc = 30 )           # Joint moves to the
12 begining pose
13 amove_spiral(rev = 9.5 ,rmax = 50.0 ,lmax = 50.0 ,time = 10.0
14 .axis = DR_AXIS_Z,ref = DR_TOOL)
15 wait( 3 )
16 set digital output( 1 , 1 )                 # D-Out (no. 1 channel) ON
17 mwait( 0 )                                  # Waits until the motion
18 stops.

```



Related commands

- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [set_tcp\(name\)\(p. 51\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [move_spiral\(\)\(p. 85\)](#)

3.4.9 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: Starting from the current position, reaching the end of the periodic trajectory, stopping, and then executing the following command
- amove_periodic: Executes the next command immediately regardless of whether the end of the periodic trajectory is reached from the current position

Parameters

| Parameter Name | Data Type | Default Value | Range | Description |
|----------------|-----------------|---------------|-------|--|
| amp | list (float[6]) | - | 0≤amp | Amplitude (motion between -amp and +amp) [mm] or [deg] |

| Parameter Name | Data Type | Default Value | Range | Description |
|----------------|-----------------------------|---------------|----------|---|
| period | float or list (float[6]) | | 0≤period | Period (time for 1 cycle) [sec] |
| atime | float | 0.0 | 0≤atime | Acc-, dec- time [sec] |
| repeat | int | 1 | > 0 | Repetition count |
| ref | int | DR_TOOL | - | reference coordinate <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined |

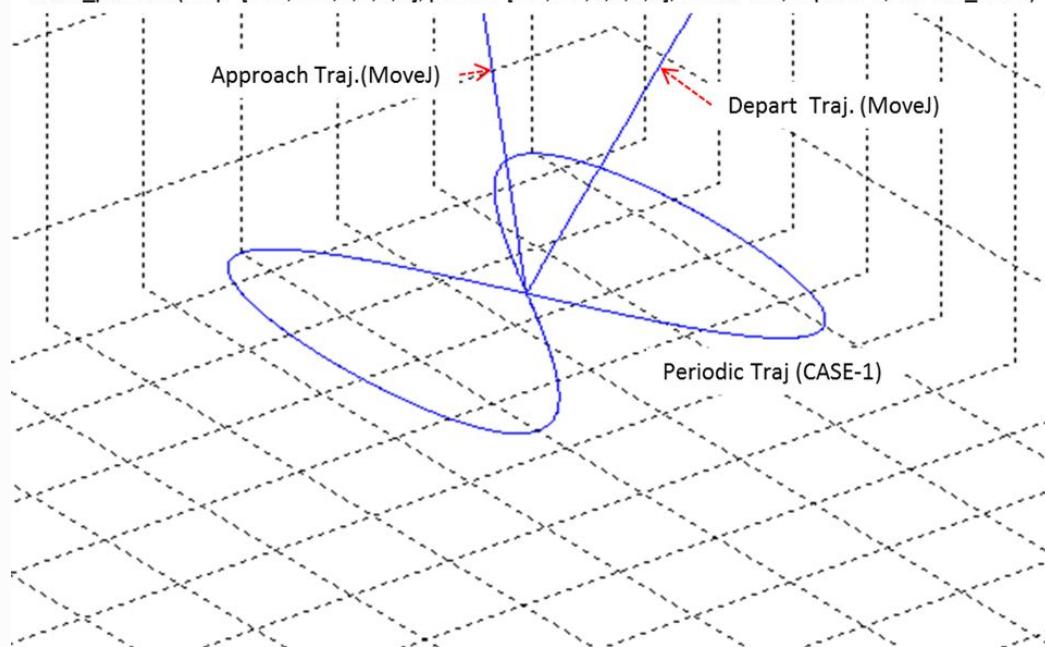
(i) Note

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

are not terminated at the same time. Refer to the following image for more information.

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

`move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.6,0,0,0,0], atime=3.1, repeat=2, ref=DR_BASE)`



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

Return

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

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movei(P0)
3 amove_periodic(amp = [ 10 . 0 . 0 . 0 . 0.5 , 0 ], period = 1 ,
4 atime = 0.5 , repeat = 5 , ref = DR_TOOL)
5 wait( 1 )
6 set_digital_output( 1 , 1 )
7 mwait( 0 )
# Repeats the x-axis (10mm amp and 1 sec. period) motion and v
# 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\(coord\)\(p. 52\)](#)
- [move_periodic\(\)\(p. 89\)](#)

3.5 Additional Functions

3.5.1 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 #Example 1. The robot moves to q1 and stops the motion 3 seconds
2 after it begins the motion at q0 and then moves to q99
3 q0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
4 amovej (q0, vel = 10 , acc = 20 )           # Moves to q0 and
5 performs the next command immediately after
6 wait( 3 )                                # Temporarily suspends the
                                             program execution for 3 seconds (while the motion continues).
7 q1 = posj( 0 , 0 , 0 , 0 , 90 , 0 )
8 amovej (q1, vel = 10 , acc = 20 )

```

```

7      # Maintains the q0 motion (DUPLICATE blending if the ra
8      argument is omitted) and iterates to q1.
9      # Performs the next command immediately after the blending
10     motion.
11     mwait( 0 )                                # Temporarily suspends the
12     program execution until the motion is terminated.
13     q99 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
14     movej( q99, vel = 10 , acc = 20 )        # Joint motion to q99.

```

Related commands

- [wait\(time\)\(p. 265\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovel\(\)\(p. 98\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovec\(\)\(p. 104\)](#)
- [amovesj\(\)\(p. 108\)](#)
- [amovesx\(\)\(p. 111\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)
- [amove_periodic\(\)\(p. 122\)](#)

3.5.2 begin_blend(radius=0)

Features

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

Parameter

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

Return

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

Exception

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

Example

```

1  #1
2  begin_blend(radius = 30 )
3      # The motion commands with the following radius option sets
4      # the blending section to 30mm.
5  Q1 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
6  Q2 = posi( 0 , 0 , 0 , 0 , 90 , 0 )
7  movej(Q1, vel = 10 , acc = 20 )
8      # Moves to the Q1 joint angle and is set to execute the next
motion
9      # when the global distance from the Q1 space position is
30mm.
10 movej(Q2, time = 5 )
11     # Moves to the Q2 joint angle after the blend while
maintaining the last motion (motion iteration).
12     # It is set to execute the next motion
13     # when the global distance from the Q2 space position is
30mm.
14 movej(Q1, v = 30 , a = 60 , r = 200 )
15     # Moves to the Q1 joint angle after the blending while
maintaining the last motion (motion iteration).
16     # It is set to execute the next motion
17     # when the distance from the Q1 space position is 200mm (the
global setting is not applied).
18 movej(Q2, v = 30 , a = 60 , ra = DR_MV_RA_OVERRIDE)
19     # Immediately terminates the last motion and blends it to
move to the Q2 joint angle.
20
21 end_blend()      # Ends the batch setting of the blending
sections.
```

Related commands

- [end_blend\(\)](#)(p. 129)
- [movej\(\)](#)(p. 54)
- [movel\(\)](#)(p. 59)
- [movejx\(\)](#)(p. 64)
- [movec\(\)](#)(p. 68)

3.5.3 end_blend()

Features

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

Return

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

Example

```

1      #1
2      begin_blend(radius = 30 )
3          # The motion commands that have the following radius option
4          # set the blending section to 30mm.
5          Q1 = pos( 0 , 0 , 90 , 0 , 90 , 0 )
6          Q2 = pos( 0 , 0 , 0 , 0 , 90 , 0 )
7          movej(Q1, vel = 10 , acc = 20 )
8              # Moves to the Q1 joint angle and is set to execute the next
9              # motion
10             # when the global distance from the Q1 space position is
11             # 30mm.
12             movej(Q2, time = 5 )
13                 # Immediately terminates the last motion and blends it to
14                 # move to the Q2 joint angle (motion iteration).
15                 # It is set to execute the next motion
16                 # when the global distance from the Q2 space position is
17                 # 30mm.
18                 movej(Q1, v = 30 , a = 60 , r = 200 )
19                     # 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\(radius=0\)\(p. 127\)](#)
- [movej\(\)\(p. 54\)](#)
- [movel\(\)\(p. 59\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movec\(\)\(p. 68\)](#)

3.5.4 check_motion()

Features

This function checks the status of the currently active motion.

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 #1. The next motion (q99) is executed when an asynchronous motion
2 to q0 begins decelerating
3 q0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
4 q99 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
amovei (q0, vel = 10 , acc = 20 )           # Executes the next
command immediately after the motion to q0.

```

```

5   while True :
6     if check_motion() == 0 :           # A motion is completed.
7       amovei( q99, vel = 10 , acc = 20 )    # Joint motion to q99.
8       break
9     if check_motion() == 2 :           # In motion
10    pass
11    mwait( 0 )                      # Temporarily suspends the
                                         program execution until the motion is terminated.

```

Related commands

- [movej\(\)](#)(p. 54)
- [movel\(\)](#)(p. 59)
- [movejx\(\)](#)(p. 64)
- [movec\(\)](#)(p. 68)
- [movesj\(\)](#)(p. 74)
- [movesx\(\)](#)(p. 77)
- [moveb\(\)](#)(p. 81)
- [move_spiral\(\)](#)(p. 85)
- [move_periodic\(\)](#)(p. 89)
- [amovej\(\)](#)(p. 95)
- [amovel\(\)](#)(p. 98)
- [amovejx\(\)](#)(p. 101)
- [amovec\(\)](#)(p. 104)
- [amovesj\(\)](#)(p. 108)
- [amovesx\(\)](#)(p. 111)
- [amoveb\(\)](#)(p. 114)
- [move_spiral\(\)](#)(p. 118)
- [move_periodic\(\)](#)(p. 122)

3.5.5 stop(st_mode)

Features

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

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

Parameter

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

Return

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

Exception

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

Example

```

1 #1. The motion is terminated with a soft stop 2 seconds after
2 moving to x1
3 p0 = posi( - 148 , - 33 , - 54 , 180 , 92 , 32 )
movej(p0, v = 30 , a = 30 )

```

```

4  x1 = posx( 784 , 543 , 570 , 0 , 180 , 0 )
5  amovel(x1, vel = 100 , acc = 200 )      # Executes the next command
immediately after the motion with x1.
6  wait( 2 )                                # Temporarily suspends the
program for 2 seconds.
7  stop(DR_SSTOP)                          # Stops the motion with a soft
stop.

```

Related commands

- [movej\(\)](#)(p. 54)
- [movel\(\)](#)(p. 59)
- [movejx\(\)](#)(p. 64)
- [movec\(\)](#)(p. 68)
- [movesj\(\)](#)(p. 74)
- [movesx\(\)](#)(p. 77)
- [moveb\(\)](#)(p. 81)
- [move_spiral\(\)](#)(p. 85)
- [move_periodic\(\)](#)(p. 89)
- [amovej\(\)](#)(p. 95)
- [amovel\(\)](#)(p. 98)
- [amovejx\(\)](#)(p. 101)
- [amovec\(\)](#)(p. 104)
- [amovesj\(\)](#)(p. 108)
- [amovesx\(\)](#)(p. 111)
- [amoveb\(\)](#)(p. 114)
- [amove_spiral\(\)](#)(p. 118)
- [amove_periodic\(\)](#)(p. 122)

3.5.6 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.

Parameter

| 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  change_operation_speed( 10 )
2  change_operation_speed( 100 )
3  #1. Motion with velocity specified to q0 and 20% of the specified
   velocity
4  q0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
5  movei (q0, vel = 10 , acc = 20 )           # Moves to q0 at a
   velocity of 10mm/sec
6  change_operation_velocity( 10 )            # The velocities of all
   following motions executed are 10% of the specified velocity.
7  q1 = posi( 0 , 0 , 0 , 0 , 90 , 0 )
8  movei (q1, vel = 10 , acc = 20 )           # Moves to q1 at a
   velocity of 10% of 10mm/sec.
9  change_operation_speed( 100 )             # The velocities of all
   following motions executed are 100% of the specified velocity.
10 movei (q0, vel = 10 , acc = 20 )          # Moves to q0 at a
    velocity 100% of 10mm/sec

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [movel\(\)\(p. 59\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesj\(\)\(p. 74\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [move_periodic\(\)\(p. 89\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovel\(\)\(p. 98\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovec\(\)\(p. 104\)](#)
- [amovesj\(\)\(p. 108\)](#)
- [amovesx\(\)\(p. 111\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)
- [amove_periodic\(\)\(p. 122\)](#)

3.5.7 wait_manual_guide()

Features

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

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

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

Return

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

Exception

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

Example

```

1 # Sets up the collaborative workspace before executing the
2 # program.
3 # Surface 1: Point-1(1000,1000), Point-2(0,0)
4 # Surface 2: Point-1(1000,-1000), Point-2(0,0)
5 # Activation of domain 1 - Point(1000,0)
6
7 i00 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
8 movej(i00,vel = 20 ,acc = 40 ) # Enters the collaborative
9 workspace.
10 wait manual guide() # Direct teaching until the "Finish"
11 button on the popup window generated by the TP is pressed.
12 pos1 = get_current_posx() # Stores the directly taught point in
pos1.
13 dposa = posx( 0 , 0 , - 100 , 0 , 0 , 0 )
14 movel(dposa, vel = 300 , acc = 600 , ref = DR_TOOL)
# Retract 100 mm in the tool-Z direction from the taught
position.

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [movel\(\)\(p. 59\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movec\(\)\(p. 68\)](#)
- [movesj\(\)\(p. 74\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [move_periodic\(\)\(p. 89\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovel\(\)\(p. 98\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovec\(\)\(p. 104\)](#)
- [amovesj\(\)\(p. 108\)](#)
- [amovesx\(\)\(p. 111\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)
- [amove_periodic\(\)\(p. 122\)](#)

3.5.8 wait_nudge()

Features

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

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

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

Return

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

| Value | Description |
|----------------|-------------|
| Negative value | Error |

Exception

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

Example

```

1 # Sets up the collaborative workspace before executing the
2   program.
3   # Surface 1: Point-1(1000,1000), Point-2(0,0)
4   # Surface 2: Point-1(1000,-1000), Point-2(0,0)
5   # Activation of domain 1 - Point(1000,0)
6
7   i00 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
8   movei(i00,vel = 20 ,acc = 40 ) # Enters the collaborative
9     workspace.
10    wait_nudge()           # Wait for applying external force
11      exceeding the threshold to the robot
12      dposa = posx( 0 , 0 , - 100 , 0 , 0 , 0 )
13      movel(dposa, vel = 300 , acc = 600 , ref = DR_TOOL)
14          # Retract 100 mm in the tool-Z direction from the taught
15          position

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [movel\(\)\(p. 59\)](#)
- [movejx\(\)\(p. 64\)](#)
- [movec\(\)\(p. 68\)](#)

- [movesj\(\)\(p. 74\)](#)
- [movesx\(\)\(p. 77\)](#)
- [moveb\(\)\(p. 81\)](#)
- [move_spiral\(\)\(p. 85\)](#)
- [move_periodic\(\)\(p. 89\)](#)
- [amovej\(\)\(p. 95\)](#)
- [amovel\(\)\(p. 98\)](#)
- [amovejx\(\)\(p. 101\)](#)
- [amovec\(\)\(p. 104\)](#)
- [amovesj\(\)\(p. 108\)](#)
- [amovesx\(\)\(p. 111\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [amove_spiral\(\)\(p. 118\)](#)
- [amove_periodic\(\)\(p. 122\)](#)

3.5.9 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($n * 100\text{msec}$) 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 amout and increment amount can be set through input paramet 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"> • DR_DPOS : accumulation amount • DR_DVEL : increment amount |

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| ref | int | None | <p>reference coordinate</p> <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined |
| limit_dPOS | list(float[2]) | None | <p>First value : limitation of position[mm] Second value : limitation of orientation[deg]</p> |
| limit_dPOS_per | list(float[2]) | None | <p>Fist value : limitation of position[mm] Second value : limitation of orientation[deg]</p> |

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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  def alter_thread():
2      alter_motion(dx) #dx : amount of alter motion

```

```

3   dX  =  [ 10 , 0 , 0 , 10 , 0 , 0 ]
4
5   J00  =  posi( 0 , 0 , 90 , 0 , 90 )
6   X1  =  posx( 559.0 , 200 , 651.5 , 180 , -180.0 , 180 )
7   X2  =  posx( 559.0 , 200 , 400 , 180 , -180.0 , 180 )
8
9
10  movej(J00,vel = 50 ,acc = 100 )
11
12  enable_alter_motion(n = 10 ,mode = DR_DPOS, ref = DR_BASE,
13    limit_dPOS = [ 50 , 90 ], limit_dPOS_per = [ 50 , 50 ])
# cycle time:(5*100)msec, mode:accumulate, reference
# coordination:base coordination
# Limitation of accumulation amount :50mm,50deg
# Limitation of increment amount :10mm, 10deg
14
15
16
17  th_id  =  thread_run(alter_thread, loop = True )
18
19  movel(X1,v = 50 ,a = 100 ,r = 30 )
20  movel(X2,v = 50 ,a = 100 )
21
22  thread_stop(th_id)
23  disable_alter_motion()  # deactivates alter motion

```

Related commands

- [alter_motion\(pos\)](#)(p. 141)
- [disable_alter_motion\(\)](#)(p. 143)

3.5.10 alter_motion(pos)

Features

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

⚠ Caution

- `alter_motion()` can be executed only in user thread.

ⓘ 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]) | - | 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

```

1  def alter_thread():
2      alter_motion(dX) #dX : amount of alter motion
3
4  dX = [ 10 , 0 , 0 , 10 , 0 , 0 ]
5
6  J00 = posi( 0 , 0 , 90 , 0 , 90 )
7  X1 = posx( 559.0 , 200 , 651.5 , 180 , - 180.0 , 180 )
8  X2 = posx( 559.0 , 200 , 400 , 180 , - 180.0 , 180 )
9
10 movej(J00,vel = 50 ,acc = 100 )
11
12 enable_alter_motion(n = 5 ,mode = DR_DPOS, ref = DR_BASE,
13 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
14
15
16
17 th_id = thread_run(alter_thread, loop = True )

```

```

18    movel(X1,v = 50 ,a = 100 ,r = 30 )
19    movel(X2,v = 50 ,a = 100 )
20
21    thread_stop(th_id)
22    disable.Alter_motion() # deactivates alter motion
23

```

Related commands

- [enable.Alter_motion\(n,mode,ref,limit_dPOS,limit_dPOS_per\)](#)(p. 139)
- [disable.Alter_motion\(\)](#)(p. 143)

3.5.11 disable.Alter_motion()

Features

This function deactivates alter motion.

Exception

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

Example

```

1  def alter_thread():
2      alter_motion(dX) #dX : amount of alter motion
3
4      dX = [ 10 , 0 , 0 , 10 , 0 , 0 ]
5
6      J00 = posi( 0 , 0 , 90 , 0 , 90 )
7      X1 = posx( 559.0 , 200 , 651.5 , 180 , - 180.0 , 180 )
8      X2 = posx( 559.0 , 200 , 400 , 180 , - 180.0 , 180 )

```

```

9      movej(J00,vel = 50 ,acc = 100 )
10
11
12      enable_alter_motion(n = 10 ,mode = DR_DPOS, ref = DR_BASE,
13      limit_dPOS = [ 50 , 90 ], limit_dPOS_per = [ 50 , 50 ])
# cycle time:(5*100)msec, mode:accumulate, reference
# coordination:base coordination
14      # Limitation of accumulation amount :50mm,50deg
15      # Limitation of increment amount :10mm, 10deg
16
17      th_id = thread_run(alter_thread, loop = True )
18
19      movel(X1,v = 50 ,a = 100 ,r = 30 )
20      movel(X2,v = 50 ,a = 100 )
21
22      thread_stop(th_id)
23      disable_alter_motion() # deactivates alter motion

```

Related commands

- [enable_alter_motion\(n,mode,ref,limit_dPOS,limit_dPOS_per\)\(p. 139\)](#)
- [alter_motion\(pos\)\(p. 141\)](#)

3.6 Servo Motion

3.6.1 servoj()

Features

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

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|--|
| pos | posj | - | posj or joint angle list |
| | list (float[6]) | | |
| vel (v) | float | None | maximum velocity (same to all axes) or maximum velocity (to an axis) [deg/ s] |
| | list (float[6]) | | |

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

Note

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

Caution

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

Return

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

Exception

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

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

Example

```

1  set_velj( 30 )
2  set_accj( 60 )
3
4  Xt = posi( 0 , 0 , 0 , 0 , 0 , 0 )
5  servoj(Xt)
6
7  Xt = posi( 0 , 0 , 0 , 0 , 0 , 0 )
8  target = posx( 90 , 0 , 120 , -50 , 50 , -90 )
9  del_t = [ 3 , 0.1 , 3 , -3 , 3 , -3 ]
10
11 while 1 :
12     for i in range( 0 , 6 ):
13         Xt[i] = Xt[i] + del_t[i]
14
15         if del_t[i] > 0 :
16             if Xt[i] > target[i]:
17                 Xt[i] = target[i]
18             else :
19                 if Xt[i] < target[i]:
20                     Xt[i] = target[i]
21
22     servoj(Xt)

```

Related commands

- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [mwait\(time=0\)\(p. 125\)](#)

3.6.2 servol()

Features

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

Parameters

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

Note

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

Caution

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

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  set veli( 30 )
2  set acci( 60 )
3  set velx( 50 )
4  set accx( 100 )
5
6  movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ))
7
8  Xt = posx( 368 , 34.5 , 442.5 , 50.26 , -180 , 50.26 )
9  servol(Xt,v = [ 100 , 100 ], a = [ 200 , 300 ])
10
11 Xt = posx( 368 , 34.5 , 442.5 , 50.26 , -180 , 50.26 )
12 target = posx( 368 , 34.5 , 200 , 50.26 , -180 , 50.26 )
13 del_t = [ 0 , 0 , -3 , 0 , 3 , 0 ]
14
15 while 1 :
16     for i in range ( 0 , 6 ):
17         Xt[i] = Xt[i] + del_t[i]
18
19         if del_t[i] > 0 :
20             if Xt[i] > target[i]:
21                 Xt[i] = target[i]
22             else :
23                 if Xt[i] < target[i]:
```

```

24      Xt[i] = target[i]
25      servol(Xt,v = [ 100 , 100 ], a = [ 200 , 300 ])

```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel1, vel2\)\(p. 45\)](#)
- [set_accx\(acc1, acc2\)\(p. 48\)](#)
- [mwait\(time=0\)\(p. 125\)](#)

3.6.3 speedj()

Features

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

Parameters

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

(i) Note

- Abbreviated parameter names are supported. (a:acc, t:time)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum and acceleration, the reach time is adjusted automatically and notice through information message.
- If you want to stop normally during movement, input vel as [0,0,0,0,0,0] or use the stop command.
- For safety, if a new speedj command is not transmitted for 0.1 [sec] during movement, an error message is displayed and it stops.

⚠ Caution

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

[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 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ), v = 30 , a = 60 )
2
3 v1 = 30
4 go_plus = True
5 while True :
6     q = get_desired_posj()
7     if go_plus:
8         speedi([v1, 5 , 5 , 5 , 5 , 5 ], a = 60 )
9         if q[0] > 90 :
10            go_plus = False
11     else :
12         speedi([- v1, - 5 , - 5 , - 5 , - 5 , - 5 ], a = 60 )
13         if q[0] < - 90 :
14            go_plus = True

```

Related commands

- [posj\(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0\)\(p. 29\)](#)
- [set_velj\(vel\)\(p. 42\)](#)
- [set_accj\(acc\)\(p. 43\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [stop\(st_mode\)\(p. 131\)](#)

3.6.4 speedl()

Features

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

Parameters

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

(i) Note

- Abbreviated parameter names are supported. (a:acc, t:time)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum and acceleration, the reach time is adjusted automatically and notice through information message.
- If you want to stop normally during movement, input vel as [0,0,0,0,0,0] or use the stop command.
- For safety, if a new speedj command is not transmitted for 0.1 [sec] during movement, an error message is displayed and it stops.

⚠ Caution

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

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 movej(posi( 0 , 0 , 90 , 0 , 90 , 0 ), v = 30 , a = 60 )
2 x = get_desired_posx()
3
4 v_y = 250
5 v = 5
6 at_max = 500
7 ar_max = 60
8 go_plus = True
9 while True :
10     xd = get_desired_posx()
11     if go_plus:
12         speedl([v, v, v, 30, 0, 30], [at_max, ar_max])
13         if xd[1] > x[1] + 200 :
14             go_plus = False
15     else :
```

```
16      speedl([-v, -v_y, -v, -30, -0, -30],  
17          [at_max, ar_max])  
18      if xd[1] < x[1] - 200:  
           go_plus = True
```

Related commands

- [posx\(X=0, Y=0, Z=0, A=0, B=0, C=0\)\(p. 30\)](#)
- [set_velx\(vel\)\(p. 46\)](#)
- [set_accx\(acc\)\(p. 49\)](#)
- [mwait\(time=0\)\(p. 125\)](#)
- [stop\(st_mode\)\(p. 131\)](#)

4 Auxiliary Control Commands

4.1 Robot Current Value

4.1.1 get_current_posj()

Features

This function returns the current joint angle.

Return

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

Exception

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

Example

```
1 q1 = get_current_posj()
```

Related commands

- [get_desired_posj\(\)](#)(p. 162)

4.1.2 get_current_velj()

Features

This function returns the current joint velocity.

Return

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

Exception

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

Example

```
1 velj1 = get_current_velj()
```

Related commands

- [get_desired_velj\(\)](#)(p. 163)

4.1.3 get_current_posx(ref)

Features

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

Parameters

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

i 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

```

1  x1, sol  =  get_current_posx()  #x1 w.r.t. DR_BASE
2
3  x1_wld, sol  =  get_current_posx(ref = DR_WORLD)  #x1 w.r.t.
4    DR_WORLD

```

```

5   DR_USR1 = set_user_cart_coord(x1, x2, x3, pos)
6   set_ref_coord(DR_USR1)
7
8   x1, sol = get_current_posx(DR_USR1) #x1 w.r.t. DR_USR1

```

Related commands

- [get_desired_posx\(ref\)](#)(p. 164)

4.1.4 get_current_tool_flange_posx(ref)

Features

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

Parameters

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

```

1   x1 = get_current_tool_flange_posx()
2   #x1 : Flange pose base on the base coordinate(default value)
3   x2 = get_current_tool_flange_posx(DR_BASE)
4   #x2 : Flange pose based on the base coordinate

```

| | |
|---|---|
| 5 | x3 = get_current_tool_flange_posx(DR_WORLD) |
| 6 | #x3 : Flange pose based on the world coordinate |

4.1.5 get_current_velx(ref)

Features

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

Parameters

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

Return

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

Exception

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

Example

| | |
|---|--|
| 1 | velx1 = get_current_velx() |
| 2 | # velx1 : velocity based on the base coordinate(default value) |
| 3 | velx2 = get_current_velx(DR_BASE) |
| 4 | # velx2 (=velx1) : velocity based on the base coordinate |
| 5 | velx3 = get_current_velx(DR_WORLD) |
| 6 | #velx3 : velocity based on the world coordinate |

Related commands

- [get_desired_velx\(ref\)\(p. 165\)](#)

4.1.6 get_current_rotm(ref)

Features

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

Parameters

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

Return

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

Exception

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

Example

```

1  rotm1 = get_current_rotm(DR_WORLD)
2  #rotm1 : rotation matrix(3x3) based on the world coordinate

```

The result value is stored in a 3x3 matrix

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

4.1.7 get_joint_torque()

Features

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

Return

| Value | Description |
|----------|------------------|
| float[6] | JTS torque value |

Exception

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

Example

```
1 j_trq1 = get_joint_torque()
```

Related commands

- [get_external_torque\(\)](#)(p. 160)
- [get_tool_force\(ref\)](#)(p. 161)

4.1.8 get_external_torque()

Features

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

Return

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

Exception

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

Example

```
1 trq_ext = get_external_torque()
```

Related commands

- [get_joint_torque\(\)](#)(p. 160)
- [get_tool_force\(ref\)](#)(p. 161)

4.1.9 get_tool_force(ref)

Features

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

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

Parameters

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

Return

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

Exception

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

Example

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

Related commands

- [get_joint_torque\(\)](#)(p. 160)
- [get_external_torque\(\)](#)(p. 160)

4.2 Robot Target Value

4.2.1 get_desired_posj()

Features

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

Return

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

Exception

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

Example

```
1   jp1 = get_desired_posj()
```

Related commands

- [get_current_posj\(\)\(p. 154\)](#)

4.2.2 get_desired_velj()

Features

This function returns the current target joint velocity. It cannot be used in the moveJ, moveC, moveSx, moveB, moveSpiral, or move_Periodic command.

Return

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

Exception

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

Example

```
1   velj1 = get_desired_velj()
```

Related commands

- [get_current_velj\(\)\(p. 154\)](#)

4.2.3 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 <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: User defined |

ⓘ 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

```

1  x1 = get_desired_posx() #x1 w.r.t. DR BASE
2  x2 = posx( 100 , 0 , 0 , 0 , 0 , 0 )
3  x3 = posx( 0 , 0 , 20 , 20 , 20 , 20 )
4  pos = x3
5  DR_USR1 = set_user_cart_coord(x1, x2, x3, pos)
6  set_ref_coord(DR_USR1)
7
8  xa = get_desired_posx(DR_USR1) #xa w.r.t. DR_USR1

```

| | |
|---|--|
| 9 | <code>xb = get_desired_posx(DR_WORLD) #xb w.r.t. DR_WORLD</code> |
|---|--|

Related commands

- [get_desired_posx\(ref\)](#)(p. 164)

4.2.4 get_desired_velx(ref)

Features

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

Parameters

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

Return

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

Exception

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

Example

| | |
|---|---|
| 1 | <code>vel_x1 = get_desired_velx()</code> |
| 2 | <code>#vel_x1 : desired velocity of the tool based on the base</code> |
| | <code>coordinate(default value)</code> |
| 3 | <code>vel_x2 = get_desired_velx(DR_BASE)</code> |

```

4 #vel_x2 : desired velocity of the tool based on the base
5 coordinate
6 vel_x3 = get_desired_velx(DR_WORLD)
#vel_x3 : desired velocity of the tool based on the world

```

Related commands

- [get_current_velx\(ref\)\(p.158\)](#)

4.3 Control State Value

4.3.1 get_control_mode()

Features

This function returns the current control mode.

Return

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

Exception

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

Example

```

1 mode = get_control_mode()

```

4.3.2 get_control_space()

Features

This function returns the current control space.

Return

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

Exception

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

Example

```
1 x1 = get_control_space()
```

4.3.3 get_current_solution_space()**Features**

This function returns the current solution space value.

Return

| Value | Description |
|--------------|------------------------|
| int | Solution space (0 ~ 7) |

Exception

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

Example

```
1 sol = get_current_solution_space()
```

Related commands

- [get_solution_space\(pos\)\(p. 168\)](#)

4.3.4 get_solution_space(pos)

Features

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

Parameters

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

Return

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

Exception

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

Example

```

1  q1 = posi( 0 , 0 , 0 , 0 , 0 , 0 )
2  sol1 = get solution space(q1)
3  sol2 = get_solution_space([ 10 , 20 , 30 , 40 , 50 , 60 ])

```

Related commands

- [get_current_solution_space\(\)](#)(p.167)

4.3.5 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"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |

Return

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

Exception

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

Example

```

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

```

Related commands

- [get_current_rotm\(ref\)\(p.159\)](#)

5 Other Settings Command

5.1 Tool/Workpiece Settings

5.1.1 get_workpiece_weight()

Features

This function measures and returns the weight of the workpiece.

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

```
1   weight = get_workpiece_weight()
```

Related commands

- [reset_workpiece_weight\(\)](#)(p. 172)

5.1.2 reset_workpiece_weight()

Features

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

Return

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

Exception

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

Example

```
1    reset_workpiece_weight()
```

Related commands

- [get_workpiece_weight\(\)](#)(p. 171)

5.1.3 set_workpiece_weight(weight=0.0, cog=[0.0,0.0,0.0], cog_ref=DR_CUR_TCP, add_up=DR_REPLACE, start_time=None, transition_time=None)

Features

In addition to the tool weight/center of gravity at the end of the robot, set the weight/center of gravity of the work piece and other information. The weight and center of gravity of the entire payload is reflected by combining the set tool weight/center of gravity and the work piece's weight/center of gravity. It can be used in applications where the type of workpiece is frequently varied or the weight needs to be dynamically changed.

⚠ Caution

- Workpiece weight change is allowed only when both Collision Detection and TCP SLF Violation check are mute or deactivated during Auto Mode.
- In the current version, Collision Detection considers the function mute when Collision Sensitivity is overridden to 0 and TCP SLF considers the function mute when the TCP SLF Limit is overridden to the maximum. This override can be set using Collision Sensitivity Reduction Zone and Custom Zone.
- Otherwise, trigger an SS1 protective stop unless the workpiece weight is set to zero.
- If the robot stops due to an error and needs to be manually restored, place the robot in the desired position in the Recovery Mode and unload the workpiece through Servo On and I/O operation while the corresponding zones are activated in Auto Mode.
- When changing the set tool weight, the workpiece weight is initialized to 0.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|-----------------|----------------|---------------|--|
| weight | int | 0 | Weight [kg] |
| cog | list(float[3]) | [0, 0, 0] | Center of gravity of the workpiece (x, y, z) [mm] |
| cog_ref | int | DR_CUR_TCP | Reference coordinates of center of gravity position, DR_CUR_TCP : TCP coordinates, DR_FLANGE : FLANGE coordinates, |
| add_up | int | DR_REPLACE | DR_REPLACE(0): Replace workpiece DR_ADD(1): Add workpiece DR_REMOVE(2): Remove workpiece |
| start_time | float | None | Starting time of changing workpiece weight [sec] |
| transition_time | float | None | Transition time of changing workpiece weight [sec] |

ⓘ Note

- Workpiece weight cannot exceed the maximum payload weight (margin 10%) for each model.
The same is true for the weight of total payload.

- The length (x, y, z) of the center of gravity of the workpiece cannot exceed 1000 mm. The same is true for the length of the center of gravity of the entire payload.
- It is possible to change the weight of the workpiece after the set time through start_time.
- Transition_time allows you to gradually change the weight of the workpiece through transition_time.
- When using the set_tool and set_workpiece_weight functions in succession, you must use the wait(transition_time) function as much as transition_time between them. Otherwise, there may be errors in the weight change.

Return

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

Exception

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

Example

```

1 # Replace 1 kgf workpiece weight at the origin of the current TCP
2   frame
3   set workpiece weight( 1 )
4   # Add 0.5 kgf workpiece weight at the origin of the current TCP
5   # frame. In total, 1.5kgf
6   set workpiece weight( 0.5 , add up = DR ADD)
7   # Remove 0.1 kgf workpiece weight at [0, 0, 10] in the current TCP
8   # frame. In total, 1.4kgf
9   set workpiece weight( 0.1 , [ 0 , 0 , 10 ] , add up = DR REMOVE)
10  # Remove 0.1 kgf workpiece weight at [0, 0, 10] in the flange
11  # frame. In total, 1.3kgf

```

```

8   set_workpiece_weight( 0.1 , [ 0 , 0 , 10 ] , DR_FLANGE ,
9     DR_REMOVE )
# Reset. And the weight transition is being occurred after 0.1 sec
10    for 0.2 sec
11      set_workpiece_weight( 0 , start_time = 0.1 , transition_time = 0.2 )

```

Related Commands

- [set_tool\(name, start_time, transition_time\)\(p. 176\)](#)

5.1.4 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 |

| Exception | Description |
|--------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

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

Related commands

- [set_tcp\(name\)\(p. 51\)](#)

5.1.5 set_tool(name, start_time, transition_time)

Features

Teach Pendant>Workcell Manager>Activate the Tool Weight workcell item with the entered name from among the Tool Weight workcell items registered in the robot.

Tool weight can be changed after setting time(start_time) and during setting time(transition_time).

Parameters

| Parameter Name | Data Type | Default Value | Description |
|-----------------|-----------|---------------|---|
| name | string | - | The name of the tool weight registered in the Workcell Manager. |
| start_time | float | None | Tool weight is changed after setting time |
| transition_time | float | None | Tool weight is changed during setting time |

Note

- When using the set_tool and set_workpiece_weight functions in succession, you must use the wait(transition_time) function as much as transition_time between them. Otherwise, there may be errors in the weight change.

Return

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

Exception

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

Example

```

1 set_tool ( "tool1" ,start_time = 1 ,transition_time = 2 )
2 # After 1s and then activate the information of "tool1"
   registered in TP during 2s.

```

Related commands

- [set_tcp\(name\)\(p.51\)](#)

5.2 Control Mode Settings

5.2.1 set_singularity_handling(mode)

Features

Allows the user to select a response policy when a path deviation occurs due to a singularity in task motion. The mode can be set as follows

- Automatic avoidance mode(Default) : DR_AVOID
- Path first mode : DR_TASK_STOP

- Variable velocity mode : DR_VAR_VEL

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

Parameters

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

Return

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

Exception

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

Example

| | |
|---|--|
| 1 | P1 = posx(400 , 500 , 800 , 0 , 180 , 0) |
| 2 | P2 = posx(400 , 500 , 500 , 0 , 180 , 0) |

```

3   P3 = posx( 400 , 500 , 200 , 0 , 180 , 0 )
4   set_singularity_handling(DR_AVOID) # Automatic avoidance mode
5   for singularity
6     movel(P1, vel = 10 , acc = 20 )
7     set_velx( 30 )
8     set_accx( 60 )
9     set_singularity_handling(DR_TASK_STOP) # Task motion path first
10    movel(P2)
11    set_singularity_handling(DR_VAR_VEL) # Variable velocity mode for
singularity
12    movel(P3)

```

Related commands

- [movel\(\)](#)(p. 59)
- [movec\(\)](#)(p. 68)
- [movesx\(\)](#)(p. 77)
- [moveb\(\)](#)(p. 81)
- [move_spiral\(\)](#)(p. 85)
- [amovel\(\)](#)(p. 98)
- [amovec\(\)](#)(p. 104)
- [amovesx\(\)](#)(p. 111)
- [amoveb\(\)](#)(p. 114)
- [amove_spiral\(\)](#)(p. 118)

5.2.2 set_singularity_handling_force(mode)

Features

The program is terminated by default through error processing when compliance or force control are used within the singularity area. It is possible to ignore error processing within the singularity area by changing the Mode setting.

- Error Processing : DR_SINGULARITY_ERROR
- Ignore Error Processing : DR_SINGULARITY_IGNORE

⚠ Caution

- Compliance and force control within the singularity area are not recommended. The force estimate in a particular direction can be inaccurate.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|----------------------|--|
| mode | int | DR_SINGULARITY_ERROR | DR_SINGULARITY_ERROR : Error processing DR_SINGULARITY_IGNORE : Ignore error processing |

Return

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

Exception

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

Example

```

1 set_ref_coord(DR_BASE)
2 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
3 movej(P0,vel = 30 ,acc = 60 )
4
5 #Ignoring error when entering singularity
6 set_singular_handling_force(DR_SINGULARITY_IGNORE)
7
8 task_compliance_ctrl()
9 set_stiffnessx([ 500 , 500 , 500 , 100 , 100 , 100 ], time = 0
.5 )
10 fd = [ 0 , 0 , 30 , 0 , 0 , 0 ]
11 fctrl_dir = [ 0 , 0 , 1 , 0 , 0 , 0 ]
```

```

12   set_desired_force(fd, dir = fctrl_dir, mod = DR_FC_MOD_REL)
13   release_compliance_ctrl()

```

Related Commands

- [task_compliance_ctrl\(stx, time\)\(p. 186\)](#)
- [set_stiffnessx\(stx, time\)\(p. 187\)](#)
- [set_desired_force\(fd, dir, time, mod\)\(p. 189\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

5.2.3 set_palletizing_mode(mode)

Features

During palletizing application motion, path tracking and velocity can be maintained around the wrist singularity point using this function. there is no instability in wrist singular region when B in motion command is set to 0deg or 180deg.

- Deactivate mode : DR_OFF
- Activate mode : DR_ON

⚠ Caution

- Setting tool orientation, B must be set to 0deg or 180deg when setting tcp information. If this condition don't be satisfied, Error is occurred when using this function.
- Normally velocity don't be changed in this mode. But if current joint velocity exceed allowable max joint velocity, velocity can be reduced automatically.
- In case of H series model, Rx, Ry moment control is restricted and external moment value of each Rx, Ry direction is 0.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| mode | int | DR_ON | DR_OFF : Deactivate mode DR_ON : Activate mode |

Return

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

| Value | Description |
|----------------|-------------|
| Negative value | Error |

Exception

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

Example

```

1 set_singularity_handling(DR_VAR_VEL)
2
3 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ),vel = 30 ,acc = 60 )
4
5 set_palletizing_mode(DR_ON)
6 movel(posx( 559 , 34.5 , - 400 , 45 , 180 , 45 ),vel = 500 ,acc =
7 1000 )
8 set_palletizing_mode(DR_OFF)

```

Related Commands

- [set_singularity_handling\(mode\)\(p. 177\)](#)

5.2.4 set_motion_end(mode)

Features

This command sets whether to operate the function to check the stop status of the robot after motion is completed. Stop time between consecutive motions decreases if it set to deactivate mode (DR_CHECK_OFF) and can be used for purposes of decreasing the overall work time. It is recommended to set it to DR_CHECK_ON when the tool is heavy and an accurate stop position is required for motion commands driven with high acceleration.

⚠ Caution

- It is not possible to change the mode, during the blending movements between consecutive motions without stopping.
- In the case of continuous motion that does not require a stop state, using motion blending is more effective in reducing tact time.
- After the program is finished, it is initialized to the default value again

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| mode | int | DR_CHECK_ON | DR_CHECK_OFF(0) : Deactivate mode DR_CHECK_ON(1) : Activate mode |

Return

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

Exception

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

Example

```

1   set_motion_end(DR_CHECK_OFF)
2
3   movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ) ,vel = 30 ,acc = 60 ,mod =
      DR_MV_MOD_ABS )

```

```
4  while 1 :
5      movei(posi( 0 , 0 , 10 , 0 , 10 , 0 ) ,vel = 30 ,acc = 60 ,,mo
d = DR_MV_MOD_REL )
6      movei(posi( 0 , 0 , - 10 , 0 , - 10 , 0 ) ,vel = 30 ,acc = 60 ,
,mod = DR_MV_MOD_REL )
```

Related Commands

- [movei\(\)](#)(p. 59)
- [movec\(\)](#)(p. 68)
- [movesx\(\)](#)(p. 77)
- [moveb\(\)](#)(p. 81)
- [move_spiral\(\)](#)(p. 85)
- [amovei\(\)](#)(p. 98)
- [amovec\(\)](#)(p. 104)
- [amovesx\(\)](#)(p. 111)
- [amoveb\(\)](#)(p. 114)
- [amove_spiral\(\)](#)(p. 118)

6 Force/Stiffness Control and Other User-Friendly Features

6.1 Force/Compliance Control

6.1.1 release_compliance_ctrl()

Features

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

Return

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

Exception

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

Example

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movei(P0)
3 task_compliance_ctrl()
4 set_stiffnessx([ 100 , 100 , 300 , 100 , 100 , 100 ])
5 release_compliance_ctrl()

```

Related commands

- task_compliance_ctrl(stx, time)(p. 186)
- set_stiffnessx(stx, time)(p. 187)

6.1.2 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 Three rotational stiffnesses |
| time | float | 0 | Stiffness varying time [sec] Range: 0 - 1.0 * Linear transition during the specified time |

Return

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

Exception

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

Example

```

1 P0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
2 movei(P0)
3 task compliance_ctrl()          # Begins with the default stiffness
4 set_stiffnessx([ 500 , 500 , 500 , 100 , 100 , 100 ], time = 0
.5 )
5 # Switches to the user-defined stiffness for 0.5 sec.
6 release_compliance_ctrl()
7
8 task compliance_ctrl([ 500 , 500 , 500 , 100 , 100 , 100 ])
9 # Begins with the user-defined stiffness.
10 release_compliance_ctrl()
```

Related commands

- [set_stiffnessx\(stx, time\)\(p. 187\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

6.1.3 set_stiffnessx(stx, time)

Features

This function sets the stiffness value based on the global coordinate (refer to [set_ref_coord\(\)](#)). The stiffness linearly changes for a given time from the current stiffness or default value to STX. The user-defined ranges of the translational stiffness and rotational stiffness are 0-20000N/m and 0-400Nm/rad, respectively.

Parameters

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

Return

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

Exception

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

Example

```

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

Related commands

- [task_compliance_ctrl\(stx, time\)\(p. 186\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

6.1.4 set_desired_force(fd, dir, time, mod)

Features

This function define 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 Three rotational target moments |
| dir | int[6] | [0, 0, 0, 0, 0, 0] | Force control in the corresponding direction if 1 Control compliance of corresponding direction if value is 0 |
| time | float | 0 | Transition time of target force to take effect [sec] Range: 0 - 1.0 |
| mod | int | DR_FC_MOD_ABS | DR_FC_MOD_ABS: Force control with absolute value DR_FC_MOD_REL: force control with relative value to initial state (the instance when this function is called) |

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

```

1 # Example # 1
2 # Executed in the global coordinate(tool coordinate)
3 # Zero force control in the z-axis direction of the tool, moment
4 # control in the z-axis direction of the tool, and compliance
5 # control in the other directions
6 # Force control with the relative value to the sensor measurement
7 # at starting the force control
8
9 set_ref_coord(DR_TOOL)
10 x0 = posx( 0 , 0 , 90 , 0 , 90 , 0 )
11 movej(x0)
12 task_compliance_ctrl(stx = [ 500 , 500 , 500 , 100 , 100 ,
13 100 ])
14 fd = [ 0 , 0 , 0 , 0 , 0 , 10 ]
15 fctrl_dir = [ 0 , 0 , 1 , 0 , 0 , 1 ]
16 set_desired_force(fd, dir = fctrl_dir, mod = DR_FC_MOD_REL)

```

```

14 # Example #2
15 # 1. Move to initial posj: [J1, J2, J3, J4, J5, J6] = [0, 0, 90,
16 0, 90, 0]
17 # 2. Approach to the position to start force control: move -100mm
18 along Base-z direction
19 # 3. Start force control : apply -20N force along Base-z direction
20 # 4. Force & compliance control after detecting external force :
21 while maintaining -20N force along Base-z direction (force
22 control), move 200mm along Base-y direction.
23 # 5. Retract 150mm in Base-z direction and move to initial posj
24
25 # 1. Move to initial posi
26 q0 = posj( 0.0 , 0.0 , 90.0 , 0.0 , 90.0 , 0.0 )
27 set veli( 30.0 )
28 set acci( 60.0 )
29 movej(q0)
30
31 # 2. Approach to the position to start force control
32 set velx( 75.0 )
33 set accx( 100.0 )
34 delta_approach = [ 0.0 , 0.0 , - 100.0 , 0.0 , 0.0 , 0.0 ]
35 movel(delta_approach, mod = DR_MV_MOD_REL)
36
37 # 3. Start force control (apply -20N force along Base-z direction)
38 k_d = [ 3000.0 , 3000.0 , 3000.0 , 200.0 , 200.0 , 200.0 ]
39 task_compliance_ctrl(k_d)
40 force_desired = 20.0
41 f_d = [ 0.0 , 0.0 , - force_desired, 0.0 , 0.0 , 0.0 ]
42 f_dir = [ 0 , 0 , 1 , 0 , 0 , 0 ]
43 set_desired_force(f_d, f_dir)
44
45 # 4. Force & compliance control after detecting external force
46 force_check = 20.0
47 force_condition = check_force_condition(DR_AXIS_Z, max = force_
48 check)
49 while (force_condition):
50     force_condition = check_force_condition(DR_AXIS_Z, max = fo
51 rce_check)
52     if force_condition == 0 :
53         break
54 delta_motion = [ 0.0 , 200.0 , 0.0 , 0.0 , 0.0 , 0.0 ]
55 movel(delta_motion, mod = DR_MV_MOD_REL)
56
57 # 5. Retract 150mm in Base-z direction and move to initial posj
58 release force()
59 wait( 0.5 )
60 delta_retract = [ 0.0 , 0.0 , 150.0 , 0.0 , 0.0 , 0.0 ]
61 release compliance_ctrl()
62 movel(delta_retract, mod = DR_MV_MOD_REL)
63 movej(q0)

```

Related commands

- [release_force\(time=0\)\(p. 192\)](#)
- [task_compliance_ctrl\(stx, time\)\(p. 186\)](#)
- [set_stiffnessx\(stx, time\)\(p. 187\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

6.1.5 release_force(time=0)

Features

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

Parameters

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

Return

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

Exception

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

Example

```
1 | j0 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
```

```

2  x0 = posx( 0 , 0 , 0 , 0 , 0 , 0 )
3  x1 = posx( 0 , 500 , 700 , 0 , 180 , 0 )
4  x2 = posx( 300 , 100 , 700 , 0 , 180 , 0 )
5  x3 = posx( 300 , 100 , 500 , 0 , 180 , 0 )
6  set_velx( 100 , 20 )
7  set_accx( 100 , 20 )
8  movei(i0, vel = 10 , acc = 10 )
9  movel(x2)
10 task compliance_ctrl(stx = [ 500 , 500 , 500 , 100 , 100 ,
11 100 ])
12 fd = [ 0 , 0 , 0 , 0 , 0 , 10 ]
13 fctrl_dir = [ 0 , 0 , 1 , 0 , 0 , 1 ]
14 set_desired_force(fd, dir = fctrl_dir, time = 1.0 )
15 movel(x3, v = 10 )
16 release_force( 0.5 )
release_compliance_ctrl()

```

Related commands

- [set_desired_force\(fd, dir, time, mod\)\(p. 189\)](#)
- [task_compliance_ctrl\(stx, time\)\(p. 186\)](#)
- [set_stiffnessx\(stx, time\)\(p. 187\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

6.1.6 get_force_control_state()

Features

It monitors the state of compliance and force control.

Return

[singularity, mode, stx, fd, ref]

| Value | Description |
|-------------|--|
| singularity | Risk : 0 < 1 < 2 0 : Safe section 1 : Stage 1 risk section of the singularity area 2 : Stage 2 risk section of the singularity area |
| mode | Information of 6 modes x, y, z, rx, ry and rz (sequential) 0 : Compliance control 1 : Force control 2 : None |

| Value | Description |
|-------|--|
| Stx | 6 in the order of x, y, z, rx, ry and rz Information of set target stiffness |
| fd | 6 in the order of x, y, z, rx, ry and rz Information of set target force |
| ref | Information of set target force 0 : Base coordinate 1 : Tool coordinate 2 : World coordinate 101 ~ 120 : User coordinate |

Exception

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

Example

```

1  set_ref_coord(DR_BASE)
2  P0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
3  movej(P0,vel = 30 ,acc = 60 )
4
5  task compliance_ctrl()
6  set_stiffnessx([ 500 , 500 , 500 , 100 , 100 , 100 ], time = 0
.5 )
7
8  while True :
9      [singularity, mod, stx, fd, ref] = get_force_control_state()
10     tp_log( "s={0}, m={1}, k={2}, f={3}, r={4}" . format (singularity,
mod,stx,fd,ref))
11     wait( 0.5 )
12     release_compliance_ctrl()
```

Related Commands

- [set_singular_handling_force\(mode\)\(p. 179\)](#)
- [task_compliance_ctrl\(stx, time\)\(p. 186\)](#)
- [set_stiffnessx\(stx, time\)\(p. 187\)](#)
- [set_desired_force\(fd, dir, time, mod\)\(p. 189\)](#)
- [release_compliance_ctrl\(\)\(p. 185\)](#)

6.2 User-friendly Functions

6.2.1 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 position list |
| | list (float[6]) | | |
| x2 | posx | - | posx or position list |
| | list (float[6]) | | |
| x3 | posx | - | posx or position list |
| | list (float[6]) | | |
| axis | int | - | axis <ul style="list-style-type: none"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |

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

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  x0 = posx( 0 , 0 , 90 , 0 , 90 , 0 )
2  movej(x0)
3  x1 = posx( 0 , 500 , 700 , 30 , 0 , 90 )
4  x2 = posx( 500 , 0 , 700 , 0 , 0 , 45 )
5  x3 = posx( 300 , 100 , 500 , 45 , 0 , 45 )
6  parallel_axis(x1, x2, x3, DR AXIS X, DR WORLD)
7  # 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\(x1, x2, x3\)\(p. 296\)](#)
- [parallel_axis\(vect, axis, ref\)\(p. 197\)](#)

- [parallel_axis\(x1, x2, x3, axis, ref\)\(p. 195\)](#)
- [align_axis\(vect, pos, axis, ref\)\(p. 200\)](#)
- [align_axis\(x1, x2, x3, pos, axis, ref\)\(p. 198\)](#)

6.2.2 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"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |
| ref | int | DR_BASE | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate: user defined |

Return

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

Exception

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

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

Example

```

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

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [parallel_axis\(x1, x2, x3, axis, ref\)\(p. 195\)](#)
- [align_axis\(vect, pos, axis, ref\)\(p. 200\)](#)
- [align_axis\(x1, x2, x3, pos, axis, ref\)\(p. 198\)](#)

6.2.3 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]) | | |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|--|
| 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"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |
| ref | int | DR_BASE | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate: user defined |

Return

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

Exception

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

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

Example

```

1 p0 = posi( 0 , 0 , 45 , 0 , 90 , 0 )
2 movej(p0, v = 30 , a = 30 )
3
4 x1 = posx( 0 , 500 , 700 , 30 , 0 , 0 )
5 x2 = posx( 500 , 0 , 700 , 0 , 0 , 0 )
6 x3 = posx( 300 , 100 , 500 , 0 , 0 , 0 )
7 pos = posx( 400 , 400 , 500 , 0 , 0 , 0 )
8 align axis(x1, x2, x3, pos, DR AXIS X, DR BASE)
9 # match the tool x axis and the normal vector in the plane
10 # consists of points(x1, x2,
# x3) based on the base coordinate

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [get_normal\(x1, x2, x3\)\(p. 296\)](#)
- [align_axis\(vect, pos, axis, ref\)\(p. 200\)](#)
- [parallel_axis\(vect, axis, ref\)\(p. 197\)](#)
- [parallel_axis\(x1, x2, x3, axis, ref\)\(p. 195\)](#)

6.2.4 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 position list |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|---|
| | list (float[6]) | | |
| axis | int | - | axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |

Return

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

Exception

| Exception | Description |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_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 , 45 , 0 , 90 , 0 )
2  movej(p0, v = 30 , a = 30 )
3
4  vect = [ 10 , 20 , 30 ]
5  pos = posx( 100 , 500 , 700 , 45 , 0 , 0 )
6  align_axis(vect, pos, DR_AXIS_X)

```

Related commands

- [movej\(\)\(p. 54\)](#)
- [align_axis\(x1, x2, x3, pos, axis, ref\)\(p. 198\)](#)

- [parallel_axis\(vect, axis, ref\)\(p. 197\)](#)
- [parallel_axis\(x1, x2, x3, axis, ref\)\(p. 195\)](#)

6.2.5 `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 <ul style="list-style-type: none"> • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis |

Return

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

Exception

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

Example

```

1  p0 = posi( 0 , 0 , 90 , 0 , 90 , 0 )
2  movej(p0, v = 30 , a = 30 )
3
4  task compliance_ctrl()
5  xd = posx( 559 , 34.5 , 651.5 , 0 , 180.0 , 60 )
6  amovel(xd, vel = 50 , acc = 50 ) # Bolt tightening motion
7
8  res = is done bolt tightening( 10 , 5 , DR AXIS Z)
9      # Returns True if the tightening torque of 10Nm is reached
# within 5 seconds.
10     # Returns False otherwise.
11  if res == True :
12      # some action comes here for the case that bolt tightening is
done
13      x = 1
14  else :
15      # some action comes here for the case that it fails
16      x = 2

```

Related commands

- [movej\(\)](#)(p. 54)
- [amovel\(\)](#)(p. 98)

6.2.6 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 list (float[6]) | - | Posx or position list |
| ref | int | - | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate |
| mod | int | - | input mode (only valid when the number of input poses is 2) <ul style="list-style-type: none"> • 0: defining z-axis based on the current Tool-z direction • 1: defining z-axis based on the z direction of x1 |

Return

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

Exception

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

| Exception | Description |
|--------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 pos1 = posx( 500 , 30 , 500 , 0 , 0 , 0 )
2 pos2 = posx( 400 , 30 , 500 , 0 , 0 , 0 )
3 pos3 = posx( 500 , 30 , 600 , 45 , 180 , 45 )
4 pos4 = posx( 500 , -30 , 600 , 0 , 180 , 0 )
5 pose_user1 = calc_coord(pos1, ref = DR_BASE, mod = 0 )
6 pose_user21 = calc_coord(pos1, pos2, ref = DR_WORLD, mod = 0 )
% % Define z - axis based on the Tool - z direction.
7 pose_user22 = calc_coord(pos1, pos2, ref = DR_BASE, mod = 1 )
% % Define z - axis based on the z direction of pos1
8 pose_user3 = calc_coord(pos1, pos2, pos3, ref = DR_BASE, mod = 0 )
9 pose_user4 = calc_coord(pos1, pos2, pos3, pos4, ref = DR_WORLD,
10 mod = 0 )
11 ucart1 = set_user_cart_coord(pose_user1, ref = DR_BASE)
12 ucart2 = set_user_cart_coord(pose_user21, ref = DR_WORLD)
13

```

Related commands

- [set_user_cart_coord\(pos, ref\)\(p. 205\)](#)
- [set_user_cart_coord\(u1, v1, pos, ref\)\(p. 208\)](#)
- [set_user_cart_coord\(x1, x2, x3, pos, ref\)\(p. 207\)](#)

6.2.7 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 (position and orientation) |
| | list (float[6]) | | |

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

Return

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

Exception

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

Example

```

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

```

Related commands

- [set_ref_coord\(coord\)\(p. 52\)](#)

6.2.8 set_user_cart_coord(x1, x2, x3, pos, ref)

Features

This function sets a new user cartesian coordinate system using [x1], [x2], and [x3] based on ref coordinate system[ref]. Creates a user coordinate system with ux, uy, and uz as the vector for each axis and origin position is the position of [pos] based on [ref]. ¹⁾ux is defined as the unit vector of x1x2 , uz is defined as the unit vector defined by the cross product of x1x2 and x1x3 (x1x2 cross x1x3). uy is can be determined by right hand rule (uz cross ux). Up to 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.

¹⁾ In software versions lower than M2.0.2, ux is used as the unit vector of x2x1

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|---|
| x1 | Posx | - | 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 <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate |

Return

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

Exception

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

Example

```

1  x1 = posx( 0 , 500 , 700 , 0 , 0 , 0 ) # Ignores the Euler
2  angle.
3  x2 = posx( 500 , 0 , 700 , 0 , 0 , 0 )
4  x3 = posx( 300 , 100 , 500 , 0 , 0 , 0 )
5  x4 = posx( 300 , 110 , 510 , 0 , 0 , 0 )
6  pos = posx( 10 , 20 , 30 , 0 , 0 , 0 )
7  user_tc1 = set_user_cart_coord(x1, x2, x3, pos, ref = DR_BASE)
    user_tc2 = set_user_cart_coord(x2, x3, x4, pos, ref = DR_WORLD)

```

Related commands

- [set_ref_coord\(coord\)\(p. 52\)](#)

6.2.9 set_user_cart_coord(u1, v1, pos, ref)

Features

This function sets a new user cartesian coordinate system using [u1] and [v1] based on [ref] coordinate system. The origin position the position of [pos] based on the [ref] coordinate while the direction of x-axis and y-axis

bases are given in the vectors u1 and v1, respectively. Other directions are determined by u1 cross v1. If u1 and v1 are not orthogonal, v1', that is perpendicular to u1 on the surface spanned by u1 and v1, is set as the vector in the y-axis direction. Up to 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 list (float[6]) | - | posx or position list |
| ref | int | DR_BASE | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate |

Return

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

Exception

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

| Exception | Description |
|--------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

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

```

Related commands

- [set_ref_coord\(coord\)\(p.52\)](#)

6.2.10 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 list (float[6]) | - | posx or position list |
| ref | int | DR_BASE | reference coordinate <ul style="list-style-type: none"> • DR_BASE: base coordinate • DR_WORLD: world coordinate |

Return

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

Exception

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

Exception

```

1  pose user1  =  posx( 30 ,  40 ,  50 ,  0 ,  0 ,  0 )
2  id_user   =  set_user_cart_coord(pose_user1, ref = DR_BASE)
3  pose user2  =  posx( 100 , 150 , 200 , 45 , 180 , 0 )
4  overwrite_user_cart_coord(id_user, pose_user2, ref = DR_BASE)

```

Related commands

- [set_user_cart_coord\(pos, ref\)\(p. 205\)](#)
- [set_user_cart_coord\(u1, v1, pos, ref\)\(p. 208\)](#)
- [set_user_cart_coord\(x1, x2, x3, pos, ref\)\(p. 207\)](#)

6.2.11 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

```

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

```

Related commands

- [set_user_cart_coord\(pos, ref\)\(p. 205\)](#)
- [set_user_cart_coord\(u1, v1, pos, ref\)\(p. 208\)](#)
- [set_user_cart_coord\(x1, x2, x3, pos, ref\)\(p. 207\)](#)

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

Note

- The absolution position is used if the mod is DR_MV_MOD_ABS.

- The pos position is used if the mod is DR_MV_MOD_REL.
- Pos is meaningful only if the mod is DR_MV_MOD_REL.

Return

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

Exception

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

Example

```

1 CON1 = check_position_condition(DR_AXIS_X, min = - 5 , max = 0 ,
2 ref = DR_WORLD)
3 CON2 = check_position_condition(DR_AXIS_Y, max = 700 )
4 CON3 = check_position_condition(DR_AXIS_Z, min = - 10 , max = - 5
# -10≤z≤-5
5 CON4 = check_position_condition(DR_AXIS_Z, min = 30 ) # 30≤z
6 CON5 = check_position_condition(DR_AXIS_Z, min = - 10 , max = - 5 ,
7 ref = DR_BASE) # -10≤z≤-5
8 CON6 = check_position_condition(DR_AXIS_Z, min = - 10 , max = - 5 ,
9 mod = DR_MV_MOD_ABS) # -10≤z≤-5
10 posx1 = posx( 400 , 500 , 800 , 0 , 180 , 0 )
11 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\(axis, min, max, ref\)\(p. 215\)](#)
- [check_orientation_condition\(axis, min, max, ref, mod\)\(p. 217\)](#)
- [check_orientation_condition\(axis, min, max, ref, mod, pos\)\(p. 219\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)

6.2.13 check_force_condition(axis, min, max, ref)

Features

This function checks the status of the given force. It disregards the force direction and only compares the sizes. This condition can be repeated with the while or if statement. Measuring the force and ¹⁾moment, axis is based on the ref coordinate.

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

Parameters

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

Return

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

Exception

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

Example

```

1   fcon1 = check_force_condition(DR_AXIS_Z, min = 5 , max = 10 ,
2                                 ref = DR_WORLD)           # 5 ≤f_z≤10
3
4   while (fcon1):
5       fcon2 = check_force_condition(DR_AXIS_C, min = 30 )
6       # 30≤m_z
7       pcon1 = check_position_condition(DR_AXIS_X, min = 0 , max = 0
8       .1 )      # 0≤x≤0.1
9
10      if (fcon2 and pcon1):
11          break

```

Related commands

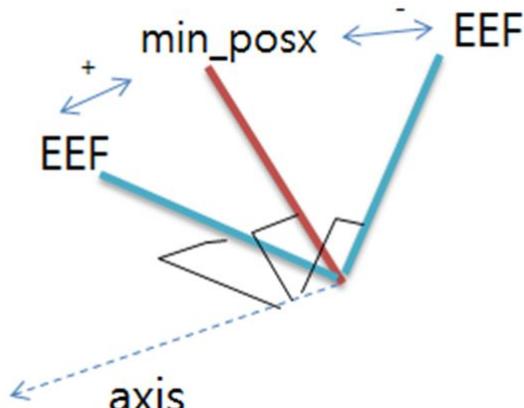
- [check_position_condition\(axis, min, max, ref, mod, pos\)\(p. 213\)](#)
- [check_orientation_condition\(axis, min, max, ref, mod\)\(p. 217\)](#)
- [check_orientation_condition\(axis, min, max, ref, mod, pos\)\(p. 219\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)

6.2.14 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"> • DR_AXIS_A: x-axis rotation • DR_AXIS_B: y-axis rotation • DR_AXIS_C: z-axis rotation |
| min | posx | - | posx or position list |
| | list (float[6]) | | |

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

| | |
|---|--|
| 1 | posx1 = posx(400 , 500 , 800 , 0 , 180 , 30) |
| 2 | posx2 = posx(400 , 500 , 500 , 0 , 180 , 60) |
| 3 | |

```

4 CON1 = check_orientation_condition(DR_AXIS_C, min = posx1, max =
5   posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0,
6   180, 40)
# CON1=True since posx1 Rz=30 < posxc Rz=40 < posx2 Rz=60
7
8 CON2 = check_orientation_condition(DR_AXIS_C, min = posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0,
9   180, 15)
# CON2=False since posx1 Rz=30 > posxc Rz=15
10
11 CON3 = check_orientation_condition(DR_AXIS_C, max = posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0,
12   180, 75)
# CON2=False since posx1 Rz=75 > posxc Rz=60
13
14

```

Related commands

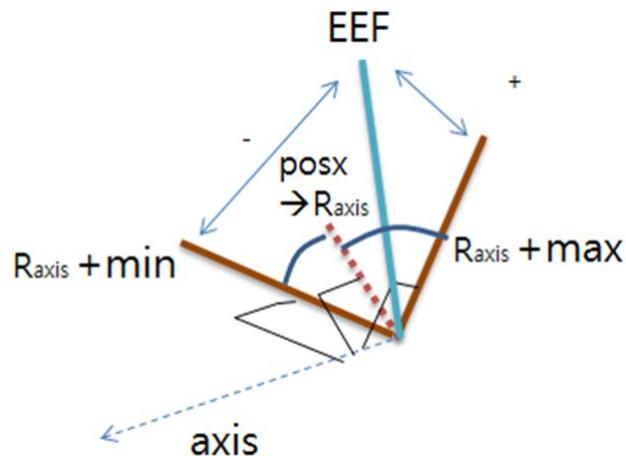
- [check_position_condition\(axis, min, max, ref, mod, pos\)](#)(p. 213)
- [check_force_condition\(axis, min, max, ref\)](#)(p. 215)
- [check_orientation_condition\(axis, min, max, ref, mod\)](#)(p. 217)
- [check_orientation_condition\(axis, min, max, ref, mod, pos\)](#)(p. 219)
- [set_ref_coord\(coord\)](#)(p. 52)

6.2.15 **check_orientation_condition(axis, min, max, ref, mod, pos)**

Features

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

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



Note

Range of rotating angle: Refers to the relative angle range (min, max) based on the set axis from the given position. The reference coordinate is defined according to the given position based on ref.

Parameters

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

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

```

1 posx1 = posx( 400 , 500 , 800 , 0 , 180 , 15 )
2 CON1 = check_orientation_condition(DR_AXIS_C, min = - 5 , mod = DR
3 MV_MOD_REL, pos = posx1, DR_WORLD)
# If the current task coordinate posxc = posx(400, 500, 500, 0,
4 180, 40)
# CON1=True since posx1 Rz=15 - (min=5) < posxc Rz=40
5
6 CON1 = check orientation condition(DR_AXIS_C, max = 5 , mod = DR
7 MV MOD REL, pos = posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0,
8 180, 40)
# CON1=False since posxc Rz=40 > posx1 Rz=15 + (max=5)

```

Related commands

- [check_position_condition\(axis, min, max, ref, mod, pos\)\(p. 213\)](#)
- [check_force_condition\(axis, min, max, ref\)\(p. 215\)](#)

- [check_orientation_condition\(axis, min, max, ref, mod\)\(p. 217\)](#)
- [check_orientation_condition\(axis, min, max, ref, mod, pos\)\(p. 219\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)

6.2.16 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 | <p>reference coordinate before transformation</p> <ul style="list-style-type: none"> • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined |

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

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

```

1  base_pos = posx( 400 , 500 , 800 , 0 , 180 , 15 )
2  # If task position based on base reference coordinate base_pos =
3  # posx(400,500,800,0,180,15)
4
5  tool_pos = coord transform(base_pos, DR_BASE, DR_TOOL)
6  # 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\(pos, ref\)\(p. 205\)](#)
- [set_user_cart_coord\(u1, v1, pos, ref\)\(p. 208\)](#)
- [set_user_cart_coord\(x1, x2, x3, pos, ref\)\(p. 207\)](#)
- [get_current_posx\(ref\)\(p. 155\)](#)
- [get_desired_posx\(ref\)\(p. 164\)](#)
- [set_ref_coord\(coord\)\(p. 52\)](#)

7 System Commands

7.1 IO Related

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

 **Note**

If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

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

| Value | Description |
|----------------|-------------|
| Negative value | Failed |

Exception

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

Example

```

1 set_digital_output( 1 , ON)           # No. 1 contact ON
2 set_digital_output( 16 , OFF)        # No. 16 contact OFF
3 set_digital_output( 3 )             #No. 3 contact ON (A positive
number means ON if the argument val is omitted.)
4 set_digital_output( -3 )            #No. 3 contact OFF (A negative
number means OFF if the argument val is omitted.)
```

7.1.2 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) | - | <p>List of multiple output contacts</p> <ul style="list-style-type: none"> • The positive contact number outputs ON: 1~16 • The negative contact number outputs OFF: -1~-16 |

Return

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

Exception

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

Example

```

1  set digital outputs(bit list = [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 ])
# Contact number 1-8 ON
2  set digital outputs([- 1 , - 2 , - 3 , - 4 , - 5 , - 6 , - 7 , -
8 ])    # Contact number 1-8 OFF
3  set digital outputs([- 1 , - 2 , 3 ])                      # Contact no.
1 ON, no. 2 OFF, and no. 3 ON
4  set digital outputs([- 4 , - 9 , - 12 ])                  # Contact no.
4 ON, no. 9 OFF, and no. 12 OFF

```

7.1.3 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 |

(i) 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact
2 set digital outputs(bit start = 1 , bit_end = 4 , val = 0b0011 ) # 0b means a binary number.
3
4 # Outputs contact 3=ON and contact 4=OFF.
5 set digital outputs(bit start = 3 , bit_end = 4 , val = 0b01 ) # 0b means a binary number.
6
7 # Outputs the ON signal from contacts 1 through 8.
8 set_digital_outputs( 1 , 8 , 0xff ) # 0x means a hexadecimal number.

```

7.1.4 set_digital_output(index, val=None, time=None, val2=None)

Features

This function sends a signal at the digital contact point of the controller. A value saved in the digital output register is output as a digital signal. After sending out the specified signal for the set time, the next signal is sent out.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| index | int | - | I/O contact number mounted on the controller <ul style="list-style-type: none"> Val argument existing: A number between 1 and 16 No val argument: 1 ~ 16, -1 ~ -16 (A positive number means ON while a negative number means OFF.) |
| val | int | - | I/O value <ul style="list-style-type: none"> ON: 1 OFF: 0 |
| time | float | - | Time(0.01 ~ 3,000,000) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| val2 | int | - | I/O value <ul style="list-style-type: none"> • ON: 1 • OFF: 0 |

 **Note**

If val is omitted, the positive number becomes ON and the negative number OFF according to the sign of the argument index.

Return

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

Exception

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

Example

```

1  set_digital_output( 1 , ON,  2.0 , OFF)  # No. 1 contact ON, OFF
after 2 seconds
2  set_digital_output( 5 , OFF,  0.5 , OFF)      # No. 16 contact OFF,
ON after 0.5 seconds

```

7.1.5 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

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

```

7.1.6 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 once.

Parameters

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

Return

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

Exception

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

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

Example

```

1 # input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
2 res = get_digital_inputs(bit_list = [1, 2, 3, 4])
#res expected value = 0b1100 (binary number), 12 (decimal number),
or 0x0C (hexadecimal number)
4
5 # input contacts: No. 5=ON, No. 6=ON, No. 7=OFF, and No. 8=ON
6 res = get_digital_inputs([5, 6, 7, 8])
#res expected value = 0b1011 (binary number), 11 (decimal number),
or 0x0B (hexadecimal number)
7

```

7.1.7 get_digital_inputs(bit_start, bit_end)

Features

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

Parameters

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

Note

Bit_end must be a larger number than bit_start.

Return

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

Exception

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

Example

```

1 # input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
2 res  = get digital inputs(bit start = 1 , bit end = 4 )
3 #res expected value = 0b1100 (binary number), 12 (decimal number),
or 0x0C (hexadecimal number)

```

7.1.8 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 <ul style="list-style-type: none"> • ON : 1 • OFF : 0 |
| timeout | float | - | Waiting time (sec) This function waits indefinitely if the timeout is not set. |

Return

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

Exception

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

Example

```

1  wait digital input( 1 , ON) # Indefinite wait until the no. 1
2  contact becomes ON
3  wait digital input( 2 , OFF) # Indefinite wait until the no. 2
4  contact becomes OFF
5  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.

```

7.1.9 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"> • Val argument existing: A number between 1 and 6 • No val argument: 1 ~ 6, -1 ~ -6 (A positive number means ON while a negative number means OFF.) |
| val | int | - | I/O value: The value to output |

Note

If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

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

Exception

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

Example

```

1 set tool digital_output( 1 , ON)    # Sets the no. 1 contact of the
2 set tool digital_output( 6 , OFF)   # Sets the no. 6 contact of the
3 set tool digital output( 3          #No. 3 contact ON (A positive
4      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.)
```

7.1.10 set_tool_digital_outputs(bit_list)

Features

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

Parameters

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

Return

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

Exception

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

Example

```

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

```

7.1.11 set_tool_digital_outputs(bit_start, bit_end, val)

Features

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

Parameters

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

(i) Note

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

Return

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

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact
2   4=OFF.
3   set tool digital outputs(bit_start = 1 , bit_end = 4 , val = 0b001
4     1 ) # 0b means a binary number.
5
6 # Outputs contact 3=ON and contact 4=OFF.
7   set tool digital outputs(bit_start = 3 , bit_end = 4 , val = 0b01 )
8     # 0b means a binary number.
9
10 # Outputs the ON signal from contacts 1 through 8.
11   set tool digital outputs( 1 , 8 , 0xff )
12     # 0x means a hexadecimal number.

```

7.1.12 set_tool_digital_output(index, val=None, time=None, val2=None)

Features

This function sends the signal of the robot tool from the digital contact point. After sending out the specified signal for the set time, the next signal is sent out.

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"> • Val argument existing: A number between 1 and 6 • No val argument: 1 ~ 6 , -1 ~ -6 (A positive number means ON while a negative number means OFF.) |
| val | int | - | I/O value <ul style="list-style-type: none"> • ON: 1 • OFF: 0 |
| time | float | - | Time(0.01 ~ 3,000,000) |
| val2 | int | - | I/O value <ul style="list-style-type: none"> • ON: 1 • OFF: 0 |

i 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  set_tool_digital_output( 1 , ON,  2.0 , OFF)      # Sets the no. 1
2  contact of the robot arm ON, OFF after 2 seconds
   set_tool_digital_output( 5 , OFF,  0.5 , ON)      # Sets the no. 5
   contact of the robot arm OFF, ON after 0.5 seconds

```

7.1.13 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 |

| Value | Description |
|----------------|-------------|
| 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

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

```

7.1.14 get_tool_digital_inputs(bit_list)

Features

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

Parameters

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

Return

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

Exception

| Exception | Description |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| 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 # input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
2 res  = get_tool_digital_inputs(bit_list = [ 1 , 2 , 3 ]) # Reads
3   the contacts 1, 2, and 3 at once.
4   #res expected value = 0b100 (binary number), 4 (decimal number),
5   or 0x04 (hexadecimal number)
6
# input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
res  = get_tool_digital_inputs([ 4 , 5 , 6 ])

```

| | |
|---|---|
| 7 | #res expected value = 0b011 (binary number), 3 (decimal number), or 0x03 (hexadecimal number) |
|---|---|

7.1.15 get_tool_digital_inputs(bit_start, bit_end)

Features

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

Parameters

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

Return

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

Exception

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

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
2 res = get_tool_digital_inputs(bit_start = 1, bit_end = 3)
#res expected value = 0b100 (binary number), 4 (decimal number),
or 0x04 (hexadecimal number)
4
5 # input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
6 res = get_tool_digital_inputs( 4, 6 )
#res expected value = 0b011 (binary number), 3 (decimal number),
or 0x03 (hexadecimal number)
7

```

7.1.16 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 <ul style="list-style-type: none"> • ON:1 • OFF:0 |
| timeout | float | - | Waiting time (sec) This function waits indefinitely if the timeout is not set. |

Return

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

Exception

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

Example

```

1  wait tool digital input( 1 , ON)    # Indefinite wait until the no.
2  1 contact becomes ON
2  wait tool digital input( 2 , OFF)   # Indefinite wait until the no.
3  2 contact becomes OFF
4
5  res  = wait tool digital input( 1 , ON,  3 ) # Wait for up to 3
6  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.

```

7.1.17 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"> • 1 : channel 1 • 2 : channel 2 |
| mod | int | - | analog io mode <ul style="list-style-type: none"> • DR_ANALOG_CURRENT: Current mode • DR_ANALOG_VOLTAGE: Voltage mode |

Return

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

Exception

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

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Sets analog_output channel 1 to the current mode.
2 set_mode_analog_output(ch = 1 , mod = DR_ANALOG_CURRENT)
3
4 # Sets analog_output channel 2 to the voltage mode.
5 set_mode_analog_output(ch = 2 , mod = DR_ANALOG_VOLTAGE)

```

7.1.18 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"> • 1 : channel 1 • 2 : channel 2 |
| mod | int | - | analog io mode <ul style="list-style-type: none"> • DR_ANALOG_CURRENT: Current mode • DR_ANALOG_VOLTAGE: Voltage mode |

Return

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

Exception

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

Example

```

1 # Sets analog_input channel 1 to the current mode.
2 set_mode_analog_input(ch = 1 , mod = DR_ANALOG_CURRENT)
3
4 # Sets analog_input channel 2 to the voltage mode.
5 set_mode_analog_input(ch = 2 , mod = DR_ANALOG_VOLTAGE)

```

7.1.19 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"> • 1 : channel 1 • 2 : channel 2 |
| val | float | - | analog output value <ul style="list-style-type: none"> • Current mode: 4.0~20.0 [mA] • Voltage mode: 0~10.0 [V] |

Return

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

Exception

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

Example

```

1 set mode analog output(ch = 1 , mod = DR_ANALOG_CURRENT) #out
2 ch1=current mode
3 set mode analog output(ch = 2 , mod = DR_ANALOG_VOLTAGE) #out
4 ch1=voltage mode
5
6 set analog_output(ch = 1 , val = 5.2 )      # Outputs 5.2 mA to
7 channel 1
8 set_analog_output(ch = 2 , val = 10.0 )     # Outputs 10V to channel
9 2

```

7.1.20 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"> • 1 : channel 1 • 2 : channel 2 |

Return

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

Exception

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

Example

```

1  set_mode_analog_input(ch = 1 , mod = DR_ANALOG_CURRENT)    #input
2  ch1=current mode
3  set_mode_analog_input(ch = 2 , mod = DR_ANALOG_VOLTAGE)    #input
4  ch2=voltage mode
5
6  Cur   = get_analog_input( 1 )     # Reads the analog input current
7  value of channel 1

```

| | |
|---|---|
| 5 | Vol = get_analog_input(2) # Reads the analog input voltage value of channel 2 |
|---|---|

7.2 TP Interface

7.2.1 tp_popup(message, pm_type=DR_PM_MESSAGE, button_type=0)

Features

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

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| message | string | - | <p>Message provided to the user</p> <ul style="list-style-type: none"> Messages are limited to within 256 bytes. It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (...). Formatting-related code such as newline (\n) or carriage return (\r) is not allowed. |
| pm_type | int | DR_PM_MESSAGE | <p>Message type</p> <ul style="list-style-type: none"> DR_PM_MESSAGE DR_PM_WARNING DR_PM_ALARM |
| button_type | int | 0 | <p>button type of TP pop message</p> <ul style="list-style-type: none"> 0 : show Stop & Resume button 1 : show Stop button |

Return

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

Exception

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

Example

```

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

```

7.2.2 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 | - | <p>Log message</p> <ul style="list-style-type: none"> • Messages are limited to within 256 bytes. • It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (...). • Formatting-related code such as newline (\n) or carriage return (\r) is not allowed. |

Return

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

Exception

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

Example

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

7.2.3 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 | - | <p>Character string message to be displayed on the TP user input window</p> <ul style="list-style-type: none"> • Messages are limited to within 256 bytes. • It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (...). • Formatting-related code such as newline (\n) or carriage return (\r) is not allowed. |
| input_type | int | - | <p>TP user input message type</p> <ul style="list-style-type: none"> • DR_VAR_INT: Integer type • DR_VAR_FLOAT: Real number type • DR_VAR_STR: Character string • DR_VAR_BOOL: Boolean |

Return

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

Exception

| Exception | Description |
|-----------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| 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 | q1 = posj(10 , 10 , 10 , 10 , 10 , 10) |
| 2 | q2 = posj(20 , 20 , 20 , 20 , 20 , 20) |

```

3   q3 = posi( 30 , 30 , 30 , 30 , 30 , 30 )
4   q4 = posi( 40 , 40 , 40 , 40 , 40 , 40 )
5   q5 = posi( 50 , 50 , 50 , 50 , 50 , 50 )
6   q6 = posj( 60 , 60 , 60 , 60 , 60 , 60 )

7
8   int_v = tp_get_user_input( "message1" , input_type = DR_VAR_INT)
9   if int_v == 1 :          # Moves to q1 if the TP user input is 1.
10    movei(q1, vel = 30 , acc = 30 )
11   else :                  # Moves to q2 if the TP user input is not 1.
12    movej(q2, vel = 30 , acc = 30 )

13
14   float_v = tp_get_user_input( "message2" , input_type =
15      DR_VAR_FLOAT)
16   if float_y == 3.14 :     # Moves to q3 if the TP user input is
17      3.14.
18    movei(q3, vel = 30 , acc = 30 )
19   else :                  # Moves to q4 if the TP user input is not 3.14.
20    movej(q4, vel = 30 , acc = 30 )

21   str_v = tp_get_user_input( "message3" , input_type = DR_VAR_STR)
22   if str_y == "a" :        # Moves to q5 if the TP user input is
23      "a".
24    movei(q5, vel = 30 , acc = 30 )
25   else :                  # Moves to q6 if the TP user input is not "a".
26    movej(q6, vel = 30 , acc = 30 )

27   bool_v = tp_get_user_input( "message3" , input_type =
28      DR_VAR_BOOL)
29   if bool_v == True :       # Moves to q5 if the TP user input
30      is "True or 1".
31    movei(q5, vel = 30 , acc = 30 )
32   else :                  # Moves to q6 if the TP user input is
33      "False or 0"
34    movej(q6, vel = 30 , acc = 30 )

```

7.3 Thread

7.3.1 `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 |
|----------------|-----------|---------------|--|
| th_func_name | callable | - | Name of the function run by the thread |
| loop | bool | False | Flag indicates whether the thread will be repeated <ul style="list-style-type: none"> • True: Repeated calling of th_func_name (interval 0.01second) • False: One-time calling of th_func_name |

Return

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

Exception

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

| Exception | Description |
|--------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  #----- Thread -----
2  def fn_th_func():
3      if check_motion() == 0 :    # No motion in action
4          set_digital_output( 1 , OFF)
5      else :
6          set_digital_output( 1 , ON)
7
8  #----- Main routine -----
9  th_id  =  thread_run(fn_th_func, loop = True ) # Thread run
10
11 while 1 :
12     # do something...
13     wait( 0.1 )

```

7.3.2 `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 |
|----------------|-----------|---------------|-------------------|
| th_id | int | - | Thread ID to stop |

Return

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

Exception

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

Example

```

1  def fn_th_func():
2      if check_motion() == 0 :    # No motion in action
3          set_digital_output( 1 , OFF)
4      else :
5          set_digital_output( 1 , ON)
6  ----- Main routine -----
7  th_id  =  thread_run(fn_th_func, loop = True )
8
9  # do something...
10 thread_stop(th_id) # Stops the thread.

```

7.3.3 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 |

| Value | Description |
|----------------|-------------|
| Negative value | Failed |

Exception

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

Example

```

1  def fn_th_func():
2      if check_motion() == 0 :    # No motion in action
3          set_digital_output( 1 , OFF)
4      else :
5          set_digital_output( 1 , ON)
6  ----- Main routine -----
7  th_id  =  thread_run(fn_th_func, loop = True )
8
9  # do something...
10
11  thread_pause(th_id)  # Suspends the thread.

```

7.3.4 thread_resume(th_id)

Features

This function resumes a temporarily suspended thread.

Parameters

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

Return

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

Exception

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

Example

```

1  def fn_th_func():
2      if check_motion() == 0 :    # No motion in action
3          set_digital_output( 1 , OFF)
4      else :
5          set_digital_output( 1 , ON)
6
7  ----- Main routine -----
8  th_id = thread_run(fn_th_func, loop = True )
9
10 # do something...
11 thread_pause(th_id) # Suspends the thread.
12
13 # do something...
14 thread_resume(th_id) # Resumes the suspended thread.

```

7.3.5 **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

```

1  def fn_th_func():
2      if check_motion() == 0 :      # No motion in action
3          set_digital_output( 1 , OFF)
4      else :
5          set_digital_output( 1 , ON)
6
7      th_id   = thread_run(fn_th_func, loop = True )
8      state1 = thread_state(th_id)
9
10     thread_pause(th_id)
11     state2 = thread_state(th_id)

```

7.3.6 Integrated example - Thread

This example explains how to use the thread.

Example 1: Thread example

```

1  ----- thread 1: client comm. -----
2  def fn th client():
3      global g_sock
4      global g_cmd
5      res, rx_data = client_socket_read(g_sock)
6      if res > 0 :
7          g cmd = rx_data.decode() #decode: Converts byte type into a
8          string.
9          else : # Communication error
10             client socket close(g_sock)
11             exit() # Terminates the program.
12             wait( 0.1 )
13             return 0
14
15 ----- thread 2: check IO -----
16 def fn th check io():
17     if get_digital_input( 1 ) == ON:
18         exit() # Terminates the program.
19         wait( 0.1 )
20         return 0
21
22 ----- main -----
23 g_sock = client_socket_open( "192.168.137.2" , 20002 ) #
24 Connects to the server.
25 g_cmd = ""
26
27 g_th_id1 = thread_run(th_client, loop = True ) # Runs the
28 th_client thread.
29 g_th_id2 = thread_run(th_check_io, loop = True ) # Runs the
30 th_check_io thread.
31
32 p1 = posj( 0 , 0 , 90 , 0 , 90 , 0 )
33 p2 = posj( 10 , 0 , 90 , 0 , 90 , 0 )
34 p3 = posj( 20 , 0 , 90 , 0 , 90 , 0 )
35
36 while 1 :
37     if g cmd == "a" :
38         g cmd = ""
39         movei(p1,vel = 100 ,acc = 100 )
40         client_socket_write(g_sock, b "end" )
41     if g cmd == "b" :
42         g cmd = ""
43         movei(p2,vel = 100 ,acc = 100 )
44         client_socket_write(g_sock, b "end" )
45     if g cmd == "c" :
46         g cmd = ""
47         movei(p3,vel = 100 ,acc = 100 )
48         client_socket_write(g_sock, b "end" )

```

| | |
|----|--------------------------|
| 45 | <code>wait(0.1)</code> |
|----|--------------------------|

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.

7.4 Others

7.4.1 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 |

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

Example

```

1  wait( 1.3 )    # Waits for 1.3 seconds.
2
3  while 1 :    # Checks contact no. 1 every 0.1 second.
4      if get_digital_input( 1 ) == ON:
5          set_digital_output( 1 , ON)
6          wait( 0.1 )

```

7.4.2 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

```

1  exit()

```

7.4.3 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Note

- The first line of the subprogram must have the phrase "from DRCF import *".
- When programming with a teaching pendant, this phrase is automatically inserted.
- If the global variable names of the main program and subprograms are the same, they operate as different variables. Variables cannot be referenced by each other.
- If you need to share variables between the main program and subprograms, use system variables.
- System variables are set through the teaching pendant. Please refer to the user manual for detailed usage.

Example

```

1 # subprogramA and subprogramB must be created and saved in
2 # advance.
3
4 <subProgramA.drl>
5 from DRCF import *
6 movej([ 0 , 0 , 90 , 0 , 90 , 0 ], vel = 30 , acc = 30 )
7
8 <subProgramB.drl>
9 from DRCF import *
10 movej[( 10 , 0 , 90 , 0 , 90 , 0 ), vel = 30 , acc = 30 ]
11
12 <main program>
13 while True :
14     var_select = tp_get_user_input( "Select File" , DR_VAR_INT)
15     if var_select == 0 :
16         sub_program_run( "subProgramA" )      # execute subProgramA
17     elif var_select == 1 :
18         sub_program_run( "subProgramB" )      # execute subProgramB

```

7.4.4 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 ON(1) OFF(0) |

Return

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

Example

```

1 x = 0
2 y = 0
3
4 drl report line(OFF)      # Execution line display function OFF
5 while x < 1000 :          # Execution line not displayed (speed
6   up execution)
7     x += 1                 # Execution line not displayed
8   (speed up execution)
9   drl_report_line(ON)      # Execution line display function ON
10  x = 0                   # Execution line shown
11  y = 0                   # Execution line shown

```

7.4.5 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 float string | - | Interlocking Data Variable Possible data types <ul style="list-style-type: none"> • Integer data • Real data • string data |

Return

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

Example

```

1 count = 0
2
3 movei(posi( 0 , 0 , 90 , 0 , 90 , 0 ), vel = 30 , acc = 30 )
4 while True :
5     movej(posj( 0 , 0 , - 90 , 0 , 90 , 0 ), vel = 30 , acc = 3
0 )
6     movei(posi( 0 . 0 , 90 , 0 , 90 , 0 ), vel = 30 , acc = 30 )
7     count = count + 1
8     set_fm("TotalCount", count)

```

7.4.6 get_robot_model()

Features

This is a command to read the model name of the robot.

Return

| Value | Description |
|------------|--|
| model name | Returns the model name in String type. "M1013", "M0617", "M0609", "M1509", "A0307", "A0307S", "A0509", "A0509S", "A0912", "A0912S" "H2515", "H2017" |

Example

```

1   model = get_robot_model()
2
3   if model == "M1013" :
4       set_velj( 30 )
5   else :
6       set_velj( 50 )

```

7.4.7 get_robot_serial_num()

Features

This is a command to read the serial number of the robot.

Return

| Value | Description |
|---------------|---|
| serial_number | Returns the serial number in String type. Serial number: 6-character string consisting of numbers and English characters |

Example

```

1   serial_num = get_robot_serial_num()

```

7.4.8 check_robot_jts()

Features

This is a command to check whether the robot is equipped with a joint torque sensor.

Return

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

Example

```

1 if check_robot_jts() != True :
2     movej([ 0 , 0 , 90 , 0 , 90 , 0 ], 60 , 30 )
3 else :
4     movej([ 0 , 0 , 0 , 0 , 0 , 0 ], 60 , 30 )

```

7.4.9 check_robot_fts()

Features

This is a command to check whether the robot is equipped with a force torque sensor.

Return

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

Example

```

1 if check_robot_fts() != True :
2     movej([ 0 , 0 , 90 , 0 , 90 , 0 ], 60 , 30 )
3 else :

```

4

```
movej([ 0 , 0 , 0 , 0 , 0 , 0 ], 60 , 30 )
```

7.4.10 start_timer()

Features

This is a command to measure the execution time of the simulation program of the controller. When used with the end_timer() command, it returns the execution time of the script between the two functions.

⚠ Caution

This function is for measuring motion execution time in Windows environment and Linux environment. When measuring in Real mode in an emulator (virtual controller) environment, incorrect values may be returned.

Return

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

Example

```
1 start_timer()
2 wait( 1 )
3 t = end_timer()
4 tp_log( "tttt={0} sec" . format (t))
```

Related Commands

- [end_timer\(\)](#)(p. 273)

7.4.11 end_timer()

Features

This is a command to measure the execution time of the simulation program of the controller. When used with the start_timer() command, it returns the execution time of the script between the two functions.

⚠ Caution

This function is for measuring motion execution time in Windows environment and Linux environment. When measuring in Real mode in an emulator (virtual controller) environment, incorrect values may be returned.

Return

| Value | Description |
|-------|--|
| float | Measured time information (process execution time) |

Example

```
1 start_timer()
2 wait( 1 )
3 t = end_timer()
4 tp_log( "tttt={0} sec" . format (t))
```

Related Commands

- [start_timer\(\)](#)(p. 273)

8 Mathematical Function

8.1 Basic Function

8.1.1 ceil(x)

Features

This function returns the smallest integer value of integers equal to or larger than x. It truncates up to the integer.

Parameters

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

Return

| Value | Description |
|-----------------|-------------|
| rounded integer | - |

Exception

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

8.1.2 floor(x)

Features

This function returns the largest integer value of integers equal to or smaller than x. It rounds down to the nearest one.

Parameters

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

Return

| Value | Description |
|-----------------|-------------|
| rounded integer | - |

Exception

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

8.1.3 pow(x, y)

Features

Return x raised to the power of y.

Parameters

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

Return

| Value | Description |
|-------------------------|-------------|
| x raised to the power y | - |

Exception

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

8.1.4 sqrt(x)

Features

This function returns the square root of x.

Parameters

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

Return

| Value | Description |
|----------------------|-------------|
| the square root of x | Success |

Exception

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

8.1.5 log(x, b)

Features

This function returns the log of x with base b.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-----------------------------|
| x | float | - | - |
| b | float | - | base, e (natural logarithm) |

Return

| Value | Description |
|--------------------------------------|-------------|
| the logarithm of f to the base of b. | - |

Exception

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

8.1.6 d2r(x)

Features

This function returns the x degrees value to radians.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|----------------------|
| x | float | - | The angle in degrees |

Return

| Value | Description |
|----------------------|-------------|
| The angle in radians | - |

Exception

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

8.1.7 r2d(x)

Features

This function returns the x radians value to degrees.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|----------------------|
| x | float | | The angle in radians |

Return

| Value | Description |
|----------------------|-------------|
| The angle in degrees | - |

Exception

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

8.1.8 random()

Features

This function returns a random number between 0 and 1.

Return

| Value | Description |
|---------------|---------------------------------------|
| random number | Random number between 0 and 1 (float) |

Exception

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

8.2 Trigonometric functions

8.2.1 sin(x)

Features

This function returns the sine value of x radians.

Parameters

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

Return

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

Exception

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

8.2.2 cos(x)

Features

This function returns the sine value of x radians.

Parameters

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

Return

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

Exception

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

8.2.3 tan(x)

Features

This function returns the tangent value of x radians.

Parameters

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

Return

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

Exception

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

8.2.4 asin(x)

Features

This function returns the arc sine value of x radians.

Parameters

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

Return

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

Exception

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

8.2.5 `acos(x)`

Features

This function returns the arc cosine value of x radians.

Parameters

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

Return

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

Exception

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

8.2.6 `atan(x)`

Features

This function returns the arc tangent value of x radians.

Parameters

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

Return

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

Exception

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

8.2.7 atan2(y, x)

Features

This function returns the arc tangent value of y/x radians.

Parameters

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

Return

| Value | Description |
|------------------------|----------------------------------|
| the arc tangent of y/x | The result is between -pi and pi |

Exception

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

8.3 Linear algebra

8.3.1 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 |

8.3.2 rotx(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the x-axis.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|----------------------|
| angle | float | 0 | Rotating angle [deg] |

Return

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

Exception

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

Example

| | |
|--|--------------------------|
| | 1 rotm = rotx(30) |
|--|--------------------------|

8.3.3 roty(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the y-axis.

Parameters

| Parameter Name | Data type | Default Value | Description |
|-----------------------|------------------|----------------------|----------------------|
| angle | float | 0 | Rotating angle [deg] |

Return

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

Exception

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

Example

```
1  rotm = roty( 30 )
```

8.3.4 rotz(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the z-axis.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|----------------------|
| angle | float | 0 | Rotating angle [deg] |

Return

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

Exception

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

Example

```
1 rotm = rotz( 30 )
```

8.3.5 rotm2eul(rotm)

Features

This function receives a rotation matrix and returns the Euler angle (zyz order) to degrees. Of the Euler angle (rx, ry, rz) returned as a result, ry is always a positive number.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-------------|---------------|-----------------|
| rotm | Float[3][3] | - | Rotation matrix |

Return

| Value | Description |
|----------|-----------------------|
| float[3] | ZYZ Euler angle [deg] |

Exception

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

Example

```
1 rotm = [[ 1 , 0 , 0 ],[ 0 , 0.87 , - 0.5 ],[ 0 , 0.5 , 0.87 ]]
2 eul = rotm2eul(rotm)
```

8.3.6 rotm2rotvec(rotm)

Features

This function receives a rotation matrix and returns the rotation vector (angle/axis representation).

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-------------|---------------|-----------------|
| rotm | float[3][3] | - | Rotation matrix |

Return

| Value | Description |
|----------|-----------------|
| float[3] | rotation vector |

Exception

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

Example

```

1  rotm = [[ 1 , 0 , 0 ], [ 0 , 0.87 , - 0.5 ], [ 0 , 0.5 , 0.87 ]]
2  eul = rotm2rotvec(rotm)

```

8.3.7 eul2rotm([alpha,beta,gamma])

Features

This function transforms a Euler angle (zyz order) to a rotation matrix.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------------------|
| eul | float[3] | [0 0 0] | Euler angle (zyz) [deg] |

Return

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

Exception

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

Example

```

1   eul  = [ 90 , 90 , 0 ]
2   rotm = eul2rotm (eul)

```

8.3.8 eul2rotvec([alpha,beta,gamma])

Features

This function transforms a Euler angle (zyz order) to a rotation vector.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------------------|
| eul | float[3] | [0 0 0] | Euler angle (zyz) [deg] |

Return

| Value | Description |
|--------------|--------------------|
| float[3] | rotation vector |

Exception

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

Example

```

1  eul  = [ 90 , 90 , 0 ]
2  rotvec = euler2rotvec (eul)

```

8.3.9 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

```

1 rotvec = [ 0.7854 , 0 , 0 ]
2 eul   = rotvec2eul(rotvec) # eul=[45,0,0]

```

8.3.10 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

```
1 rotm = rotvec2eul([ 0.7854 , 0 , 0 ])
```

8.3.11 htrans(posx1,posx2)

Features

This function returns the pose corresponding to $T_1 \cdot T_2$ assuming that the homogeneous transformation matrices obtained from posx1 and posx2 are T_1 and T_2 , respectively.

$$H_1 H_2 = \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & r_1 + R_1 r_2 \\ 0 & 1 \end{bmatrix}$$

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|-------|-------------|
| posx | [mm, deg] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 posx = htrans(posx1,posx2)

```

8.3.12 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

| Parameters Name | Data Type | Default Value | Description |
|-----------------|----------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |
| alpha | float | - | $0.0 \leq \text{alpha} \leq 1.0$ |

Return

| Value | Description |
|-------|-------------|
| posx | [mm, deg] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 alpha = 0.5
4 posx = get_intermediate_pose(posx1,posx2,alpha)

```

8.3.13 get_distance(posx1, posx2)

Features

This function returns the distance between two pose positions in [mm].

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|-------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm] |
| posx2 | posx list (float[6]) | - | posx or position list [mm] |

Return

| Value | Description |
|-------|-------------|
| float | [mm] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 dis_posx = get_distance(posx1, posx2)

```

8.3.14 get_normal(x1, x2, x3)

Features

This function returns the normal vector of a surface consisting of three points (posx) in the task space. This direction is clockwise.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|--------------------------|
| x1 | posx list (float[6]) | - | posx or position list |
| x2 | posx list (float[6]) | - | posx or position list |
| x3 | posx list (float[6]) | - | posx or position list |

Return

| Value | Description |
|----------|---------------|
| float[3] | normal vector |

Exception

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

Example

```

1  x1 = posx( 0 , 500 , 700 , 30 , 0 , 90 )
2  x2 = posx( 500 , 0 , 700 , 0 , 0 , 45 )
3  x3 = posx( 300 , 100 , 500 , 45 , 0 , 45 )
4  vect = get_normal(x1, x2, x3)

```

8.3.15 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 list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|-------|-------------|
| posx | [mm, deg] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 add_posx = add_pose(posx1, posx2)

```

8.3.16 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 list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|-------|-------------|
| posx | [mm, deg] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 subtract_posx = subtract_pose(posx1, posx2)

```

8.3.17 inverse_pose(posx1)

Feature

This function returns the posx value that represents the inverse of posx.

$$\text{inv_pose} \cdot \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_1^T & -R_1^T r_1 \\ 0 & 1 \end{bmatrix}$$

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|-------|-------------|
| posx | [mm, deg] |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 inv_posx = inverse_pose(posx1)

```

8.3.18 dot_pose(posx1, posx2)

Features

This function obtains the inner product of the translation component when two poses are given.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|-------|-----------------------------|
| float | Inner product of two poses. |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 res = dot_pose(posx1, posx2)

```

8.3.19 cross_pose(posx1, posx2)

Features

This function obtains the outer product of the translation component when two poses are given.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |
| posx2 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|----------|-----------------------------|
| float[3] | Outer product of two poses. |

Exception

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

Example

```

1 posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2 posx2 = [ 200 , 50 , 100 , 90 , 30 , 150 ]
3 res = cross_pose(posx1, posx2)

```

8.3.20 unit_pose(posx1)

Features

This function obtains the unit vector of the given posx translation component.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------------------|---------------|------------------------------------|
| posx1 | posx list (float[6]) | - | posx or position list [mm, deg] |

Return

| Value | Description |
|----------|-------------------------------|
| float[3] | Unit vector of the given posx |

Exception

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

Example

```

1  posx1 = [ 100 , 20 , 300 , 90 , 0 , 180 ]
2  res = unit_pose(posx1)

```

9 External Communication Commands

9.1 Serial

9.1.1 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 | <ul style="list-style-type: none"> D-SUB(9 pin) Connection : "COM" USB to Serial Connection : "COM_USB" |
| baudrate | int | 115200 | Baud rate 2400, 4800, 9600, 19200, 38400, 57600, 115200 |
| bytesize | int | 8 | Number of data bits <ul style="list-style-type: none"> DR_FIVEBITS: 5 DR_SIXBITS: 6 DR_SEVENBITS: 7 DR_EIGHTBITS: 8 |
| parity | str | "N" | Parity checking <ul style="list-style-type: none"> DR_PARITY_NONE: "N" DR_PARITY EVEN: "E" DR_PARITY ODD: "O" DR_PARITY MARK: "M" DR_PARITY SPACE: "S" |
| stopbits | int | 1 | Number of stop bits <ul style="list-style-type: none"> DR_STOPBITS_ONE =1 DR_STOPBITS_ONE_POINT_FIVE = 1.5 DR_STOPBITS_TWO =2 |

[Return](#)

| Value | Description |
|------------------------|-----------------------|
| serial.Serial instance | Successful connection |

[Exception](#)

| Exception | Description |
|---------------------------------|---------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | Serial.SerialException error occurred |

[Example](#)

```

1 # When connected to serial port D-SUB (9 pin)
2 ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = DR_EIGHTBITS,
3 parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
4
5 res = serial_write(ser, b "123ABC" )
6
7 serial_close(ser)
8
9
10 # When a USB to serial device is connected to a USB port
11 ser = serial_open(port = "COM_USB" , baudrate = 115200 ,
12 bytesize = DR_EIGHTBITS,
13 parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
14
15 res = serial_write(ser, b "123ABC" )
16
17 serial_close(ser)

```

9.1.2 serial_close(ser)

Features

This function closes a serial communication port.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|---------------|---------------|-----------------|
| ser | serial.Serial | - | Serial instance |

Return

| Value | Description |
|-------|-------------------------------------|
| 0 | Successful closing of a serial port |

Exception

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

Example

```

1  ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = D
R_EIGHTBITS ,
2  parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
3
4  res = serial_write(ser, b "123456789" )
5
6  serial_close(ser)

```

9.1.3 serial_state(ser)

Featuers

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

```

1  ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = D
R_EIGHTBITS ,
2      parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
3
4  state = serial_state(ser)
5
6  serial_close(ser)

```

9.1.4 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 |
|----------------|---------------|---------------|--|
| ser | serial.Serial | - | Serial instance |
| timeout | float | None | Timeout between bytes during reading or writing <ul style="list-style-type: none"> • Continued processing of data that was processed before the timeout • None: inter-byte timeout not specified |

Return

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

Exception

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

Example

```

1  ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = DR_EIGHTBITS ,
2      parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
3
4  res = serial_set_inter_byte_timeout(ser, 0.1 )
5
6  serial_close(ser)

```

9.1.5 serial_write(ser, tx_data)

Features

This function writes the data (tx_data) to a serial port.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|---------------|---------------|---|
| ser | serial.Serial | - | Serial instance |
| tx_data | byte | - | Data to be transmitted <ul style="list-style-type: none"> • The data type must be a byte. • Refer to the example below. |

Return

| Value | Description |
|-------|---------------------------------------|
| 0 | Success |
| -1 | The port is not open. |
| -2 | serial.SerialException error occurred |

Exception

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

Example

```

1  ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = DR_EIGHTBITS,
2                      parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
3
4  serial_write(ser, b "123456789" ) # b means the byte type.
5
6  # Convert string to byte
7  msg = "abcd"                      # msg is a string variable
8  serial_write(ser, msg.encode())    # encode() converts string type
9  to byte type
10
11 serial_close(ser)

```

9.1.6 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 |
|----------------|---------------|---------------|---|
| Ser | serial.Serial | - | Serial instance |
| Length | int | -1 | Number of bytes to read <ul style="list-style-type: none"> • -1: Not specified(The number of bytes to read is not specified) • n(>=0): The specified number of byte is read. |
| timeout | int float | -1 | Read waiting time <ul style="list-style-type: none"> • -1: Indefinite wait • n(>0): n seconds |

Return

| Value(res, rx_data) | | Description |
|---------------------|----|---------------------------------------|
| res | n | Number of bytes of the received data |
| | -1 | The port is not open. |
| | -2 | serial.SerialException error occurred |
| rx_data | | Number of bytes read (byte type) |

Exception

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

| Exception | Description |
|------------------------------|----------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |

Example

```

1  ser = serial_open(port = "COM" , baudrate = 115200 , bytesize = D
R_EIGHTBITS,
2      parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)
3
4  res, rx_data = serial_read(ser)
5  #Wait indefinitely until data is received
6
7  res, rx_data = serial_read(ser, timeout = 3 )
8  #Wait until data is received, set a 3 seconds timeout
9  # If received within 3 seconds, the read data is returned
immediately
10 # Return the value read so far after 3 seconds have elapsed
11
12 res, rx_data = serial_read(ser, length = 100 )
13 # Wait indefinitely until reading 100 bytes
14
15 res, rx_data = serial_read(ser, length = 100 , timeout = 3 )
16 #Wait until reading 100byte, set 3 seconds timeout
17 # If 100 bytes are received within 3 seconds, the read data is
returned immediately.
18 # Return the value read so far after 3 seconds have elapsed
19
20 # Convert the received byte type to string type
21 rx_msg = rx_data.decode()
22 # rx data is a byte type and decode() is used to convert it to a
string type.
23 # For example, if rx_data = b"abcd", then rx_msg="abcd".
24
25 res, rx_data = serial_close(ser)

```

9.1.7 serial_get_count()

Features

This function reads the number of devices connected to USB to Serial.

Return

| Value (port_info, device_name) | Description |
|--------------------------------|----------------------------------|
| count | Number of connected serial ports |

Exception

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

Example

```

1 count = serial_get_count() # read number of connected serial
2 ports
3
4 for i in range(count):
5     port_info, device_name = serial_get_info(i + 1)
6     tp_popup( "i={}, port ={}, dev ={}".format(i, port_info,
device_name))

```

9.1.8 serial_get_info(id)

Features

This function reads the port information and device name of the connected USB to Serial.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--------------------------------------|
| id | int | 1 | ID of "USB to Serial" to read (1-10) |

[Return](#)

| Value (port_info, device_name) | Description |
|---------------------------------------|--|
| port_info | Port information (NULL means no device is connected) |
| device_name | Device name (NULL means no device is connected) |

[Exception](#)

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

[Example](#)

```

1 port_info, device_name = serial_get_info( 1 ) #1 connected
2 device information
3 #port_info = "COM USB"
4 ser = serial_open(port = port_info, baudrate = 115200 , bytesize =
R_EIGHTBITS,
5 parity = DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)

```

9.1.9 Combined Example - Serial

This is an example for performing a self-loop-back test on RXD (#2 pin) and TXD (#3 pin) are connected with the serial port.

Example 1 : Self-loop-back test example

```

1 # serial port open
2 # if D-SUB (9pin) is connected: port="COM"
3 # if USB is connected with USB to Serial: port="COM_USB"
4 ser = serial_open(port = "COM_USB" , baudrate = 115200 ,
bytesize = DR_EIGHTBITS, parity = DR_PARITY_NONE, stopbits = DR_ST
OPBITS_ONE)
5 wait( 1 )
6

```

```

7 # SEND DATA : "123ABC"
8 res = serial_write(ser, b "123ABC" ) # b means byte type
9 wait( 1 )
10
11 # READ DATA
12 res, rx_data = serial_read(ser)
13 # RXD and TXD are H/W connected res=6 (byte) rx_data = b"123ABC"
14 are recevied
15 tp_popup( "res ={0}, rx_data={1}" . format (res, rx_data))
16
17 # close corresponding serial port
18 serial_close(ser)

```

Received data is collected as is and the result is outputted as a TP pop-up message.

If executed properly, it outputs a result of res=6 rx_data = b'123ABC'.

Example 2 : Various packet transmission examples

Transmission packet: “MEAS_START” +data1[4byte]+data2[4byte]

data1: Converts integer to 4 bytes ex) 1 → 00000001

data2: Converts integer to 4 bytes ex) 2 → 00000002

ex) In case data1=1, data2=2: “MEAS_START”+00000001+00000002

Actual Packet: 4D4541535F53544152540000000100000002

Received packet: res=18, rx_data=“MEAS_START”+00000001+00000002

Extract rxd1 : Convert 10th to 14th byte into integer

Extract rxd2 : Convert 14th to 18th byte into integer

```

1 ser = serial_open(port = "COM_USB" , baudrate = 115200 ,
2 bytesize = DR_EIGHTBITS, parity = DR_PARITY_NONE, stopbits = DR_ST
3 OPBITS_ONE)
4 wait( 1 )
5
6 send_data = b "MEAS_START" # b means byte type
7 data1 = 1
8 data2 = 2
9 send_data += (data1).to_bytes( 4 , byteorder = 'big' )
10 send_data += (data2).to_bytes( 4 , byteorder = 'big' )
11
12 # SEND DATA
13 res = serial_write(ser, send_data)
14 wait( 1 )
15
16 # READ DATA
17 # RXD, TXD are connected by H/W, so send_data is received as it is
18 res, rx_data = serial_read(ser)

```

```

17 tp_popup( "res ={0}, rx_data={1}" . format (res, rx_data))
18
19 rxd1 = int .from_bytes(rx_data[ 10 : 10 + 4 ], byteorder = 'big'
20   , signed = True )
21 rxd2 = int .from_bytes(rx_data[ 14 : 14 + 4 ], byteorder = 'big'
22   , signed = True )
23
24 tp_popup( "res={0}, rxd1={1}, rxd2={2}" . format (res, rxd1,
25 rxd2))
26
#Close the serial port
serial_close(ser)

```

Connect the USB to serial device to the USB port and send byte type send_data.

Since RXD(2pin) and TXD(3pin) are connected to receive the transmitted data as it is,

res = 18, rx_data has the same packet as send_data.

Extract rxd1 : Convert 10th to 14th byte into integer

Extract rxd2 : Convert 14th to 18th byte into integer

The end result will be res=18, rxd1=1, rxd2=2

9.2 Tcp/Client

9.2.1 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

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2  # An indefinite connection is attempted to the server
3  (ip="192.168.137.200", port=20002).
4  # The connected socket is returned if the connection is
5  successful.
6  # The data is read, written, and closed using the returned socket
7  as shown below.
8
9  client_socket_write(sock, b "123abc" )      # Sends data to the
10 server (b represents the byte type).
11 res, rx_data = client_socket_read(sock)    # Receives the data
12 from the server.
13 client_socket_close(sock)                  # Closes the connection to
14 the server.

```

9.2.2 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

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2  # An indefinite connection is attempted to the server
3  # (ip="192.168.137.200", port=20002).
4
5  # do something...
6
7  client_socket_close(sock)           # Closes the connection to the
8  server.
```

9.2.3 client_socket_state(sock)

Features

This function returns the socket connection status. To know the connection status with the server, check the return value of client_socket_read or client_socket_write (see Example 2).

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|---------------|---------------|--|
| sock | socket.socket | - | Socket instance returned from client_socket_open() |

Return

| Value | Description |
|-------|-----------------------|
| 1 | Socket normal state |
| 0 | Socket abnormal state |

Exception

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

Example

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2
3  state = client_socket_state(sock) # Reads the socket state.
4
5  client_socket_close(sock)

```

Example 2

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2  res, rx_data = client_socket_read(sock)
3  tp_log( "[RX] res={0}, rx_data ={1}" . format (res, rx_data))
4  if (res < 0):
5      tp_log( "[RX] server disconnect" ) #When the server connection
6      is disconnected
7      client_socket_close(sock)
8      exit()
9
10 client_socket_close(sock)

```

9.2.4 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"> • The data type must be a byte. • Refer to the example below. |

Return

| Value | Description |
|-------|--|
| 0 | Success |
| -1 | The server is not connected. |
| -2 | Server is disconnected, or socket.error occurred during a data transfer |

Exception

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

Example

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2
3  client_socket_write(sock, b "1234abcd" ) # b means the byte type.
4
5  msg = "abcd" # msg is a string variable.

```

```

6   client_socket_write(sock, msg.encode()) # encode() converts a
7   string type to a byte type.
8   client_socket_close(sock)

```

9.2.5 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 client_socket_open() |
| length | int | -1 | Number of bytes of the received data <ul style="list-style-type: none"> • -1 : Not specified (The number of bytes to read is not specified.) • n(>=0) : The specified number of bytes is read. |
| timeout | int float | -1 | Waiting time for receipt <ul style="list-style-type: none"> • -1 : Indefinite wait • n(>0) : n seconds |

Return

| Value (res, rx_data) | Description | |
|----------------------|---------------------------|---|
| res | >0 | Number of bytes of the received data |
| | -1 | The server is not connected. |
| | -2 | Socket.error occurred during data reception |
| | -3 | Timeout during data reception |
| rx_data | Received data (byte type) | |

Exception

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

Example

```

1  sock = client_socket_open( "192.168.137.200" , 20002 )
2
3  res, rx_data = client_socket_read(sock)      # Indefinite wait
4  until the data is received
5  # Reads all received data since the length is omitted.
6  # Waits indefinitely until the data is received since timeout is
7  omitted.
8  # (res = size of received data, rx_data=received data) is returned
9  when the data is received.
10
11 res, rx_data = client_socket_read(sock, timeout = 3 ) # Waits
12 for up to 3 seconds until the data is received.
13 # (res = size of received data, rx_data=received data) is returned
14 if the data is received within 3 seconds.
15 # (res = -3, rx_data=None) is returned if the data is not received
16 within 3 seconds.
17
18 res, rx_data = client_socket_read(sock, length = 64 )      # Reads
19 64 bytes of the received data.
20
21 res, rx_data = client_socket_read(sock, length = 64 , timeout = 3
22 # Reads 64 bytes of the received data within the 3-second timeout.
23
24 rx_msg = rx_data.decode() # rx_data is a byte type and can be
25 converted to a string type
26
27 # using decode().
28 # For example, if rx_data = b"abcd",
29 # rx_msg= "abcd".
30
31 client_socket_close(sock)

```

9.2.6 Integrated example (Tcp/Client)

Assume that server IP = 192.168.137.200 and open port =20002

and that the received packets are sent to the client as they are (mirroring).

Example 1 : Example of a default TCP client

```

1 # Assume server IP = 192.168.137.200 and open port =20002.
2 g_sock = client_socket_open( "192.168.137.200" , 20002 )
3
4 tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
5 while 1 :
6     client_socket_write(g_sock, b "abcd" ) # The string "abcd" is
7     sent in a byte type.
8     wait( 0.1 )
9     res, rx_data = client_socket_read(g_sock) # Waits for the data
10    from the server.
11    tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data) ,
12 DR_PM_MESSAGE)
13    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

```

1 g_sock = client_socket_open( "192.168.137.100" , 20002 )
2 tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
3
4 send_data = b "MEAS_START"
5 data1 = 1
6 data2 = 2
7 send_data += (data1).to_bytes( 4 , byteorder = 'big' )
8 send_data += (data2).to_bytes( 4 , byteorder = 'big' )
9
10 client_socket_write(g_sock, send_data)
11
12 wait( 0.1 )

```

```

13     res, rx_data = client_socket_read(g_sock)
14     tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data),
15     DR_PM_MESSAGE)
16
17     rxd1 = int .from_bytes(rx_data[ 10 : 10 + 4 ], byteorder = 'big'
18     ' , signed = True )
18     rxd2 = int .from_bytes(rx_data[ 14 : 14 + 4 ], byteorder = 'big'
19     ' , signed = True )
19
20     tp_popup( "res={0}, rxd1={1}, rxd2={2}" . format (res, rxd1,
21     rxd2), DR_PM_MESSAGE)
21
22     client_socket_close(g_sock)

```

The example connects to the server and sends a byte type send_data.

res=18 and rx_data=send_data since the server transmits the received data as is.

- rxd1 extraction: Conversion of 10th - 14th bytes to an integer
- rxd2 extraction: Conversion of 14th - 18th bytes to an integer

The final result is res=18, rxd1=1, and rxd2=2.

Example 3 : Reconnection

```

1 def fn_reconnect():
2     global g_sock
3     client_socket_close(g_sock)
4     g_sock = client_socket_open( "192.168.137.200" , 20002 )
5     return
6
7     g_sock = client_socket_open( "192.168.137.200" , 20002 )
8     tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
9
10    client_socket_write(g_sock, b "abcd" )
11    wait( 0.1 )
12
13    while 1 :
14        res, rx_data = client_socket_read(g_sock)
15        if res < 0 :
16            fn_reconnect()
17        else :
18            tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data),
19            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.

9.3 Tcp/Server

9.3.1 server_socket_open(port)

Features

The robot controller creates a server socket and waits for the connection to the client. Returns the connected socket when the client is connected.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---------------------|
| port | int | - | Port number to open |

Return

| Value | Description |
|------------------------|-----------------------|
| socket.socket instance | Successful connection |

Exception

| Exception | Description |
|--------------------------------|---|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIME) | socket.error occurred during a connection |

Example

```
1   sock = server_socket_open( 20002 )
```

```

2 # Opens the port 20002 and waits until the client connects.
3 # The connected socket is returned if the connection is
4 successful.
5 # The data is read, written, and closed using the returned socket
6 as shown below.
7
8 server_socket_write(sock, b "123abc") # Sends data to the client
9 (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.

```

9.3.2 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() 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

```

1  sock = server_socket_open( 20002 )
2  # Opens the port 20002 and waits until the client connects.
3
4  # do something...
5
6  server_socket_close(sock) ) # Closes the connection to the
client.

```

9.3.3 server_socket_state(sock)

Features

This function returns the socket status.

To know the connection status with the client, check the return value of server_socket_read or server_socket_write (see Example 2).

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|---------------|---------------|--|
| sock | socket.socket | - | Socket instance returned from server_socket_open() socket instance |

Return

| Value | Description |
|-------|-----------------------|
| 1 | Socket normal state |
| 0 | Socket abnormal state |

Exception

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

Example 1

```

1  sock = server_socket_open( 20002 )
2
3  state = server_socket_state(sock) # Reads the socket state.
4
5  server_socket_close(sock)

```

Example 2

```

1  sock = server_socket_open( 20002 )
2
3  res, rx_data = server_socket_read(sock)
4  tp_log( "[RX] res={0}, rx_data ={1}" . format (res, rx_data))
5  if (res < 0): #When the client connection is disconnected
6      tp_log( "[RX] client disconnect" )
7      server_socket_close(sock)
8      exit()
9
10 server_socket_close(sock)

```

9.3.4 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() socket instance |
| tx_data | byte | - | Data to be transmitted <ul style="list-style-type: none"> • The data type must be a byte. • Refer to the example below. |

Return

| Value | Description |
|-------|--|
| 0 | Success |
| -1 | The client is not connected. |
| -2 | client is disconnected, or socket.error occurred during a data transfer |

Exception

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

Example

```

1  sock = server_socket_open( 20002 )
2
3  server_socket_write(sock, b "1234abcd" ) # b means the byte type.
4
5  msg = "abcd"      # msg is a string variable.
6  server_socket_write(sock, msg.encode()) # encode() converts a
7  # string type to a byte type.
8  server_socket_close(sock)

```

9.3.5 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() socket instance |
| length | int | -1 | Number of bytes of the received data <ul style="list-style-type: none"> • -1 : Not specified (The number of bytes to read is not specified.) • n(>=0) : The specified number of bytes is read. |
| timeout | int float | -1 | Waiting time for receipt <ul style="list-style-type: none"> • -1 : Indefinite wait • n(>0) : n seconds |

Return

| Value (res, rx_data) | Description |
|----------------------|---|
| res | 0 Number of bytes of the received data |
| | -1 The client is not connected. |
| | -2 socket.error occurred during data reception |
| | -3 Timeout during data reception |
| rx_data | Received data (byte type) |

Example

```

1  sock = server_socket_open( 20002 )
2
3  res, rx_data = server_socket_read(sock)    # Indefinite wait
4  until the data is received
5  # Reads all received data since the length is omitted.
6  # Waits indefinitely until the data is received since timeout is
7  omitted.
8  # (res = size of received data, rx_data=received data) is returned
9  when the data is received.

```

```

8   res, rx_data = server_socket_read(sock, timeout = 3) # Waits
9     for up to 3 seconds until the data is received.
10    # (res = size of received data, rx_data=received data) is returned
11    if the data is received within 3 seconds.
12    # (res = -3, rx_data=None) is returned if the data is not received
13    within 3 seconds.
14
15    res, rx_data = server_socket_read(sock, length = 64) # Reads
16      64 bytes of the received data.
17
18    res, rx_data = server_socket_read(sock, length = 64, timeout = 3)
19      # Reads 64 bytes of the received data within the 3-second timeout.
20
21    rx_msg = rx_data.decode() # rx_data is a byte type and can be
22      converted to a string type
23          # using decode().
24          # For example, if rx_data = b"abcd",
25          # rx_msg= "abcd".
26
27    server_socket_close(sock)

```

9.3.6 Integrated example - Tcp/Server

The example assumes that the client connects to the controller with IP = 192,168,137.100 and port = 20002 and that the received packets are sent to the server as they are (mirroring).

Example 1 : Default TCP server example

```

1 g_sock = server_socket_open( 20002 )
2 tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
3
4 while 1 :
5   server_socket_write(g_sock, b "abcd") # The string "abcd" is
6     sent in a byte type.
7   wait( 0.1 )
8   res, rx_data = server_socket_read(g_sock) # Waits for the data
9     from the server.
10  tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data),
11    DR_PM_MESSAGE)
12  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

```

1  g_sock = server_socket_open( 20002 )
2  tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
3
4  send_data = b "MEAS_START"
5  data1 = 1
6  data2 = 2
7  send_data += (data1).to_bytes( 4 , byteorder = 'big' )
8  send_data += (data2).to_bytes( 4 , byteorder = 'big' )
9
10 server_socket_write(g_sock, send_data)
11
12 wait( 0.1 )
13
14 res, rx_data = server_socket_read(g_sock)
15 tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data),
16 DR_PM_MESSAGE)
17
18 rxd1 = int .from_bytes(rx_data[ 10 : 10 + 4 ], byteorder = 'big'
19 ' , signed = True )
20 rxd2 = int .from_bytes(rx_data[ 14 : 14 + 4 ], byteorder = 'big'
21 ' , signed = True )
22
23 tp_popup( "res={0}, rxd1={1}, rxd2={2}" . format (res, rxd1,
rxd2), DR_PM_MESSAGE)
24
25 server_socket_close(g_sock)
```

The example sends the byte type send_data.

res = 18 and rx_data=send_data since the client transmits the received data as is.

rxd1 extraction: Conversion of 10th - 14th bytes to an integer

rxd2 extraction: Conversion of 14th - 18th bytes to an integer

The final result is res=18, rxd1=1, and rxd2=2.

Example 3 : Reconnection

```

1 def fn_reopen():
2     global g_sock
3     server_socket_close(g_sock)
4     g_sock = server_socket_open( 20002 )
5     return
6
7 g_sock = server_socket_open( 20002 )
8 tp_popup( "connect O.K!" ,DR_PM_MESSAGE)
9
10 server_socket_write(g_sock, b "abcd" )
11 wait( 0.1 )
12
13 while 1 :
14     res, rx_data = server_socket_read(g_sock)
15     if res < 0 :
16         fn_reopen()
17     else :
18         tp_popup( "res={0}, rx_data ={1}" . format (res, rx_data),
19 DR_PM_MESSAGE)
20         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.

9.4 Modbus

9.4.1 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 <ul style="list-style-type: none"> • DR_MODBUS_DIG_INPUT • DR_MODBUS_DIG_OUTPUT • DR_MODBUS_REG_INPUT • DR_MODBUS_REG_OUTPUT |
| index | int | - | Modbus signal index |
| value | int | 0 | Output when the type is DR_COIL or DR_HOLDING_REGISTER (ignored otherwise) |
| slaveid | int | 255 | <ul style="list-style-type: none"> • Slave ID of the ModbusTCP module (0 or 1-247 or 255) 0 : Broadcast address 255 : Default value for ModbusTCP |

Return

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

Exception

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

Example

```

1 # An example of connecting two Modbus IO and allocating the
2 contacts
3 # Modbus IO 1 : IP address 192.168.137.254, port 502
4 Input Register Address 0 ~ 3 , signal name "input1"~"input4"
5 Holding Register Addreess 0 ~ 3 , signal name
6 "output1"~"output4"
7 # Modbus IO 2 : IP address 192.168.137.253, port 502
8 Input Register Address 0 ~ 3 , signal name "input1"~"input4"
9 Holding Register Addreess 0 ~ 3 , signal name
10 "output1"~"output4"
11
12 # set < Modbus IO 1 > "input1"~"input4", "output1"~"output4"
13 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "inp
14 ut1" , reg_type = DR_INPUT_REGISTER, index = 0 , slaveid = 255 )
15 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "inp
16 ut2" , reg_type = DR_INPUT_REGISTER, index = 1 , slaveid = 255 )
17 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "inp
18 ut3" , reg_type = DR_INPUT_REGISTER, index = 2 , slaveid = 255 )
19 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "inp
20 ut4" , reg_type = DR_INPUT_REGISTER, index = 3 , slaveid = 255 )

21
22 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "out
23 put1" , reg_type = DR_HOLDING_REGISTER, index = 0 , value = 0 ,
24 slaveid = 255 )
25 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "out
26 put2" , reg_type = DR_HOLDING_REGISTER, index = 1 , value = 0 ,
27 slaveid = 255 )

```

```

17 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "output3" , reg_type = DR_HOLDING_REGISTER, index = 2 , value = 0 , slaveid = 255 )
18 add_modbus_signal(ip = "192.168.137.254" ,port = 502 , name = "output4" , reg_type = DR_HOLDING_REGISTER, index = 3 , value = 0 , slaveid = 255 )
19
20 # set < Modbus IO 1 > "input5"~"input8", "output5"~"output8"
21 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "input5" ,
22 reg_type = DR_INPUT_REGISTER, index = 0 , slaveid = 255 )
23 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "input6" ,
24 reg_type = DR_INPUT_REGISTER, index = 1 , slaveid = 255 )
25 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "input7" ,
26 reg_type = DR_INPUT_REGISTER, index = 2 , slaveid = 255 )
27 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "input8" ,
28 reg_type = DR_INPUT_REGISTER, index = 3 , slaveid = 255 )
29
30 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "output5" , reg_type = DR_HOLDING_REGISTER, index = 0 , value = 0 , slaveid = 255 )
31 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "output6" , reg_type = DR_HOLDING_REGISTER, index = 1 , value = 0 , slaveid = 255 )
32 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "output7" , reg_type = DR_HOLDING_REGISTER, index = 2 , value = 0 , slaveid = 255 )
33 add_modbus_signal(ip = "192.168.137.253" ,port = 502 , name = "output8" , reg_type = DR_HOLDING_REGISTER, index = 3 , value = 0 , slaveid = 255 )

```

9.4.2 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 | Slave ID of the ModbusRTU (0 or 1-247) 0 : Broadcast address |
| port | string | None | E.g.) "COM", "COM_USB", "/dev/ttyUSB0" |
| baudrate | int | 115200 | Baud rate 9600, 19200, 57600, 115200, etc |
| bytesize | int | 8 | Number of data bits <ul style="list-style-type: none"> • DR_FIVEBITS: 5 • DR_SIXBITS: 6 • DR_SEVENBITS: 7 • DR_EIGHTBITS: 8 |
| parity | string | "N" | Parity checking <ul style="list-style-type: none"> • DR_PARITY_NONE: "N" • DR_PARITY EVEN: "E" • DR_PARITY ODD: "O" |
| stopbits | int | 1 | Number of stop bits <ul style="list-style-type: none"> • DR_STOPBITS_ONE =1 • DR_STOPBITS_TWO =2 |
| name | string | - | Modbus signal name |
| reg_type | int | - | Modbus signal type <ul style="list-style-type: none"> • DR_DISCRETE_INPUT =DR_MODBUS_DIG_INPUT • DR_COIL =DR_MODBUS_DIG_OUTPUT • DR_INPUT_REGISTER =DR_MODBUS_REG_INPUT • DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT |
| index | int | - | Modbus signal index |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| value | int | 0 | Output when the type is DR_COIL or DR_HOLDING_REGISTER (ignored otherwise) |

Return

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

Exception

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

Example

```

1 add modbus rtu signal(slaveid = 1 , port = port info, baudrate = 1
15200 , bvtsize = DR_EIGHTBITS, parity = DR_PARITY_NONE, stopbits =
R_STOPBITS_ONE, name = 'di1', reg_type = DR_INPUT_REGISTER, index = 6
2
3 add modbus rtu signal(slaveid = 1 , port = port info, baudrate = 1
15200 , bvtsize = DR_EIGHTBITS, parity = DR_PARITY_NONE, stopbits =
R_STOPBITS_ONE, name = 'do1', reg_type = DR_HOLDING_REGISTER,
index = 0 , value = 12345 )

```

9.4.3 add_modbus_signal_multi(ip, port, slaveid=255, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1)

Features

This function registers the ModbusTCP FC15 & FC16 multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

 **Note**

Initial value setting function is not supported.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------------|--|
| ip | string | - | IP address of the ModbusTCP module |
| port | int | - | Port number of the ModbusTCP module |
| slaveid | int | 255 | <ul style="list-style-type: none"> Slave ID of the ModbusTCP module (0 or 1-247 or 255) 0 : Broadcast address 255 : Default value for ModbusTCP |
| name | string | None | Modbus signal name |
| reg_type | int | DR_HOLDING_REGISTER | Modbus signal type <ul style="list-style-type: none"> DR_COIL =DR_MODBUS_DIG_OUTPUT DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT |
| start_address | int | 0 | Start address of Modbus multiple signal |
| cnt | int | 1 | Count of Modbus multiple signal |

Return

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

Exception

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

Example

```
1 add_modbus_signal_multi(ip = "192.168.137.200" , port = 502 ,
slaveid = 255 , name = "multi" , reg_type = DR_HOLDING_REGISTER,
start_address = 0 , cnt = 5 )
```

9.4.4 add_modbus_rtu_signal_multi(slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1)

Features

This function registers the ModbusRTU FC15 & FC16 multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

 **Note**

Initial value setting function is not supported.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| slaveid | int | 1 | Slave ID of the ModbusRTU (0 or 1-247) 0 : Broadcast address |
| port | string | None | E.g.) "COM", "COM_USB", "/dev/ttyUSB0" |
| baudrate | int | 115200 | Baud rate 9600, 19200, 57600, 115200, etc |
| bytesize | int | 8 | Number of data bits <ul style="list-style-type: none">• DR_FIVEBITS: 5• DR_SIXBITS: 6• DR_SEVENBITS: 7• DR_EIGHTBITS: 8 |
| parity | string | "N" | Parity checking <ul style="list-style-type: none">• DR_PARITY_NONE: "N"• DR_PARITY EVEN: "E"• DR_PARITY ODD: "O" |
| stopbits | int | 1 | Number of stop bits <ul style="list-style-type: none">• DR_STOPBITS_ONE =1• DR_STOPBITS_TWO =2 |
| name | string | - | Modbus signal name |
| reg_type | int | - | Modbus signal type <ul style="list-style-type: none">• DR_COIL=DR_MODBUS_DIG_OUTPUT• DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT |
| start_address | int | 0 | Start address of Modbus multiple signal |
| cnt | int | 1 | Count of Modbus multiple signal |

Return

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

Exception

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

Example

| | |
|---|---|
| 1 | add modbus rtu signal multi(slaveid = 1, port = "COM", baudrate = 15200, bytesize = DR_EIGHTBITS, parity = DR_PARITY_NONE, stopbits = R_STOPBITS_ONE, name = "multi", reg_type = DR_HOLDING_REGISTER, start_address = 0, cnt = 5) |
|---|---|

9.4.5 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Use the following command when the Modbus IO signals are
2 # registered as "input1" and "output1"
3 # and this signal registration is to be deleted. .
4 del_modbus_signal( "input1" )           # Deletes the registered
  "input1" contact
5 del_modbus_signal( "output1" )          # Deletes the registered
  "output1" contact

```

9.4.6 del_modbus_signal_multi(name)

Features

This function deletes the registered Modbus multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| name | string | - | Name of the registered Modbus multiple signal |

Return

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

Exception

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

Example

```

1 # Use the following command when the Modbus multiple signal is
2 registered as "multi1"(cnt=5)
3 # and this signal registration is to be deleted. .
del_modbus_signal_multi( "multi1" )      # Deletes the registered
"multi1" contact

```

9.4.7 set_modbus_output(iobus, value)

Features

This function sends the signal to an external Modbus system.

Function Code 05 Write Single Coil Register

Function Code 06 Write Single Holding Register

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| iobus | string | - | modbus name (set in the TP) |
| value | int | - | Value for Modbus coil register. <ul style="list-style-type: none"> • ON:1 • OFF:0 |
| | | | Value for Modbus holding register |

Note

When registered as a multiple signal, it is available by adding address value index to signal name.

Return

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

Exception

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

Example

```

1 #Modbus Coil Registers are registered as "do1" and "do2".
2 set_modbus_output( "do1" , ON)
3 set_modbus_output( "do2" , OFF)
4
5 #Modbus Holding Registers are registered as "reg1" and "reg2".
6 set_modbus_output( "reg1" , 10 )
7 set_modbus_output( "reg2" , 24 )
8
9 #Modbus multiple signal is registered as “multi”(start address=10,
10 cnt=2).
11 #“multi_10” & “multi_11” abailable
12 set_modbus_output( "multi_10" , 24 )
set_modbus_output( "multi_11" , 65535 )

```

9.4.8 set_modbus_outputs(iobus_list, val_list)

Features

This function sends multiple signals to the Modbus Slave unit.

The maximum number of outputs is 32.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-----------------------------|
| iobus | string | - | Modbus name (set in the TP) |
| 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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Modbus digital I/O output contact "d1" OFF, "d2" ON, and "d3" ON
2 set_modbus_outputs(iobus_list = [ "d1" , "d2" , "d3" ,], 
3 val_list = [ 0 , 1 , 1 ])
4 # Modbus digital I/O output contact "d3" OFF and "d4" ON

```

| | |
|---|--|
| 5 | set_modbus_outputs(["d3" , "d4"], [0 , 1]) |
|---|--|

9.4.9 set_modbus_output_multi(iobus, val_list)

Features

This function sends the signal to an external Modbus system.

Function Code 15 Write Multiple Coil Register

Function Code 16 Write Multiple Holding Register

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| iobus | string | - | Modbus multiple signal name (set in the TP) |
| Val_list | list | | Value list of modbus multiple signal |

(i) Note

An error occurs if the number of signals registered in the multiple signal name does not match the number of elements in the output value list.

Return

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

Exception

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

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

Example

```

1 #Modbus Coil Registers are registered as "do1"(cnt=5),
2   "do2"(cnt=3)
3   set_modbus_output_multi( "do1" , [ON, OFF, ON, ON, ON])
4   set_modbus_output_multi( "do2" , [ON, ON, ON])
5
6 #Modbus Holding Registers are registered as "reg1"(cnt=5),
7   "reg2"(cnt=3)
8   set_modbus_output_multi( "reg1" , [ 10 , 101 , 12345 , 777 ,
9     555 ])
10  set_modbus_output_multi( "reg2" , [ 24 , 25 , 26 ])

```

9.4.10 get_modbus_input(iobus)

Features

This function reads the signal from the Modbus Slave unit. Parameters

Parameters

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

ⓘ Note

When registered as a multiple signal, it is available by adding address value index to signal name.

Return

| Value | Description |
|-----------------|--|
| 0 or 1 or value | Modbus Descrete / Coil Register: ON or OFF Modbus Input / Holding Register : Register value |

Exception

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

Example

```

1 #Modbus digital I/O is connected, and the signals are registered
2 as "di1" and "di2".
3 get_modbus_input( "di1" )
4 get_modbus_input( "di2" )
5
6 #Modbus analog I/O is connected, and the signals are registered as
7 "reg1" and "reg2".
8 get_modbus_input( "reg1" )
9 get_modbus_input( "reg2" )
10
11 #Modbus multiple signal is registered as "multi"(start address=10,
12 cnt=2).
12 "#multi_10" & "multi_11" abailable
13 get_modbus_input( "multi_10" )
14 get_modbus_input( "multi_11" )

```

9.4.11 get_modbus_inputs(iobus_list)

Features

This function reads multiple signals from the Modbus Slave unit.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|---------------|---|
| iobus_list | list(string) | - | Modbus input name list (set in the TP) signal type can be used only in the following cases <ul style="list-style-type: none"> • DR_MODBUS_DIG_INPUT • DR_MODBUS_DIG_OUTPUT |

Return

| Value | Description |
|-----------------|--|
| int (>=0) | Multiple signals to be read at once (the value of the combination of iobus_list where the first value is LSB and the last value is MSB) |
| Negative number | Failed |

Exception

| Exception | Description |
|---------------------------------|------------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data type error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C extension module error occurred |

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Modbus digital I/O input signal "di1" =OFF, "di2"=ON, and "di3"
2 =ON
3 res  =  get_modbus_inputs(iobus_list = [ "di1" , "di2" , "di3" ]
4 )
5             #res expected value = 0b110 (binary number), 6
6 (decimal number), or 0x06 (hexadecimal number)
7
8 # Modbus digital I I/O input signal "di4" OFF and "di5" ON
9 res  =  get_modbus_inputs([ "di4" , "di5" ])
10           #res expected value = 0b10 (binary number), 2
11 (decimal number), or 0x02 (hexadecimal number)

```

9.4.12 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) | - | <p>Modbus input name list (configured at TP)</p> <p>Available only with the following signal types</p> <ul style="list-style-type: none"> • DR_MODBUS_REG_INPUT • DR_MODBUS_REG_OUTPUT |

Return

| Value | Description |
|----------|--|
| res | Number values read |
| val_list | List of multiple signal values read simultaneously |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 # Modbus Register I/O "Holding1" is 1234, "Input1" is 567,
2 # and "Holding2" is 9876
3 res, val_list = get_modbus_inputs_list(iobus_list =
4     ["Holding1" , "Input1" , "Holding2" ])
5         #res expected value = 3
          #val_list expected value = [1234, 567, 9876]

```

9.4.13 get_modbus_input_multi (iobus)

Features

This function reads the signal from the Modbus Slave unit.

Parameters

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

Return

| Value | Description |
|-------|---|
| list | List of values corresponding to the number of signals |

Exception

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

Example

| | |
|---|--|
| 1 | #Modbus multiple signal is registered as "multi"(cnt=2). |
| 2 | get_modbus_input("multi") # return = [10, 101] |

```

1 #Modbus multiple signal is registered as "multi"(cnt=2).
2 get_modbus_input( "multi" ) # return = [10, 101]

```

9.4.14 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) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| value | int | - | Modbus digital I/O <ul style="list-style-type: none"> • ON : 1 • OFF : 0 |
| | | | Value for Modbus analog I/O |
| timeout | float | - | Waiting time (sec) This function waits indefinitely if the timeout is not set. |

i Note

When registered as a multiple signal, it is available by adding address value index to signal name.

Return

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

Exception

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

Example

```

1  wait modbus input( "CIN0" , ON) # Indefinite wait until the
2  "CIN0" signal becomes ON
3  wait modbus input( "CIN0" , OFF) # Indefinite wait until the
4  "CIN0" signal becomes OFF
5
6  res = wait modbus input( "CIN0" , ON, 3 ) # Wait for up to 3
7  seconds until the "CIN0" signal becomes ON
8      # res = 0 if the "CIN0" signal becomes ON within 3 seconds.
9      # res = -1 if the "CIN0" signal does not become ON within 3
10     seconds.
11
#Modbus multiple signal is registered as "multi"(start address=10,
cnt=2).
#“multi_10” & “multi_11” abailable
wait_modbus_input( "multi_10" )
wait_modbus_input( "multi_11" )

```

9.4.15 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 set_modbus_slave( 128 , 0 )
2 set_modbus_slave( 255 , 65535 )

```

9.4.16 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) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1   value1 = get_modbus_slave( 128 )
2   value2 = get_modbus_slave( 255 )

```

9.4.17 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 |

| Value | Description |
|--------|--------------------------------------|
| crclow | Low byte of CRC16 calculation result |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  data = b "\x01\x02\x03\x04\x05\x06"
2  crchigh, crclow = modbus_crc16(data)
3  #crchigh = 186(DEC), BA(HEX)
4  #crclow = 221(DEC), DD(HEX)

```

9.4.18 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 calcuation |

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) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 senddata = b"\x01\x02\x03\x04\x05\x06"
2 resultdata = modbus_send_make(senddata)
3 #resultdata = b'\x01\x02\x03\x04\x05\x06\xba\xdd'

```

9.4.19 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 #recvdata = b"\x01\x02\x03\x04\x05\x06\xba\xdd"
2 res = modbus_recv_check(recvdata)
3 #recv = True

```

9.4.20 modbus_unsigned_to_signed(unsigned_data)

Features

When using Modbus protocol, this is a command to convert 2 bytes unsigned data into signed data.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|------------------------------|
| unsigned_data | int | - | 2byte unsigned data(0~65535) |

Return

| Value | Description |
|--------------|-----------------------------------|
| signed_data | 2byte signed data(-32769 ~ 32767) |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

| | |
|---|--|
| 1 | unsigned_data = 40000 |
| 2 | signed_data = modbus_unsigned_to_signed(unsigned_data) |

9.5 Industrial Ethernet (EtherNet/IP,PROFINET)

9.5.1 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 OFF : 0 |

Return

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

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1   set_output_register_bit ( 0 , ON)
2   set_output_register_bit ( 63 , OFF)

```

9.5.2 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 (4byte) |

Return

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

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  set_output_register_int ( 0 , 0x00FF00FF )
2  set_output_register_int ( 23 , 65535 )

```

9.5.3 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 | float | - | float value (4byte) |

Return

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

Exception

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

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

Example

```

1  set_output_register_float ( 0 , 4.5 )
2  set_output_register_float ( 23 , 2.3 )

```

9.5.4 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 |

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  a = get_output_register_bit ( 0 )
2  b = get_output_register_bit ( 63 )

```

9.5.5 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1   a = get_output_register_int( 0 )
2   b = get_output_register_int( 23 )

```

9.5.6 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1   a = get_output_register_float ( 0 )
2   b = get_output_register_float ( 63 )

```

9.5.7 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  a = get_input_register_bit ( 0 )
2  b = get_input_register_bit ( 63 )

```

9.5.8 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```
1 | a = get_input_register_int ( 0 )
```

| | |
|--|----------------------------------|
| | b = get_input_register_int(23) |
|--|----------------------------------|

9.5.9 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 a = get_input_register_float( 0 )
2 b = get_input_register_float( 63 )

```

9.6 FOCAS

9.6.1 `focus_connect(ip, port, timeout)`

Features

This command is used to connect to the Machine Center Controller. When connected normally, a handle value greater than 0 is returned.

The connectable controllers are as follows.

- FANUC Series 30i/31i/32i/35i-MODEL B
- FANUC Series 31i-MODEL B5
- FANUC Series Power Motion i-MODEL A
- FANUC Series 0i-MODEL D/F

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| ip | str | - | Server IP address: (e.g.) "192.168.137.200" |
| port | int | - | Server Port number (e.g.) 8193 |
| timeout | int | - | timeout setting (infinite for 0; unit: s) |

Return

| Value | Description |
|-----------|---|
| Handle | Success (return handle value) |
| errorCode | Error code (refer to <code>focus_get_error_str</code>) |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```
1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
10 )
```

9.6.2 focas_disconnect(handle)

Features

This command is used to disconnect from the Machine Center Controller.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

Return

| Value | Description |
|-----------|--|
| errorCode | 0 : Success Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
ErrCode = focas_disconnect(hMachineCenter)

```

9.6.3 `focas_pmc_read_bit(handle, addr_type, start_num, bit_offset)`

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is bit.

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| bit_offset | int | | Bit Offset(0~7) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (bit type) |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 ErrCode, hMachineCenter = focas_connect(" 10.10 . 0.95 ", 8193 ,
2   10 )
3 ErrCode, PMC_Data = focas_pmc_read_bit(hMachineCenter,"R", 3500 ,
4   0 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.4 focas_pmc_read_char(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is char (1Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| read_count | int | | Read from Start Address Number Number of chars (max. 5) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (char type) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 ErrCode, PMC_Data = focas_pmc_read_char(hMachineCenter, "R" , 10
0 . 3 )
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.5 focas_pmc_read_word(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is word (2Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strat Address Number(0~9999) |
| read_count | int | | Read from Start Address Number Number of words (max. 5) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (word type) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 ErrCode, PMC_Data = focas_pmc_read_word(hMachineCenter, "R" , 35
00 . 3 )
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.6 focas_pmc_read_long(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is long (4Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| read_count | int | | Read from Start Address Number Number of longs (max. 5) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (long type) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 ErrCode, PMC_Data = focas_pmc_read_long(hMachineCenter, "R" , 35
00 . 3 )
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.7 focas_pmc_read_float(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller.

It is used when the data return type is float (4Byte, 32-bit-floatring-point-type).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| read_count | int | | Read from Start Address Number Number of floats (max. 5) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (float type) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 ErrCode, PMC_Data = focas_pmc_read_float(hMachineCenter, "D" , 1
0 . 3 )
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.8 focas_pmc_read_double(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller.

It is used when the data return type is double (8Byte, 64-bit-floating-point-type).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| read_count | int | | Read from Start Address Number Number of doubles (max. 5) |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |
| pmc_data | PMC data (double type) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 ErrCode, PMC_Data = focas_pmc_read_double(hMachineCenter, "D" , 1
0 . 3 )
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.9 focas_pmc_write_bit(handle, addr_type, start_num, bit_offset, write_data)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is bit.

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| bit_offset | int | | Bit Offset(0~7) |
| write_data | Int | | ON : 1 OFF : 0 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 ErrCode = focas_pmc_write_bit(hMachineCenter, "R" , 3000 , 0 ,
ON)
3 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.10 focas_pmc_write_char(handle, addr_type, start_num, write_data, write_count)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is char (1Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) <ul style="list-style-type: none"> • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| write_data | byte | | The type of data to be transmitted is byte. |
| write_count | Int | | Enter the number of char data to be transmitted. Up to 5 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 MC_Command = [ 129 , 213 ]
4 ErrCode = focas_pmc_write_char(hMachineCenter, "R" , 3000 ,
MC_Command, 2 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.11 focas_pmc_write_word(handle, addr_type, start_num, write_data, write_count)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is word (2Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) <ul style="list-style-type: none"> • Case insensitive |
| start_num | Int | | Start Address Number(0~9999) |
| write_data | word | | The type of data to be transmitted is word(2Byte). |
| write_count | Int | | Enter the number of word data to be transmitted. Up to 5 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 MC_Command = [ - 129 , 1213 ]
4 ErrCode = focas_pmc_write_word(hMachineCenter, "R" , 3100 ,
MC_Command, 2 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.12 focas_pmc_write_long(handle, addr_type, start_num, write_data, write_count)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is long (4Byte).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) <ul style="list-style-type: none"> • Case insensitive |
| start_num | Int | | Start Address Number(0~9999) |
| write_data | int | | The type of data to be transmitted is long(4Byte). |
| write_count | Int | | Enter the number of long data to be transmitted. Up to 5 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 MC_Command = [ - 7129 , 11213 ]
4 ErrCode = focas_pmc_write_long(hMachineCenter, "G" , 3100 ,
MC_Command, 2 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.13 focas_pmc_write_float(handle, addr_type, start_num, write_data, write_count)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is float (4Byte, 32-bit-floatring-point-type).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) <ul style="list-style-type: none"> • Case insensitive |
| start_num | Int | | Start Address Number(0~9999) |
| write_data | float | | The type of data to be transmitted is float(4Byte). |
| write_count | Int | | Enter the number of float data to be transmitted. Up to 5 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

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

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 MC_Command = [ - 178.12 , 11.478 ]
4 ErrCode = focas_pmc_write_float(hMachineCenter, "G" , 3100 ,
MC_Command, 2 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.14 focas_pmc_write_double(handle, addr_type, start_num, write_data, write_count)

Features

This command is used to write PMC Data of Machine Center Controller.

It is used when the data return type is double (8Byte, 64-bit-floating-point-type).

⚠ Caution

Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| addr_type | str | | G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) <ul style="list-style-type: none"> • Case insensitive |
| start_num | Int | | Strart Address Number(0~9999) |
| write_data | double | | The type of data to be transmitted is double(8Byte). |
| write_count | Int | | Enter the number of double data to be transmitted. Up to 5 |

Return

| Value | Description |
|-----------|--|
| errorCode | 0: Success (communication is canceled normally) Value other than 0: error (see <code>focas_get_error_str</code>) |

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2 10 )
3 MC_Command = [ - 178.12 , 11.478 ]
4 ErrCode = focas_pmc_write_double(hMachineCenter, "G" , 3100 ,
MC_Command, 2 )
5 ErrCode = focas_disconnect(hMachineCenter)

```

9.6.15 focas_get_error_str(handle, errorCode)

Features

It is used to analyze the errorCode returned when using the Focas Library related function. When entering an error code, the cause of the related error is returned as a string.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| handle | int | - | Communication specific control constant value required for use of FOCAS |
| errorCode | int | - | Error code returned after FOCAS related DRL execution |

Return

| Value | Description |
|-----------------|------------------------------------|
| errorCodeString | Details of the error code (string) |

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

ErrorCode Details

| Exception | Description |
|------------------|--|
| .-17 | <p>The following message is displayed depending on the detailed cause.</p> <p>"The send data is larger than the maximum transfer unit"</p> <p>"The sending data size is illegal"</p> <p>"The number of the received packet is 0"</p> <p>"The mark of the protocol is incorrect in the received packet header"</p> <p>"The packet type flag is incorrect in the received packet header"</p> <p>"The flag of the direction is incorrect in the received packet header"</p> <p>"Illegal received data size"</p> <p>"Communication error in the Ethernet Board "</p> <p>"protocol error"</p> |

| Exception | Description |
|------------------|--|
| -16 | The following message is displayed depending on the detailed cause. "Error of socket API function ") "Error of connect API function") "Error of send API function" "Error of recv API function" "Error of select API function" "Error of setsockopt API function" "Error of gethostbyname API function" "Timeout error of send API function" "Timeout error of recv API function" "Error of Winsock API is occurred in other process" "EOF (end of file) detected " "socket error" |
| -15 | "DLL not exist error" |
| -14 | "error in APi library initial valuefile" |
| -13 | "low temperature alarm of intelligent terminal" |
| -12 | "hight temperature alarm of intelligent terminal" |
| -11 | "bus error" |
| -10 | "system error" |
| -9 | "hssb communication error" |
| -8 | "library handle error" |
| -7 | "CNC/PMC version mismatch" |
| -6 | "abnormal error" |
| -5 | "system error" |
| -4 | "shared RAM parity error" |

| Exception | Description |
|------------------|--|
| -3 | "emm386 or mmcys install error" |
| -2 | "reset or stop occured error" |
| -1 | "busy error" |
| 0 | "no problem" |
| 1 | "command prepare error or pmc not exist" |
| 2 | "data block length error" |
| 3 | "data number error or address range error" |
| 4 | "data attribute error or data type error" |
| 5 | "data error" |
| 6 | "no option error" |
| 7 | "write protect error" |
| 8 | "memory overflow error" |
| 9 | "cnc parameter not correct error" |
| 10 | "buffer error" |
| 11 | "path error" |
| 12 | "cnc mode error" |
| 13 | "execution rejected error" |
| 14 | "data server error" |
| 15 | "alarm has been occurred" |
| 16 | "CNC is not running" |

| Exception | Description |
|-----------|--|
| 17 | "protection data error" |
| 18 | "error generated by PMC" |
| 19 | "PMC handle error" |
| 20 | "overwrite stop in program read" |
| 21 | "reset interrupt in program read" |
| -100 | "Library opening failed." |
| -101 | "The maximum number of machine tools that can be connected has been exceeded." |
| -102 | "The handle is not connected" |

Example

```

1 ErrCode, hMachineCenter = focas_connect( "10.10.0.95" , 8193 ,
2   10 )
3 if ErrCode != 0 :
4   ErrorString = focas_get_error_str(hMachineCenter, ErrCode)
5   MC_Command = [- 178.12 , 11.478 ]
6   focas_pmc_write_double(hMachineCenter, "G" , 3100 , MC_Command,
7     2 )
8   focas_disconnect(hMachineCenter)

```

10 External Vision Commands

10.1 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.

10.2 2D Vision - COGNEX / SICK / VISOR

2D Vision (COGNEX, SICK, VISOR) commands are composed as follows.

| No. | Type | Command |
|-----|------------------------------|--|
| 1 | Select Manufacturer | vs_set_info(type)(p. 404) |
| 2 | Camera Connection | vs_connect(ip_addr, port_num=9999)(p. 405) vs_disconnect()(p. 406) |
| 3 | Vision Job Manage | vs_get_job()(p. 406) vs_set_job(job_name)(p. 407) cognex_set_integer (job_name, integer_number)(p. 408) visor_job_change(index)(p. 408) |
| 4 | Object Recognition/Detection | vs_trigger()(p. 410) |
| 5 | Robot Task | vs_set_init_pos(vision_posx_init, robot_posx_init, vs_pos=1)(p. 411) |

| No. | Type | Command |
|----------|--------|---|
| | | vs_get_offset_pos(vision_posx_meas, vs_pos=1)(p. 412) |
| 6 | Custom | vs_request(cmd)(p. 414) |
| | | vs_result()(p. 415) |

10.2.1 **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(0) DR_VS_COGNEX(1) DR_VS_SICK(2) DR_VS_VISOR(3) | |

Return

| Value | Description |
|------------|-------------------|
| ID of Type | ID of type to set |

Example

```

1  vs_set_info(DR_VS_COGNEX) #Vision type information setting
2  vs_connect( "192.168.137.10" ) #Connect to vision - Vision IP,
   Default port
3  # Enter your task
4  vs_disconnect()               #Disconnect to vision

```

- [Supported Model](#)

| Type | Model |
|--------------|--|
| DR_VS_COGNEX | COGNEX IS2000M-23M Series COGNEX IS5600/5705 Series COGNEX IS7000Series COGNEX IS8000Series |
| DR_VS_SICK | SICK Inspector PIM60 Series SICK Inspector PI50 Series |
| DR_VS_VISOR | Sensopart V20 Series |
| DR_VS_CUSTOM | |

[10.2.2 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 Number (ex, 9999) |

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```

1  vs_set_info(DR_VS_COGNEX) #Vision type information setting
2  vs_connect( "192.168.137.10" ) #Connect to vision - Vision IP,
   Default port
3  # Enter your task
4  vs_disconnect()           #Disconnect to vision

```

10.2.3 vs_disconnect()

Features

This function terminates the connection to the vision system.

Return

| Value | Description |
|-------|-------------|
| N/A | N/A |

Example

```

1  vs_set_info(DR_VS_COGNEX) #Vision type information setting
2  vs_connect( "192.168.137.10" ) #Connect to vision - Vision IP,
   Default port
3  # Enter your task
4  vs_disconnect()           #Disconnect to vision

```

10.2.4 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 |

| Value | Data Type | Description |
|-------|-----------|-------------------|
| -1 | int | Connection failed |

Example

```

1 vs_set_info(DR_VS_COGNEX) #Vision type information setting
2 vs_connect( "192.168.137.10" ) #Connect to vision - Vision IP,
3 Default port
4 vs_set_job( "test.job" ) # Set (load) the current vision
5 job
6 job_name = vs_get_job() # Get the current setting vision
7 job
8 tp_popup( "{0}" . format (job_name))
9
vs_disconnect() # Disconnect to vision

```

10.2.5 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

```

1 vs_set_info(DR_VS_COGNEX) #Vision type information setting

```

```

2   vs_connect( "192.168.137.10" )    #Connect to vision - Vision IP,
3   Default port
4   vs_set_job( "test.job" )           # Set (load) the current vision
5   job
6   job_name = vs_get_job()          # Get the current setting vision
7   job
8   tp_popup( "{0}" . format (job_name))
9
10  vs_disconnect()                 # Disconnect to vision

```

10.2.6 cognex_set_integer(job_name, integer_number)

Features

Loads the entered index of the corresponding job.

(*VS_TYPE: DR_VS_COGNEX)

Return

| Value | Description |
|-------|-------------|
| 1 | Success |
| -1 | Fail |

Example

```

1   vs_set_info(DR_VS_COGNEX)           #Vision type information
2   setting
3   vs_connect( "192.168.137.10" )     #Connect to vision - Vision
4   IP, Default port
5   res  =  cognex_set_integer( "Pattern_1.Scale_Tolerance" , 1 )
6   # Enter your task
7   tp_popup( "{0}" . format (res))
8
9   vs_disconnect()                   #Disconnect to vision

```

10.2.7 visor_job_change(index)

Features

Loads the Vision Sensor setting using the entered number.

(*VS_TYPE: DR_VS_VISOR)

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--------------------------|
| index | int | - | Job number to be changed |

Return

| Value | Description |
|--------|---|
| result | Job change success (1) Job change failure (-1) |

Example

```

1 vs_set_info(DR_VS_VISOR)           # Select type of vision
2 sensor
3 vs_connect( "192.168.137.101" )      # Vision IP, Default
port
4 vis_posx = posx( 410 , 310 , 300 , 0 , 0 , 0 )      # Define
the initial posx data - vision
5 rob_posx = posx( 400 , 300 , 300 , 0 , 180 , 0 )      # Define
the initial posx data - robot
6 vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial
posx data to Vision
7 visor_job_change( 2 )                  # change the job as the
input parameter
8
9 for i in range( 10 ):
10    pos, var_list = vs_trigger()          # Execute the
vision measurement
11    tp_popup( "{0}" . format (var_list))
12
13    if var_list[ 0 ] == 1 :                # Check the
inspection result
14        robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
15        tp_popup( "{0}" . format (robot_posx_meas))
16    else :
17        tp_popup( "Inspection Fail" )
vs_disconnect()

```

10.2.8 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 ex) Check pass/fail, distance measurement, angle measurement, etc. |
| -1, [] | int | failed |

Example

```

1  vs_set_info(DR_VS_COGNEX)                      # Select type of vision
2  sensor
3
4  vs_connect( "192.168.137.10" )                  # Vision IP, Default
5  port
6
7  vis_posx = posx ( 410 , 310 , 300 , 0 , 0 , 0 )   #
8  Define the initial posx data - vision
9  rob_posx = posx ( 400 , 300 , 300 , 0 , 180 , 0 )  # Define
10  the initial posx data - robot
11
12  vs_set_init_pos(vis_posx, rob_posx, VS_POS1)    # Enter the initial
13  posx data to Vision
14
15  for i in range ( 10 ):                          #
16      pos, var_list = vs_trigger()                 # Execute the
17      vision meausrement
18      if var_list[ 0 ] == 1 :                      # Check the
19          inspection result
20          robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
21          moveL(robot_posx_meas)                   # move the robot pose
22      else :
23          tp_popup( "Inspection Fail" )
24
25  vs_disconnect()

```

10.2.9 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 |
| -1 | int | Failed |

Example

```

1  vs_set_info(DR_VS_COGNEX)           # Select type of vision
sensor
2  vs_connect( "192.168.137.10" )      # Vision IP, Default
port
3  vis_posx = posx( 410 , 310 , 300 , 0 , 0 , 0 )          #
Define the initial posx data - vision
4  rob_posx = posx( 400 , 300 , 300 , 0 , 180 , 0 )        # Define
the initial posx data - robot
5  vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial
posx data to Vision
6  for i in range( 10 ):
7      pos, var_list = vs_trigger()             # Execute the
vision measusrement
8      if var_list[ 0 ] == 1 :                  # Check the
inspection result

```

```

9     robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # 
offset the robot pose
10    movel(robot_posx_meas)                                # move the robot pose
11    else :
12        tp_popup( "Inspection Fail" )
13    vs_disconnect()

```

10.2.10 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

```

1  vs_set_info(DR_VS_COGNEX)                      # Select type of vision
sensor
2  vs_connect( "192.168.137.10" )                 # Vision IP, Default
port
3
4  vis_posx = posx( 410 , 310 , 300 , 0 , 0 , 0 )   #
Define the initial posx data - vision
5  rob_posx = posx( 400 , 300 , 300 , 0 , 180 , 0 )   # Define
the initial posx data - robot
6

```

```

7  vs_set_init_pos(vis_posx, rob_posx, VS_POS1) # Enter the initial
8  posx data to Vision
9
10 for i in range(10):
11     pos, var_list = vs_trigger()                 # Execute the
12     vision measurement
13     if var_list[0] == 1:                         # Check the
14         inspection result
15         robot_posx_meas = vs_get_offset_pos(pos, VS_POS1) # offset the robot pose
16         moveL(robot_posx_meas)                      # move the robot pose
17     else:
18         tp_popup("Inspection Fail")
19
20 vs_disconnect()

```

10.2.11 Integrated example - DR_VS_COGNEX, DR_VS_SICK

Example

```

1  vs_set_info(DR_VS_COGNEX)                      # Select type of
2  vision sensor
3  if (vs_connect("192.168.137.10") != 0):          # Vision IP, Default port
4      tp_popup("connection fail", DR_PM_MESSAGE)
5      exit()
6
7  vis_posx_init = posx(410, 310, 300, 0, 0, 0)    # Define the initial posx data - vision
8  rob_posx_init1 = posx(400, 300, 300, 0, 180, 0)   # Define the initial posx data - robot
9  rob_posx_init2 = posx(420, 320, 300, 0, 180, 0)   # Define the initial posx data - robot
10
11 vs_set_init_pos(vis_posx_init, rob_posx_init1, VS_POS1) # Enter
12 the initial posx data to Vision
13 vs_set_init_pos(vis_posx_init, rob_posx_init2, VS_POS2)
14
15 for i in range(10):
16     pos_meas, var_list = vs_trigger()                # Execute the vision
17     measurement
18     if pos_meas == -1:                             # Vision Fail to
19         measure the object
20         tp_popup("Vision measure fail")
21         continue
22     if var_list[0] == 1:                            # Check the
23         inspection result
24         # Get guided posx data
25         rob_posx1_meas = vs_get_offset_pos(pos_meas, VS_POS1) # offset the robot pose

```

```

21      rob_posx2_meas = vs_get_offset_pos(pos_meas, VS_POS2) # 
offset the robot pose
22      movel(rob_posx1_meas)
23      movel(rob_posx2_meas)
24  else :
25      tp_popup( "Inspection Fail" )
26      continue
27
28  vs_disconnect()

```

10.2.12 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

| | |
|---|--|
| 1 | vs_request(1) # request the vision measurement on the “1” job |
|---|--|

10.2.13 vs_result()

Features

This function retrieves the processing result of the vision system.

(*VS_TYPE: DR_VS_CUSTOM)

Return

| Value | Description |
|--------------|---|
| cnt (>=1) | Success 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 |

Example

```

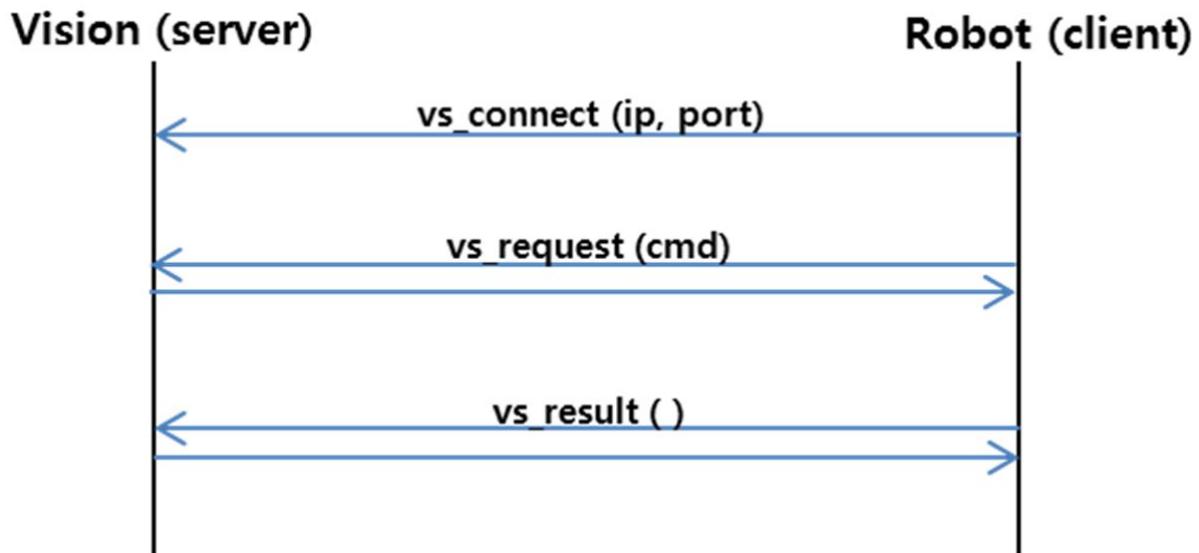
1  vs_set_info(DR_VS_CUSTOM)
2  res = vs_connect( "192.168.137.200" , 9999 )           #Vision and
   communication connection attempt
3  if res != 0 :          #Check the result of communication
   connection
4    tp_popup( "connection fail" ,DR_PM_MESSAGE)  #If connection
   fails, program ends
5    exit()
6
7  ret = vs_request( 1 )        #Request for object vision
   measurement information
8
9  cnt, result = vs_result()      # Get object measurement
   result information
10
11 for i in range (cnt):
12   x = result[i][ 0 ]
13   y = result[i][ 1 ]
14   t = result[i][ 2 ]
15   tp_popup( "x={0},y={1}, t={2}" . format (result[i][ 0 ],
   result[i][ 1 ], result[i][ 2 ]),DR_PM_MESSAGE)

```

10.2.14 Integrated example - DR_VS_CUSTOM

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**
 - Acutal 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**0000006E**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**0000006E**000000DC**0000014A0000006E**FFFF24**
FFFFFEB6

Example

```

1  vs_set_info(DR_VS_CUSTOM)
2  res = vs_connect( "192.168.137.200" , 9999 )    #Vision and
   communication connection attempt
3  if res != 0 :
   #Check the result of communication connection
4  tp_popup( "connection fail" ,DR_PM_MESSAGE)  #Upon connection
   failure, program termination
5  exit()
6
7  ret = vs_request( 1 )
   #Request of Vision Measurement Information for No. 1 Object
8
9  cnt, result = vs_result()                      #Get
   object measurement result information
10
11 for i in range (cnt):
12     x = result[i][ 0 ]
13     y = result[i][ 1 ]
14     t = result[i][ 2 ]
15     tp_popup( "x={0},y={1}, t={2}" . format (result[i][ 0 ],
   result[i][ 1 ], result[i][ 2 ]),DR_PM_MESSAGE)

```

10.3 Pickit 3D

Pickit 3D commands are composed as follows.

| No | 구분 | 명령어 |
|----|-------------------|--|
| . | | |
| 1 | Camera Connection | pickit_connect(ip)(p. 418) |

| No. | 구분 | 명령어 |
|-----|----------------------------------|--|
| . | | pickit_disconnect()(p. 419) |
| 2 | Vision Job Manage | pickit_change_configuration(setup_id, product_id))(p. 419) |
| 3 | Camera Calibration | pickit_request_calibration()(p. 420) |
| 4 | Object Recognition/ Detection | pickit_detection(offset_z)(p. 420) pickit_next_object(offset_z)(p. 422) |
| 5 | Backup | pickit_save_snapshot()(p. 424) |

10.3.1 pickit_connect(ip)

Features

This function establishes communication with the vision system.

The default IP of PickIt is 192.168.66.1, and the user must use an PickIt IP in the same bandwidth range as Robot IP. See the Pickit Support site for details on how to use it.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| ip_addr | str | - | Server IP of Pickit 3D (ex, 192.168.137.90) |

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```
1   pickit_connect( "192.168.137.90" ) #Connect to vision - Vision IP
```

2

`pickit_disconnect()`

#Disconnect to vision

10.3.2 pickit_disconnect()

Features

This function terminates the connection to the vision system.

Return

| Value | Description |
|-------|-------------|
| N/A | |

Example

1
2

```
pickit_connect( "192.168.137.90" ) #Connect to vision - Vision IP
pickit_disconnect() #Disconnect to vision
```

10.3.3 pickit_change_configuration(setup_id, product_id))

Features

This function loads setup_id and product_id set in the vision system.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| Setup_id | int | - | The setup_id number stored in the Pickit server. |
| Product_id | int | | The product_id number stored in the Pickit server |

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```

1  pickit_connect( "192.168.137.90" )
2  pickit_change_configuration( 6 , 8 ) # These numbers are defined
   in Pickit
3  pickit_disconnect()

```

10.3.4 pickit_request_calibration()

Features

This function requests a calibration once from the vision system

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```

1  pickit_connect( "192.168.137.90" )
2  pickit_request_calibration()
3  pickit_disconnect()

```

10.3.5 pickit_detection(offset_z)

Features

This function detects the input model and returns (pick_prepos, pick_pos).

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| offset_z | int | - | The offset_z sets to 'pick_prepos' distance. |

Return

| Value | Data type | Description |
|---|---|--|
| <pre>data_dictionary = {'pick_pos':pick_pos, 'pick_prepos':pick_prepos, 'object_age':data['object_age'], 'object_type':data['object_type'], 'object_dimensions':data['object_dimensions'], 'object_remaining':data['objects_remaining'], 'status':data['status']</pre> | <pre>{'pick_pos': posx, 'pick_prepos': posx, 'object_age': int, 'object_type': int, 'object_dimensions': int, 'object_remaining': int, 'status': int}</pre> | <p>'pick_pos': Model recognition position,</p> <p>'pick_prepos': Offset Value of model recognition position,</p> <p>'object_age': The amount of time that has passed between the capturing of the camera data and the moment the object information is sent to the robot. This value has to be divided by the MULT factor.,</p> <p>'object_type': The type of object detected at object_pose ,</p> <p>'object_dimensions': When reading array elements, each value has to be divided by the MULT factor.,</p> <p>'object_remaining': Only one object per pickit_to_robot_data message can be communicated. If this field is non-zero, it contains the number of remaining objects that can be sent in next messages to the robot.,</p> <p>'status': Contains the Pickit status or a response to previously received robot commands.</p> |

Example

```

1 set_singular_handling(DR_AVOID)
2 set_veli( 60.0 )
3 set_acci( 100.0 )
4 set_velx( 250.0 , 80.625 )
5 set_accx( 1000.0 , 322.5 )
6
7 pickit_connect( "192.168.137.90" )
8 pickit_change_configuration( 7 , 10 ) # These numbers are
defined in Pickit
9
10 data = pickit_detection( 100 )
11 if data[ 'status' ] == ResponseStatus.OBJECTS_FOUND:
12     # Picking motion
13     movel(data[ 'pick prepos' ])
14     movel(data[ 'pick pos' ])
15     movel(data[ 'pick_prepos' ])
16
17 pickit_disconnect()
```

10.3.6 pickit_next_object(offset_z)

Features

This function returns pick_prepos and pick_pos detected next to the input model.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| offset_z | int | - | The offset_z sets to 'pick_prepos' distance. |

Return

| Value | Data Type | Description |
|---|---|--|
| <pre>data_dictionary = {'pick_pos':pick_pos, 'pick_prepos':pick_prepos, 'object_age':data['object_age'], 'object_type':data['object_type'], 'object_dimensions':data['object_dimensions'], 'object_remaining':data['objects_remaining'], 'status':data['status']}</pre> | <pre>{'pick_pos': posx, 'pick_prepos': posx, 'object_age': int, 'object_type': int, 'object_dimension s': int, 'object_remaining' :int , 'status': int}</pre> | <p>'pick_pos': Model recognition position,</p> <p>'pick_prepos': Offset value of model recognition position,</p> <p>'object_age': The amount of time that has passed between the capturing of the camera data and the moment the object information is sent to the robot. This value has to be divided by the MULT factor.,</p> <p>'object_type': The type of object detected at object_pose ,</p> <p>'object_dimensions': When reading array elements, each value has to be divided by the MULT factor .,</p> <p>'object_remaining': Only one object per pickit_to_robot_data message can be communicated. If this field is non-zero, it contains the number of remaining objects that can be sent in next messages to the robot .,</p> <p>'status': Contains the Pickit status or a response to previously received robot commands.</p> |

Example

```

1  pickit connect( "192.168.137.90" )
2  pickit change configuration( 7 , 10 ) # These numbers are
defined in Pickit
3
4  set singular handling(DR_AVOID)
5  set veli( 60.0 )
6  set acci( 100.0 )
7  set velx( 250.0 , 80.625 )
8  set_accx( 1000.0 , 322.5 )
```

```

9   home_pos = posx( 122.09 , 35.28 , 303.63 , 140.45 , 158.9 ,
10  134.39 )
11  while True :
12      # Detection
13      data = pickit_detection( 100 )
14      if data[ 'status' ] == ResponseStatus.OBJECTS_FOUND:
15          # Picking motion
16          movel(data[ 'pick_prepos' ])
17          movel(data[ 'pick_pos' ])
18          movel(data[ 'pick_prepos' ])
19
20          remaining = data[ 'object_remaining' ]
21          while remaining > 0 :
22              data = pickit_next_object( 100 )
23              remaining = data[ 'object_remaining' ]
24
25              # Picking motion
26              movel(data[ 'pick_prepos' ])
27              movel(data[ 'pick_pos' ])
28              movel(data[ 'pick_prepos' ])
29
    pickit_disconnect()

```

10.3.7 pickit_save_snapshot()

Features

This function saves the snapshot to the server.

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```

1  pickit_connect( "192.168.137.90" )
2
3  pickit_save_snapshot()
4
5  pickit_disconnect()

```

10.3.8 Integrated example - Pickit 3D

Example 1

```

1  ### For robot-camera calibration example ####
2
3  set_singular_handling(DR_AVOID)
4  set_veli( 60.0 )
5  set_acci( 100.0 )
6  set_velx( 250.0 , 80.625 )
7  set_accx( 1000.0 , 322.5 )
8  pickit_connect( "192.168.137.90" )
9
10 # Move to Pose #
11 pos1 = posx( 476.76 , - 151.68 , 384.83 , 33.36 , 24.71 ,
12   95.44 )
13 pos2 = posx( 495.86 , - 150.08 , 435.69 , 1.43 , 8.21 ,
14   122.25 )
15 pos3 = posx( 508.79 , - 83.06 , 446.94 , 82.88 , - 35.91 ,
16   39.23 )
17 pos4 = posx( 521.66 , - 146.28 , 431.8 , 29.71 , - 33.73 ,
18   82.42 )
19 pos5 = posx( 508.35 , - 147.45 , 386.37 , 101.03 , 38.04 ,
20   42.68 )
21 moveL(pos1)
22 pickit_request_calibration()
23 moveL(pos2)
24 pickit_request_calibration()
25 moveL(pos3)
26 pickit_request_calibration()
27 moveL(pos4)
28 pickit_request_calibration()
29 moveL(pos5)
30 pickit_request_calibration()
```

Example 2

```

1  ### For simple picking example ####
2  pickit_connect( "192.168.137.90" )
3  pickit_change_configuration( 7 , 10 ) # These numbers are
4  defined in Pickit
5
6  set_singular_handling(DR_AVOID)
7  set_veli( 60.0 )
8  set_acci( 100.0 )
9  set_velx( 250.0 , 80.625 )
10 set_accx( 1000.0 , 322.5 )
```

```

10 home_pos = posx( 122.09 , 35.28 , 303.63 , 140.45 , 158.9 ,
11 134.39 )
12     #Move to home pose
13 movel(home_pos)
14
15     # Detection
16 data = pickit_detection( 100 )
17 if data[ 'status' ] == ResponseStatus.OBJECTS_FOUND:
18     # Picking motion
19     movel(data[ 'pick_prepos' ])
20     movel(data[ 'pick_pos' ])
21     movel(data[ 'pick_prepos' ])
22
23 pickit_disconnect()

```

Example 3

```

1     ### For multiple parts picking example ####
2
3     pickit_connect( "192.168.137.90" )
4     pickit_change_configuration( 7 , 10 ) # These numbers are
defined in Pickit
5
6     set_singular_handling(DR_AVOID)
7     set_veli( 60.0 )
8     set_acci( 100.0 )
9     set_velx( 250.0 , 80.625 )
10    set_accx( 1000.0 , 322.5 )
11    home_pos = posx( 122.09 , 35.28 , 303.63 , 140.45 , 158.9 ,
12 134.39 )
13 while True :
14     # Move to home pose
15     movel(home_pos)
16
17     # Detection
18     data = pickit_detection( 100 )
19     if data[ 'status' ] == ResponseStatus.OBJECTS_FOUND:
20
21         # Picking motion
22         movel(data[ 'pick_prepos' ])
23         movel(data[ 'pick_pos' ])
24         movel(data[ 'pick_prepos' ])
25
26         remaining = data[ 'object_remaining' ]
27         while remaining > 0 :
28             data = pickit_next_object( 100 )
29             remaining = data[ 'object_remaining' ]
30
31             # Picking motion
32             movel(data[ 'pick_prepos' ])
33             movel(data[ 'pick_pos' ])

```

33

```
moveL(data[ 'pick_prepos' ])
```

11 Doosan Vision(SVM) Command

Doosan Smart Vision Module(SVM) commands are composed as follows.

| | Type | Details | Command |
|---|----------------------------------|--------------------|--|
| 1 | Camera Connection | | .svm_connect(ip="192.168.137.5", port=20) v2.8working(p. 429) |
| | | | svm_disconnect()(p. 430) |
| 2 | Check Image Quality | | svm_set_led_brightness(value)(p. 430) |
| | | | svm_get_led_brightness()(p. 431) |
| | | | svm_set_camera_exp_val(value)(p. 431) |
| | | | svm_set_camera_gain_val(value)(p. 432) |
| | | | svm_set_camera_load(job_id)(p. 433) |
| 3 | Vision Job Manage | | svm_set_job(job_id)(p. 433) |
| 4 | Camera Calibration | | svm_get_robot_pose(job_id)(p. 434) |
| 5 | Object Recognition/ Detection | Object Detection | svm_get_vision_info(job_id)(p. 435) |
| | | | svm_get_variable(tool_id, var_type)(p. 436) |
| | | Landmark Detection | svm_detect_landmark(job_id)(p. 437) |
| | | | svm_get_marker_offset_pose(cpos, offset, euler_mode)(p. 438) |
| | | Barcode Detection | svm_get_detect_barcode()(p. 440) |
| | | | svm_get_barcode_db_data(index)(p. 440) |
| | | | svm_compare_barcode_db_data(dbdata)(p. 441) |
| 6 | Robot Task | | svm_set_init_pos_data(Id_list, Pos_list)(p. 442) |

| | Type | Details | Command |
|---|--------------------|---------|--|
| | | | svm_get_offset_pos(posx_robot_init, job_id, tool_id)(p. 443) |
| 7 | Other SVM commands | | svm_set_tp_popup (svm_flag)(p. 445) |

11.1 Camera Connection

11.1.1 **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 | “192.168.1 37.5” | SVM Server IP address |
| port | int | 20 | Port number |

Return

| Value | Description |
|-------|--------------------|
| 0 | Connection success |
| -1 | Connection failed |

Example

```

1  svm_connect()      #Connect to vision - Default IP address and
2  # port number
3  # Enter the vision task
4  svm_disconnect()   #Disconnect to vision

```

11.1.2 svm_disconnect()

Features

This function terminates the connection to the SVM.

Return

| Value | Description |
|-------|-------------|
| N/A | N/A |

Example

```

1  svm_connect()  #Connect to vision - Default IP address and port
2  # Enter the vision task
3  svm_disconnect()      #Disconnect to vision

```

11.2 Check Image Quality

11.2.1 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 | Description |
|-------|---|
| -1 | Fail - No measurement data or input variable error. |

Example

| | | |
|---|-------------------------------|------------------------|
| 1 | svm_connect() | # Connect to vision |
| 2 | svm_set_led_brightness(500) | # Disconnect to vision |
| 3 | svm_disconnect() | |

11.2.2 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

| | | |
|---|--------------------------|------------------------|
| 1 | svm_connect() | # Connect to vision |
| 2 | svm_get_led_brightness() | # Disconnect to vision |
| 3 | svm_disconnect() | |

11.2.3 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

| | | |
|---|---|------------------------|
| 1 | svm_connect() | # Connect to vision |
| 2 | svm_set_camera_exp_val(2 , 660 , 000) | # Disconnect to vision |
| 3 | svm_disconnect() | |

11.2.4 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

| | | |
|---|--------------------------------|------------------------|
| 1 | svm_connect() | # Connect to vision |
| 2 | svm_set_camera_gain_val(500) | # Disconnect to vision |
| 3 | svm_disconnect() | |

11.2.5 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 | int | | job id(ex - 1000, 2000, 3000) |

Return

| Value | Description |
|-------|---|
| -1 | Fail - No measurement data or input variable error. |

Example

```

1  svm_connect()                                # Connect to vision
2  svm_set_camera_load(job_id)
3  svm_disconnect()                            # Disconnect to vision

```

11.3 Vision Job Manage

11.3.1 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

```

1  svm_connect()          #Connect to vision - Vision IP, Default
2  port
3  vision_test = 1000      # Define vision job ID
4  svm_set_job(vision_test) # Load the vision_test (1000)
5  svm_disconnect()        #Disconnect to vision

```

11.4 Camera Calibration

11.4.1 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 (예. 1000, 2000, ...) |

Return

| Value | Description |
|-----------|--|
| float [6] | Robot joint coordinate information (posj type) |
| -1 | failed |

Example

```

1  svm_connect()                                # Connect to vision
2  vision_test = 1000                          # Define vision job
3  ID
4  svm_set_job(vision_test)                   # Load the vision_test
5  (1000)
6  shoot_pos = svm_get_robot_pose (vision_test) # Load the robot
pose of vision test
7  tp_popup( "{0}" . format (shoot_pos))      # Disconnect to vision
8  svm_disconnect()                           # Disconnect to vision

```

11.5 Object Recognition/Detection

11.5.1 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 | Description |
|-------|---|
| 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

| | | |
|---|---------------|---------------------|
| 1 | svm_connect() | # Connect to vision |
|---|---------------|---------------------|

```

2 vision_test = 1000 # Define vision job
ID
3 svm_set_job(vision_test) # Load the vision_test
(1000)
4 shoot_pos = svm_get_robot_pose (vision_test) # Load the robot
pose of vision_test
5 count = svm_get_vision_info(vision_test) # Execute the vision
measurement
6 tp_popup( "{0}" . format (count)) # Check the result
7 svm_disconnect() # Disconnect to vision

```

11.5.2 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) INSP_TYPE: Inspection result variable (int) VALUE_TYPE: Measurement result (int or float) |

Return

| Value | Description |
|----------|--|
| variable | POSX_TYPE – Coordinate information variable, ex. Posx(x,y,z,rx,ry,rz) INSP_TYPE: Inspection result variable - int (Returns 1 if successful) VALEU_TYPE: Measurement result variable (int or float) |
| -1 | Failed – No measurement data or input variable error. |

Example

```

1  svm_connect()                      # Connect to vision
2  vision_test = 1000                 # Define vision job
3  ID
4  print_insp = 1001                  # Define inspection
5  tool_ID
6  box_size = 1002                   # Define measurement
7  tool_ID
8  count = svm_get_vision_info(vision_test)    # Execute the vision
9  measurement
10 if (count == 1):                  # Check the result
11
12     # Get the position information (posx) of vision test tool
13     pos_result = svm_get_variable(vision_test, POSX_TYPE)
14     tp_popup( "{0}" . format (pos_result))
15
16     # Get the inspection information (PASS or Fail) of print_insp
17     tool
18     inspection_result = svm_get_variable(print_insp, INSP_TYPE)
19     tp_popup( "{0}" . format (inspection_result))
20
21     # Get the distance information (distance) of box_size tool
22     measurement_result = svm_get_variable(box_size, VALUE_TYPE)
23     tp_popup( "{0}" . format (measurement_result))
24
25 vs_disconnect()                   # Disconnect to vision

```

11.5.3 svm_detect_landmark(job_id)

Features

Detect the landmark information with respect to Camera coordinage using a job_id.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------------------------|
| job | int | | job id(ex - 1000, 2000, 3000) |

Return

| Value | Data Type | Description |
|-------|-------------------------|---|
| count | int | Number of detected landmarks |
| cpos | list[Tx,Ty,Tz,Rx,Ry,Rz] | Landmark pose based on camera coordinates |
| 0, [] | Int, list | landmark detection failure |

Example

```

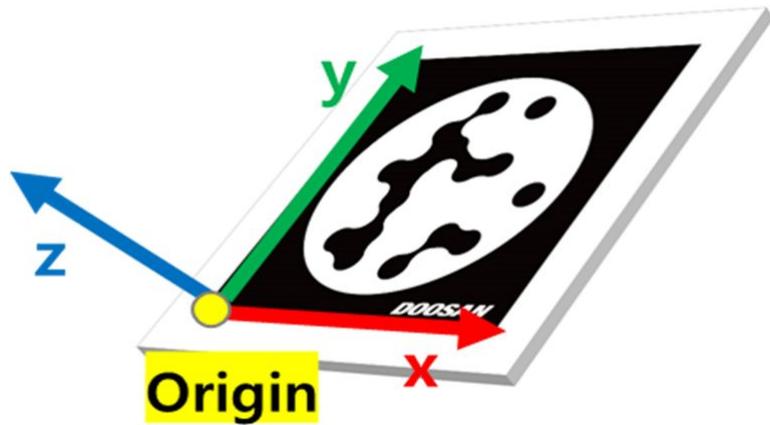
1  svm_connect()                                # Connect to vision
2  count, cpos = svm_detect_landmark( 1000 )
3  tp_popup( "Landmark number={0}, Landmark with respect to
Camera={1}" . format (count, cpos))
4  svm_disconnect()                            # Disconnect to vision

```

11.5.4 svm_get_marker_offset_pose(cpos, offset, euler_mode)

Features

Estimate the landmark pose with respect to Robot coordinate by using offset from the origin of the landmark.



Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------------|---------------|--|
| cpos | § list[Tx, Ty, Tz, Rx, Ry, Rz] | | Landmark pose based on camera coordinates |
| offset | list[Tx, Ty, Rz] | | Position and angle of rotation away from the landmark origin |
| euler_mode | boolean | | Orientation option for Landmark pose with respect to Robot coordinate: ZYZ for “True”, XYZ for “False” |

Return

| Value | Data Type | Description |
|-------|---|---|
| rpos | list[Tx, Ty, Tz, Rz, Ry, Rz] or list[Tx, Ty, Tz, Rx, Ry, Rz] | Landmark pose based on robot coordinates or Landmark pose with offset set |
| -1 | Int | Fail – no measurement data or input variable error |

Example

```
1    svm_connect()                                # Connect to vision
```

```

2   offset = [ 10 , - 20 , 45 ]
3   euler_mode = True
4   rpos = svm_get_marker_offset_pose(cpos, offset, euler_mode)
5   tp_popup( "Landmark with respect to Robot={0}" . format (rpos))
6   svm_disconnect()                                     # Disconnect to
7   vision

```

11.5.5 svm_get_detect_barcode()

Features

It detects barcodes and QR codes displayed on the screen.

Return

| Value | Data Type | Description |
|---------|-----------|--|
| ret | int | 1 on success - Barcode detected |
| Btype | string | Detected barcode type |
| bdata | string | Information contained in the detected barcode |
| 0,[],[] | Int | Fail – no measurement data or input variable error |

Example

```

1   svm_connect()                                     # Connect to vision
2   ret, btype, bdata = svm_detect_barcode()
3   tp_popup( "Detection={0}, Type={1}, Data={2}" . format (ret,btype,
4   bdata))                                         # Disconnect to vision
   svm_disconnect()

```

11.5.6 svm_get_barcode_db_data(index)

Features

Returns the value in Barcode DB in SVM using index.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--------------------------|
| index | int | | index(ex - 1, 2, 3, ...) |

Return

| Value | Data Type | Description |
|---------|-----------|--|
| db_data | string | Return the value in Barcode DB using index. |
| “-1” | string | If you use the wrong index or there is no information in the DB. |

Example

```

1  svm_connect()                                # Connect to vision
2  db_data = svm_get_barcode_db_data( 1 )
3  tp_popup( "DB Data={0}" . format (db_data))
4  svm_disconnect()                            # Disconnect to vision

```

11.5.7 svm_compare_barcode_db_data(dbdata)

Features

After comparing the entered argument with the barcode DB stored in the SVM, check whether it is a stored value.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| dbdata | string | | Information contained in the barcode to be used for comparison. |

Return

| Value | Data Type | Description |
|--------|-----------|---|
| 0 or 1 | int | <p>After comparison, check whether the value is stored in Barcode DB in SVM</p> <p>When it is '0', the value is not stored in the DB.</p> <p>When it is '1', the value stored in the DB</p> |

Example

```

1 svm_connect()                                # Connect to vision
2 ret  = svm_compare_barcode_db_data(bdata)
3 tp_popup( "Comparison result={0}" . format (ret))
4 svm_disconnect()                            # Disconnect to vision

```

11.6 Robot Task

11.6.1 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

```

1  svm_connect()                      # Connect to vision
2  vision_test = 1000                 # Define vision job
3  ID
4  count = svm_get_vision_info(vision_test)      # Execute the
vision measurement
5  if (count == 1):                   # Check the
result
6      # Get the position information (posx) of vision_test tool
7  pos_result = svm_get_variable(vision_test, POSX_TYPE)
8  tp_popup( "{0}" . format (pos_result))
9  # Get the vision guided robot pose
10 ld_list = [vision_test]
11   pos_list = [pos_result]
12   svm_set_init_pos_data(Id_list,pos_list)
13   rob_posx = svm_get_offset_pos(posx( 200 , 200 , 100 , 0 , 180 , 0
, vision_test)
14     tp_popup( "{0}" . format (rob_posx))
15     # move to the rob_posx
16     movel(rob_posx, vel = 30 , acc = 100 )
svm_disconnect()                      # Disconnect to
vision

```

11.6.2 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) (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

```

1  svm_connect()                                     # Connect to vision
2  vision_test = 1000                               # Define vision
3  job_ID
4  count = svm_get_vision_info(vision_test)         # Execute the
vision measurement
5  if (count == 1):                                 # Check the
result
6      # Get the position information (posx) of vision_test tool
7      pos_result = svm_get_variable(vision_test, POSX_TYPE)
8      tp_popup( "{0}" . format (pos_result))
9      # Get the vision guided robot pose
10     Id_list = [vision_test]
11     pos_list = [pos_result]
12     svm_set_init_pos_data(Id_list, pos_list)

13
14     rob_posx = svm_get_offset_pos(posx( 200 , 200 , 100 , 0 , 180 , 0
, vision test)
15     tp_popup( "{0}" . format (rob_posx))
16     # move to the rob_posx
17     movel(rob_posx, vel = 30 , acc = 100 )
18     svm_disconnect()                                # Disconnect to
vision

```

11.7 Other SVM commands

11.7.1 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

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

Example

```

1  svm_set_tp_popup( 0 )          # Hide tp_popup
2  svm_connect()                 # Connect to vision
3  svm_disconnect()              # Disconnect to vision

```

11.8 Integrated example (SVM)

11.8.1 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.

```

1  svm_connect()                  # Connect to vision

```

```

2      vision_test = 1000                      # Define vision job ID
3      print_insp = 1001                        # Define inspection tool
4      ID
5      box_size = 1002                         # Define measurement tool
6      ID
7      svm_set_job(vision_test)                # Load the vision_test
8      (1000)
9      movej(svm_get_robot_pose(vision_test), vel = 10 , acc = 20 )  # Move to shoot pose (movej)
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

```

```

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(Id_list,pos_list)

rob_posx = svm_get_offset_pos(posx( 200 , 200 , 100 , 0 , 180 ,
, 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

```

12 Application Commands

12.1 External Encoder Setting Commands

12.1.1 `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.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| channel | int | 1 | Encoder channel (1, 2) 1: Channel 1 2: Channel 2 |
| polarity_A | int | 0 | Polarity of phase A (0: Phase A, 1: /Phase A) |
| polarity_B | int | 0 | Polarity of phase B (0: Phase B, 1: /Phase B) |
| polarity_Z | int | 0 | Trigger method of Phase Z (0: Falling edge, 1: Rising edge) |
| polarity_S | int | 0 | Trigger method of Phase S (0: Falling edge, 1: Rising edge) |

Return

| Value | Description |
|-------|-------------|
| N/A | Not Used |

Exception

| Exception | Description |
|-----------|-------------|
| N/A | Not Used |

Example

```

1  set_extenc_polarity( 1 , 0 , 1 , 0 , 1 )
2  # External Encoder channel 1 is set to phase A, /phase B, phase Z
   (falling edge), phase S (rising edge)

```

Related commands

[set_extenc_mode](#)

- [set_extenc_mode\(channel, mode_AB, pulse_AZ, mode_Z, mode_S, inverse_cnt\)\(p. 448\)](#)

12.1.2 [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.

¹⁾ Compared to versions prior to V2.7.0, Integrated mode_S parameter option - 1: Strobe Signal à Encoder Count Clear (conveyor tracking available with a single option of Encoder Count Clear), 2: works for Encoder Count Clear (for compatibility to prior version)

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| channel | int | 1 | Encoder channel (1, 2) 1: Channel 1 2: Channel 2 |
| mode_AB | int | 0 | Use of phase AB Mode (0 ~ 4) 0: Not Used 1: Phase A Quadrature use Phase B Quadrature use 2: Phase A Count Phase B Direction use 3: Phase A Up Count use Phase B Not Used 4: Phase A Down Count use Phase B Not Used |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| pulse_AZ | int | 0 | Pulse A Count per Pulse Z (0 ~ 100000) |
| mode_Z | int | 0 | Phase Z Use Mode (0 ~ 1) 0: Not Used 1: A/B Count Error Compensation 2: Encoder Count Clear |
| mode_S | int | 0 | Phase S Use Mode (0 ~ 1) 0: Not Used 1: Encoder Count Clear |
| inverse_cnt | int | 0 | Encoder Count Direction Reverse Status 0: Forward 1: Reverse |

Return

| Value | Description |
|-------|-------------|
| N/A | Not Used |

Exception

| Exception | Description |
|-----------|-------------|
| N/A | Not Used |

Example

```

1  set extenc mode( 1 , 2 , 20000 , 1 , 1 , 0 )
2  # External Encoder channel 1 operation mode is set as follows
3  # Phase A Count, Phase B Direction Use
4  # Pulse A Count per Z Pulse is 20000
5  # Phase Z error count accumulate compensation mode use, phase S
use
6  # Encoder Count direction is set to forward

```

Related commands

`set_extenc_polarity`

- [set_extenc_polarity\(channel, polarity_A, polarity_B, polarity_Z, polarity_S\)\(p. 447\)](#)

12.1.3 `get_extenc_count(channel)`

Features

Get the count value of the corresponding encoder channel.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| channel | int | 1 | Encoder channel (1, 2) 1: Channel 1 2: Channel 2 |

Return

| Value | Description |
|-------|--|
| count | Current encoder count value of corresponding channel |

Exception

| Exception | Description |
|-----------|-------------|
| N/A | Not Used |

Example

```

1   enc cnt  =  get extenc count( 1 )
2   # External Encoder channel 1 current count value calculation

```

Related commands

- [set_extenc_polarity\(channel, polarity_A, polarity_B, polarity_Z, polarity_S\)\(p. 447\)](#)
- [set_extenc_mode\(channel, mode_AB, pulse_AZ, mode_Z, mode_S, inverse_cnt\)\(p. 448\)](#)
- [clear_extenc_count\(channel\)\(p. 451\)](#)

12.1.4 clear_extenc_count(channel)

Features

Reset counter value of the corresponding encoder channel to 0.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| channel | int | 1 | Encoder channel (1, 2) 1: Channel 1 2: Channel 2 |

Return

| Value | Description |
|-------|-------------|
| N/A | Not Used |

Exception

| Exception | Description |
|-----------|-------------|
| N/A | Not Used |

Example

| | |
|---|---|
| 1 | clear extenc count(1) |
| 2 | # External Encoder channel 1 count value reset to 0 |

Related commands

- [get_extenc_count\(channel\)\(p. 450\)](#)

12.2 Conveyor Tracking

12.2.1 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.

Parameters

| 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 |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```
1 CONV1 = set_conveyor("conveyor1")
```

Related commands

- [get_conveyor_obj\(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None\)\(p. 457\)](#)
- [tracking_conveyor\(conv_id, time=0.3\)\(p. 461\)](#)
- [untracking_conveyor\(conv_id, time=0.3\)\(p. 463\)](#)

12.2.2 set_conveyor_ex(name=""", conv_type=0, encoder_channel=1, triggering_mute_time=0.0, count_per_dist=5000, conv_coord=posx(0,0,0,0,0,0), ref=DR_BASE, conv_speed=100.0, speed_filter_size=500, min_dist=0.0, max_dist=1000.0, watch_window=100.0...)

Features

Configures the conveyor and obtains Conveyor ID to allow the Conveyor Tracking Application to start. After the command is executed, it monitors workpieces triggered in the configured conveyor until the program ends. It can be used when you need to set parameters manually if is unavailable to configure conveyor information through UI.

¹⁾ Added default value for all arguments compared to versions prior to M2.4.0

²⁾ Added 'ref' argument compared to versions prior to M2.4.0 (world coordinates available)

³⁾ Removed 'obj_offset_coord' argument compared to versions prior to M2.4.0, The 'obj_offset_coord' argument is changed to input only in get_conveyor_obj() function.

Parameter

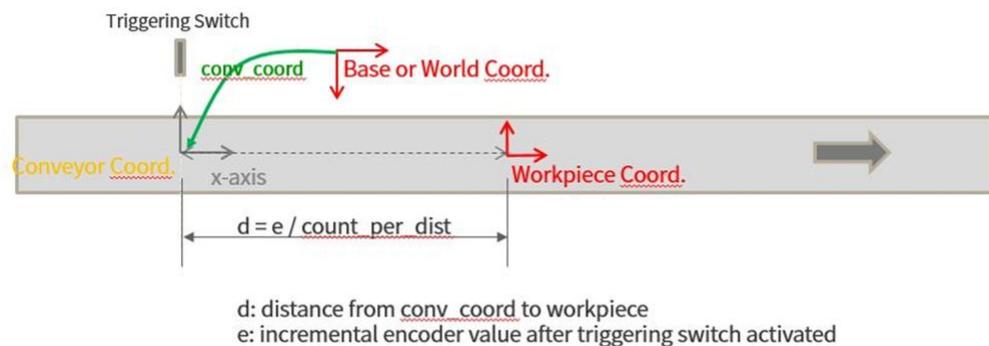
| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---------------------------------------|
| name | string | "" | Conveyor name |
| conv_type | int | 0 | Conveyor type(0: Linear, 1: Circular) |

| Parameter Name | Data Type | Default Value | Description |
|----------------------|-----------------|-------------------|---|
| 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: °) |
| max_dist | float | 1000.0 | Maximum conveyor work length (based on Triggering Switch, Linear: mm, Circular: °) |

| Parameter Name | Data Type | Default Value | Description |
|-------------------|-----------|---------------|---|
| watch_window | float | 100.0 | Conveyor work standby monitoring length (based on minimum work length, Linear: mm, Circular: °) |
| out_tracking_dist | float | 10.0 | Conveyor tracking release buffer section length (based on maximum work length, Linear: mm, Circular: °) |

Note

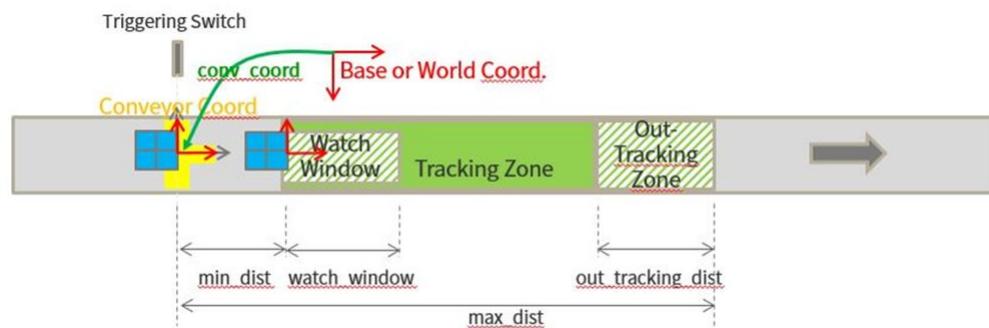
- Currently conv_type argument does not support Circular Conveyors!
- All workpieces that pass the Triggering Switch are monitored until they reach max_dist after set_conveyor() or set_conveyor_ex() function execution and before the program ends.
- However, if triggering_mute_time is configured, and if the Triggering Switch activates during the corresponding time after the previous workpiece is detected, it is not included on the monitoring list. It is used when noise is present in the Triggering Switch or when the workpiece needs to be removed for a certain amount of time.
- conv_coord is a coordinate system fixed to the conveyor relative to the base or world coordinate system. **Here, the x-axis of conv_coord represents the direction the conveyor flows.** From the moment the conveyor workpiece activates the triggering switch, the increased encoder value can be converted to the length of the workpiece travel by using the count_per_dist argument, and extending this length in the x-axis direction of the conv_coord will position the workpiece relative to the reference coordinate system.



[Conveyor/Item Coordinate]

- conv_speed is the moving speed of the conveyor. It is used to give Info only if the conveyor speed sensed by the encoder exceeds 200% of this speed. Therefore, if the measurement is not possible through the TP UI, enter the approximate value.
- Speed_filter_size is the size of the moving-average filter used to estimate the conveyor speed from the encoder. The larger the size, the more the noise can be canceled, but the tracking accuracy may deteriorate during acceleration and deceleration.

- The area on top of the conveyor is categorized into Watch Window, Tracking Zone and Out-Tracking Zone.
- Watch Window is the area that determines whether workpieces within the area are available for the job when obtaining workpiece coordinates for tracking. When the `get_conveyor_obj()` function is loaded, if a workpiece is not present within this area, the function is not returned, and if a workpiece is present within this area, it returns workpiece coordinates according to `get_conveyor_obj()` function options (FIFO, LIFO).
- The Tracking Zone is the area that performs Conveyor Tracking.
- The Out-Tracking Zone is the area where the robot automatically ends tracking after it determines that the robot has exited the work space of the robot or the work space specified by the user during continuous tracking.
- These three areas are defined with the four lengths (`min_dist`, `max_dist`, `watch_window`, `out_tracking_dist`) as shown below.



[Conveyor Area and Length]

Return

| Value | Description |
|-------------|---|
| Conveyor ID | Returns Conveyor ID if conveyor setting is successful |
| None | Conveyor setting failure |

Exception

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

```

1 CONV1 = set_conveyor_ex(name = 'conveyor_1',
2 conv_type = 0, # linear
3 encoder_channel = 1, triggering_mute_time = 0.0,
4 count_per_dist = 5000, # 5000 count/mm)
5 conv_coord = posx( 500, 100, 500, 0, -90, 0), ref = DR
6 BASE,
7 conv_speed = 100.0, # conveyor speed: 100 mm/s.
8 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\(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None\)\(p. 457\)](#)
- [tracking_conveyor\(conv_id, time=0.3\)\(p. 461\)](#)
- [untracking_conveyor\(conv_id, time=0.3\)\(p. 463\)](#)

12.2.3 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.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| conv_id | int | - | Conveyor ID |

| Parameter Name | Data Type | Default Value | Description |
|------------------|----------------|---------------|---|
| timeout | float | None | If there is no workpiece to return during this timeout, it ends standby and return the function |
| container_type | int | DR_FIFO | Workpiece container type (DR_FIFO: first-in/first-out, DR_LIFO: last-in/last-out) |
| obj_offset_coord | posx | None | Workpiece coordinates (mm, °) based on conveyor lock coordinates |
| | list(float[6]) | | |

Note

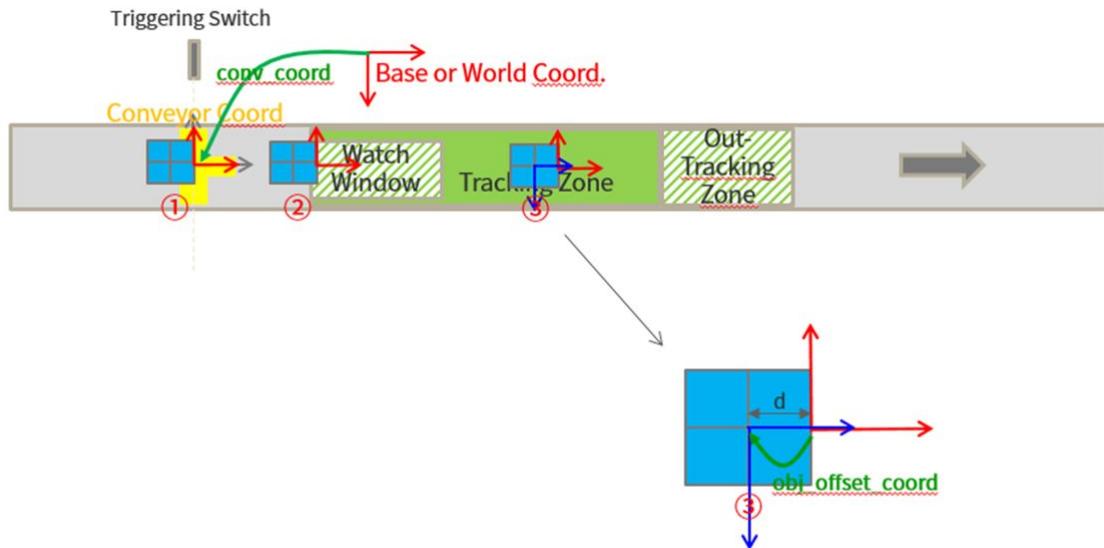
- When calling this function, it returns the coordinates ID of each workpiece in the Watch Window according to the container rule. For example, if you call get_conveyor_obj() function when the workpieces are places as shown below, the workpiece ② and ③ in the Watch Window will be candidates. At this time, if the container_type is set to DR_FIFO the 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]

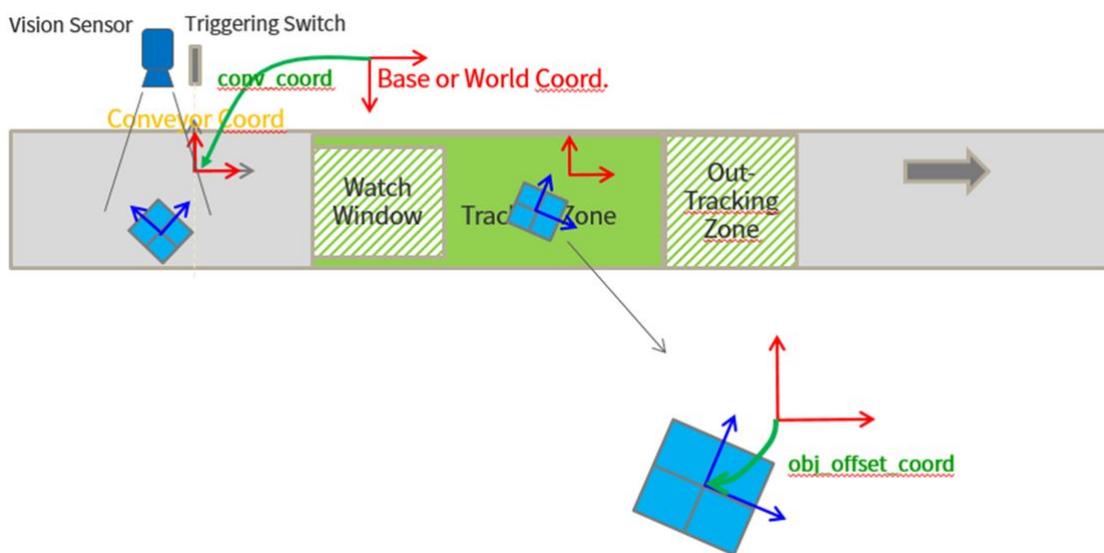
- obj_offset_coord is used when you want to apply offset to the workpiece coordinates. It is usually used for easy input of a teaching point or when you want to dynamically change the position and orientation of the workpiece coordinates in conjunction with an external sensor (ex. Vision sensor).

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 obj_offset_coord = posx (-d, 0, 0, -90, 0, 0). It is not necessary to acquire a teaching point through this TP UI, but it could utilize this method if you need to use drl only or enter the teaching point directly.



[obj_offset_coord use case 1]

Next, if the workpiece changes its position in a direction that is independent of the conveying direction, or the orientation of the workpiece changes as shown below, the encoder signal alone cannot determine the position / orientation of the workpiece. In this case, you need to detect them using external vision sensor. After detecting this value, you can input the position/orientation change detected as `obj_offset_coord` dynamically and the workpiece coordinates are created accordingly.



[obj_offset_coord use case 2: using vision sensor]

Return

| Value | Description |
|------------------|---|
| int | CONV_COORD. Conveyor user coordinate ID (121~150) |
| Negative integer | If no workpiece is present even after the timeout expires |

i Note

If no workpiece to return is present, no function is returned until the timeout time expires. If the timeout time expires but no workpiece is present, it returns -1. However, if a timeout time is not entered, it doesn't return continuously.

Exception

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error occurred |
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  ## One object in a cycle
2  CONV1 = set_conveyor('conveyor1')
3
4  movel(posx( 100 , 100 , 50 , 0 , 0 , 0 ), ref = DR_BASE) # waiting position
5  while True :
6      CONV_COORD_1 = get_conveyor_obj(CONV1)
7      tracking_conveyor(CONV1)
8
9  # synched motion
10 movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)

```

```

11 movel(posx( 0 , 0 , 0 , 0 , 0 , 0 ), ref = CONV_COORD_1)
12 set_digital_output(DO_GRIPPER, 1 )
13 movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
14
15 untracking_conveyor(CONV1)
16
17 movel(posx( 100 , 100 , 50 , 0 , 0 , 0 ), ref = DR_BASE) # waiting position
18
19 ## Multi objects in a cycle
20 CONV1 = set_conveyor('conveyor1')
21
22 while True :
23     CONV_COORD_1 = get_conveyor_obj(CONV1)
24     tracking_conveyor(CONV1)
25
26 # fist object
27 movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
28 movel(posx( 0 , 0 , 0 , 0 , 0 , 0 ), ref = CONV_COORD_1)
29 set_digital_output(DO_GRIPPER, 1 )
30 movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
31
32 # second object
33 CONV_COORD_2 = get_conveyor_obj(CONV1, time_out = 10 )
34 if CONV_COORD_2 > 0 : # -1 if no objects available during time_out
35     movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_2)
36     movel(posx( 0 , 0 , 0 , 0 , 0 , 0 ), ref = CONV_COORD_2)
37     set_digital_output(DO_GRIPPER, 1 )
38     movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_2)
39
40 # first object if you need
41 movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
42
43 untracking_conveyor(CONV1)
44
45 movel(posx( 100 , 100 , 50 , 0 , 0 , 0 ), ref = DR_BASE)

```

Related commands

- [tracking_conveyor\(conv_id, time=0.3\)\(p. 461\)](#)
- [untracking_conveyor\(conv_id, time=0.3\)\(p. 463\)](#)

12.2.4 tracking_conveyor(conv_id, time=0.3)

Features

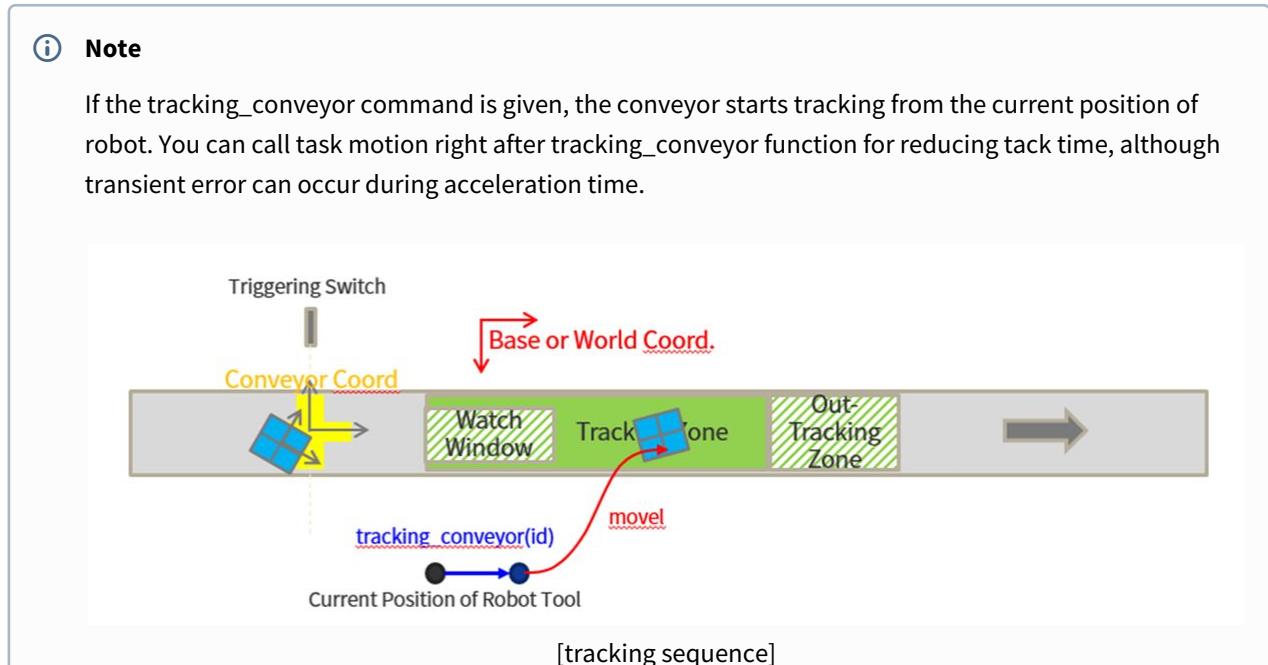
The robot starts Conveyor Tracking.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| conv_id | int | - | Conveyor ID |
| time | float | 0.3 | Acceleration time(sec) to start Tracking |

(i) Note

If the tracking_conveyor command is given, the conveyor starts tracking from the current position of robot. You can call task motion right after tracking_conveyor function for reducing task time, although transient error can occur during acceleration time.



Return

| Value | Description |
|------------------|---|
| 0 | Conveyor Tracking success |
| Negative integer | If the robot is expected to exit the robot work space during acceleration |

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------------|
| DR_Error (DR_ERROR_VALUE) | Parameter value is invalid |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1 CONV1 = set_conveyor('conveyor1')
2
3 while True :
4     CONV_COORD_1 = get_conveyor_obj(CONV1)
5
6     tracking_conveyor(CONV1) # start moving to track conveyor
7
8     # task on conveyor
9     movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
10    movel(posx( 0 , 0 , 0 , 0 , 0 , 0 ), ref = CONV_COORD_1)
11    set_digital_output(DO_GRIPPER, 1)
12    movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
13
14    untracking_conveyor(CONV1)
15    obj_count = obj_count + 1

```

Related commands

- [untracking_conveyor\(conv_id, time=0.3\)\(p. 463\)](#)

12.2.5 untracking_conveyor(conv_id, time=0.3)

Features

The robot moves as its velocity goes to 0 and finish Conveyor Tracking.

Parameters

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| conv_id | int | - | Conveyor ID |
| time | float | 0.3 | Deceleration time (sec) to end Tracking |

Note

- If a time value is shorter than the robot's maximum deceleration speed, the robot ignores the entered value and decelerates using the maximum deceleration speed.
- To reduce tack-time, deceleration motion is blended with task motion after untracking_conveyor if it is called. (However, Joint motion cannot be called during deceleration time.)

Return

| Value | Description |
|------------------|---|
| 0 | Finishing Conveyor Tracking success |
| Negative integer | If the robot is expected to exit the robot work space during deceleration |

Exception

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

Example

```

1 CONV1 = set_conveyor('conveyor1')
2
3 while True :
4     CONV_COORD_1 = get_conveyor_obj(CONV1)
5     tracking_conveyor(CONV1)
6
7     # task on conveyor
8     movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
9     movel(posx( 0 , 0 , 0 , 0 , 0 , 0 ), ref = CONV_COORD_1)
10    set_digital_output(DO_GRIPPER, 1 )
11    movel(posx( 0 , 0 , 50 , 0 , 0 , 0 ), ref = CONV_COORD_1)
12
13    untracking_conveyor(CONV1, 0.1 )

```

Related commands

- [tracking_conveyor\(conv_id, time=0.3\)\(p. 461\)](#)

12.3 Welding

12.3.1 app_weld_enable_digital()

Features

This enables the communication interface welding function. Only supports EtherNet/IP interface.

Return

| Value | Description |
|----------------|------------------------|
| 0 | Enable Welding Success |
| Negative Value | Enable Welding Failure |

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 app_weld_enable_digital()
2 app_weld_disable_digital()
```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\], wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)\(p. 491\)](#)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\], ...\)\(p. 494\)](#)
- [app_weld_enable_digital\(\)\(p. 465\)](#)

- [app_weld_set_weld_cond_digital\(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...\)\(p. 498\)](#)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)\(p. 503\)](#)
- [app_weld_disable_digital\(\)\(p. 467\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.2 app_weld_disable_digital()

Features

This disables the communication interface welding function.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

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

Example

| | |
|---|---|
| 1 | <code>app_weld_enable_digital()</code> |
| 2 | <code>app_weld_disable_digital()</code> |

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...,opt7=\[0,0,0,0,0,0,0,0\]\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)\(p. 491\)](#)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\], ...,opt8=\[0,0,0,0,0,0,0,0\]\)\(p. 494\)](#)
- [app_weld_enable_digital\(\)\(p. 465\)](#)
- [app_weld_set_weld_cond_digital\(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...\)\(p. 498\)](#)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)\(p. 503\)](#)
- [app_weld_disable_digital\(\)\(p. 467\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.3 app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0, 0,0,0,0], robot_ready=[0,0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the link signal interface between the robot controller and welder used for welding in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

app_weld_set_interface_eip_r2m_process(), app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(), app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(), app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(), app_weld_set_interface_eip_m2r_other()

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------|--------------------------|--|
| welding_start | Refer to the table below | Refer to the table below | Start Weld Command (specification for each welder) |
| robot_ready | | | Robot Status (specification for each welder) |
| error_reset | | | Reset Welder Error (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|---------------|--|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | list(float[2]) | 0 | Minimum Data Value |
| | | 0 | Maximum Data Value |

(i) Note

Communication Interface Setting Example (EWM Welder)

1. Data Type (on/off: 0): Item selected On/Off

a. Ewm Welder Data Sheet

| Byte no. | Bit no. | Function/description | Bit assignment |
|----------|---------|------------------------------|---------------------------------|
| 0 | 4 | Start signal welding process | 0 switched off 1 switched on |

b. Specification Entry Method

| Item | Setting Value |
|--------------|---------------|
| Usage Status | 1 (Used) |
| Data Type | 0 (on/off) |
| Data Digits | 0 (1) |

| Item | Setting Value |
|---------------------------------|------------------------|
| Communication Data Point (byte) | 0 |
| Communication Data Point (bit) | 4 |
| Data Size | 0 (1-bit, disable Low) |
| Valid Data Size | 1 (1 bit) |
| Minimum Data Value | 0 |
| Maximum Data Value | 1 |

2. Data Type (Select: 1): If data with an integer of 1 is selected

1. Ewm Welder Data Sheet

| Byte no. | Bit no. | Function/description | Bit assignment |
|-----------------|----------------|-----------------------------|-----------------------|
| 3 | 0-7 | Selection JOB | Range 1-255 |

b. Specification Entry Method

| Item | Setting Value |
|---------------------------------|----------------------|
| Usage Status | 1 (Used) |
| Data Type | 1 (Select) |
| Data Digits | 0 (1) |
| Communication Data Point (byte) | 3 |
| Communication Data Point (bit) | 0 |
| Data Size | 4 (8-bit) |
| Valid Data Size | 8 (8 bit) |
| Minimum Data Value | 0 |
| Maximum Data Value | 255 |

3. Data Type (Value: 2): If a real number value is entered

1. Ewm Welder Data Sheet

| Byte no. | Bit no. | Function/description | Bit assignment |
|-----------------|----------------|---------------------------------------|--|
| 6 | 0-15 | Welding voltage(current actual value) | 0 to 0x7FFF (High-Byte first) equivalent to 0.0V to 100.0V |

b. Specification Entry Method

| Item | Setting Value |
|---------------------------------|----------------------|
| Usage Status | 1 (Used) |
| Data Type | 2 (Value) |
| Data Digits | 1 (0.1) |
| Communication Data Point (byte) | 6 |
| Communication Data Point (bit) | 0 |
| Data Size | 6 (16-bit) |
| Valid Data Size | 15 (15 bit) |
| Minimum Data Value | 0.0 (V) |
| Maximum Data Value | 100.0 (V) |

- If the data type is 2 (Value), a valid data size (0x7FFF → 15 bit), minimum data value (0.0V) and maximum data value (100.0V) must be entered.

Return

| Value | Description |
|----------------|--------------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|------------------------------------|--------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |

| Exception | Description |
|-----------------------------|-----------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

| | |
|---|---|
| 1 | app_weld_set_interface_eip_r2m_process(welding_start = [1 , 0 , 0 , 0 , 4 , 0 , 1 , 0 , 0], robot_ready = [1 , 0 , 0 , 0 , 5 , 0 , 1 , 0 , 0], error_reset = [1 , 0 , 0 , 1 , 4 , 0 , 1 , 0 , 0]) |
|---|---|

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0\]\)](#)(p. 469)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)](#)(p. 474)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)](#)(p. 477)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)](#)(p. 480)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...\)](#)(p. 483)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)](#)(p. 487)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)](#)(p. 491)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\], ...\)](#)(p. 494)

12.3.4 app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0,0],wm_opt1=[0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the welding mode in the communication data sent to the welder from the robot controller. Required modes can be additionally added with the option item (wm_opt1). Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

app_weld_set_interface_eip_r2m_process(), app_weld_set_interface_eip_r2m_mode(),
 app_weld_set_interface_eip_r2m_test(), app_weld_set_interface_eip_r2m_condition(),
 app_weld_set_interface_eip_r2m_option(), app_weld_set_interface_eip_m2r_process(),
 app_weld_set_interface_eip_m2r_monitoring(), app_weld_set_interface_eip_m2r_other()

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------|--------------------------|--|
| welding_mode | Refer to the table below | Refer to the table below | Welding Mode (specification for each welder) |
| s_2t | | | Latched/Non-latched Mode (specification for each welder) |
| pulse_mode | | | Pulse Mode (specification for each welder) |
| wm_opt1 | | | Option Mode (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|---------------|--|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | list(float[2]) | 0 | Minimum Data Value |
| | | 0 | Maximum Data Value |

 **Note**

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```
1 app_weld_set_interface_eip_r2m_mode(welding_mode = [ 1 , 1 , 0 , 0
, 0 , 2 , 2 , 0 , 3 ], s_2t = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, pulse_mode = [ 1 , 1 , 0 , 0 , 2 , 0 , 1 , 0 , 1 ],wm_opt1 = [ 0 ,
])
```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)

- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0], ...)(p. 494)`

12.3.5 `app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)`

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the test signal setting in the communication data sent to the welder from the robot controller. Additional functions related to the test signal can be added with the option item (ts_opt1, ts_opt2). Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

`app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`,
`app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`,
`app_weld_set_interface_eip_r2m_option()`, `app_weld_set_interface_eip_m2r_process()`,
`app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------|--------------------------|--|
| gas_test | Refer to the table below | Refer to the table below | Gas Test Signal (specification for each welder) |
| Inching_plus | | | Forward Inching Signal (specification for each welder) |
| Inching_minus | | | Reverse Inching Signal (specification for each welder) |
| blow_out_torch | | | Torch Cleaning Signal (specification for each welder) |

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|---|
| simulation | | | Mock Welding Signal (specification for each welder) |
| ts_opt1 | | | Option Signal (specification for each welder) |
| ts_opt2 | | | Option Signal (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|---|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | | 0 | Minimum Data Value |
| | | 0 | Maximum Data Value |

 **Note**

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|---|-----------------------------|
| <code>DR_Error</code> (<code>DR_ERROR_TYPE</code>) | Parameter data error |
| <code>DR_Error</code> (<code>DR_ERROR_VALUE</code>) | Invalid parameter value |
| <code>DR_Error</code> (<code>DR_ERROR_RUNTIM E</code>) | C Extension module error |
| <code>DR_Error</code> (<code>DR_ERROR_STOP</code>) | Program terminated forcibly |

Example

```
1   app_weld_set_interface_eip_r2m_test(gas_test = [ 1 , 0 , 0 , 0 , 0 , 6
, 0 , 1 , 0 , 0 ], inching_plus = [ 1 , 0 , 0 , 1 , 0 , 0 , 1 , 0 ,
, inching_minus = [ 1 , 0 , 0 , 1 , 2 , 0 , 1 , 0 , 0 ], blow_out_torch = [ 1 . 0 . 0 . 0 . 7 . 0 . 1 . 0 . 0 ], simulation = [ 0 , 0 , 0 , 1 , 7 , 0 , 1 , 0 , 0 ], ts_opt1 = [ 0 , 0
, ts_opt2 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ])
```

Related commands

- `app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0,0,0], robot_ready=[0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0])(p. 469)`

- `app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0,0], wm_opt1=[0,0,0,0,0,0,0,0])(p. 474)`
- `app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)(p. 477)`
- `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])(p. 480)`
- `app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], ...)(p. 483)`
- `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0])(p. 487)`
- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0], ...)(p. 494)`

12.3.6 `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])`

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the welding condition setting in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

```
app_weld_set_interface_eip_r2m_process(), app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(), app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(), app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(), app_weld_set_interface_eip_m2r_other()
```

Parameter

| Parameter Name | Data Type | Default Value | Description |
|------------------|--------------------------|--------------------------|---|
| job_num | Refer to the table below | Refer to the table below | JOB Number (specification for each welder) |
| synergic_id | | | SYNERGIC Number (specification for each welder) |
| r_wire_feed_sped | | | Wire Feeding Speed Correction (specification for each welder) |
| voltage_correct | | | Voltage Correction (specification for each welder) |
| dynamic_correct | | | Dynamic Correction (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|---------------|---|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|--------------------------------|
| | list(float[2]) | 0 | Valid Data Size Value (bit) |
| | | 0 | Minimum Data Value |
| | | 0 | Maximum Data Value |

 **Note**

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|---|-----------------------------|
| <code>DR_Error</code> (<code>DR_ERROR_TYPE</code>) | Parameter data error |
| <code>DR_Error</code> (<code>DR_ERROR_VALUE</code>) | Invalid parameter value |
| <code>DR_Error</code> (<code>DR_ERROR_RUNTIM E</code>) | C Extension module error |
| <code>DR_Error</code> (<code>DR_ERROR_STOP</code>) | Program terminated forcibly |

Example

```
1 app_weld_set_interface_eip_r2m_condition(iob_num = [ 1 , 1 , 0 , 3
, 0 , 4 , 8 , 0 , 255 ], synergic_id = [ 1 , 1 , 0 , 2 , 0 , 3 , 4 ,
5 ], r_wire_feed_speed = [ 1 , 2 , 1 , 6 , 0 , 6 , 15 , 0.0 , 25.0 ]
, voltage_correct = [ 1 , 2 , 1 , 8 , 0 , 6 , 15 , - 10.0 , 10.0 ],
dynamic_correct = [ 1 , 2 , 0 , 10 , 0 , 6 , 15 , - 40 , 40 ])
```

Related commands

- `app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0,0,0,0], robot_ready=[0,0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0,0])(p. 469)`
- `app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0,0],wm_opt1=[0,0,0,0,0,0,0,0])(p. 474)`
- `app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)(p. 477)`
- `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_correct=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])(p. 480)`
- `app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], ...)(p. 483)`
- `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0])(p. 487)`
- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0], ...)(p. 494)`

12.3.7 app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], ...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication.

Required functions other than basic setting items (`app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`, `app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`) in the communication data sent to the welder from the robot

controller can be set with the corresponding command. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

 Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

`app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`,
`app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`,
`app_weld_set_interface_eip_r2m_option()`, `app_weld_set_interface_eip_m2r_process()`,
`app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------|--------------------------|---|
| opt1 | Refer to the table below | Refer to the table below | Option Item (specification for each welder) |
| opt2 | | | |
| opt3 | | | |
| opt4 | | | |
| opt5 | | | |
| opt6 | | | |
| opt7 | | | |
| opt8 | | | |
| opt9 | | | |
| opt10 | | | |
| opt11 | | | |
| opt12 | | | |
| opt13 | | | |

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|--------------------|
| opt14 | | | |
| opt15 | | | |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|---|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | | list(float[2]) | Minimum Data Value |
| | | 0 | Maximum Data Value |

Note

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|--|-----------------------------|
| <code>DR_Error (DR_ERROR_TYPE)</code> | Parameter data error |
| <code>DR_Error (DR_ERROR_VALUE)</code> | Invalid parameter value |
| <code>DR_Error (DR_ERROR_RUNTIME)</code> | C Extension module error |
| <code>DR_Error (DR_ERROR_STOP)</code> | Program terminated forcibly |

Example

```
1 app_weld_set_interface_eip_r2m_option(opt1 = [ 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 ], opt2 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
opt3 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], opt4 = [ 0 , 0 , 0 , 0 ,
, opt5 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
, opt6 = [ 0 , 0 , 0 , 0 ,
, opt7 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
, opt8 = [ 0 , 0 , 0 ,
, opt9 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
, opt10 = [ 0 , 0 , 0 ,
, opt11 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
, opt12 = [ 0 , 0 , 0 ,
, opt13 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
, opt14 = [ 0 , 0 , 0 ,
, opt15 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ]])
```

Related commands

- `app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0,0,0], robot_ready=[0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0])(p. 469)`
- `app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0],wm_opt1=[0,0,0,0,0,0,0])(p. 474)`

- `app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)(p. 477)`
- `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])(p. 480)`
- `app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], ...)(p. 483)`
- `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0])(p. 487)`
- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0], ...)(p. 494)`

12.3.8 `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0])`

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. The link signal interface between the controller and welder in the communication data sent to the welder from the robot controller can be set. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

1. To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up. `app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`, `app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`, `app_weld_set_interface_eip_r2m_option()`, `app_weld_set_interface_eip_m2r_process()`, `app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`
2. Start robot motion links with the `current_flow` signal from the welder but is linked with the corresponding signal when the `main_current` item is set.
3. End robot motion links with the `current_flow` signal from the welder but is linked with the corresponding signal when the `process_active` item is set.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------------------|--------------------------|---|
| current_flow | Refer to the table below | Refer to the table below | Welding Current Generated (specification for each welder) |
| process_active | | | Welding Process Activated (specification for each welder) |
| main_current | | | Regular Welding Current Generated (specification for each welder) |
| machine_ready | | | Welding Standby (specification for each welder) |
| comm_ready | | | Communication Standby (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|---------------|--|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | list(float[2]) | 0 | Minimum Data Value |
| | | 0 | Maximum Data Value |

(i) Note

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```
1 app_weld_set_interface_eip_m2r_process(current_flow = [ 1 , 0 , 0
, 0 , 0 , 0 , 1 , 0 , 0 ], process_active = [ 1 , 0 , 0 , 0 , 0 , 6 , 0
, main_current = [ 1 , 0 , 0 , 0 , 5 , 0 , 1 , 0 , 0 ], machine_ready = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], comm_ready = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ])
```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0.0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\],...\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)\(p. 491\)](#)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\],...\)\(p. 494\)](#)

12.3.9 app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to monitoring of the welding machine's status setting in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

`app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`,
`app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`,
`app_weld_set_interface_eip_r2m_option()`, `app_weld_set_interface_eip_m2r_process()`,
`app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`

Parameter

| Parameter Name | Data Type | Default Value | Description |
|-----------------|--------------------------|--------------------------|---|
| welding_voltage | Refer to the table below | Refer to the table below | Actual Welding Voltage (specification for each welder) |
| welding_current | | | Actual Welding Current (specification for each welder) |
| wire_feed_speed | | | Actual Wire Feeding Speed (specification for each welder) |
| wire_stick | | | Check Wire Stick Status (specification for each welder) |
| error | | | Check Error Status (specification for each welder) |
| error_num | | | Check Error Number (specification for each welder) |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|----------------|---|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | | list(float[2]) | Minimum Data Value |
| | | 0 | Maximum Data Value |

(i) Note

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```
1 app_weld_set_interface_eip_m2r_monitoring(welding_voltage = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 ], welding_current = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 ], wire_feed_speed = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], wire_stick = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], error = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], error_num = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ])
```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)

- `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0])(p. 480)`
- `app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0], ...)(p. 483)`
- `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0])(p. 487)`
- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0], ...)(p. 494)`

12.3.10 `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0], ...)`

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication.

Required functions other than basic setting items (`app_weld_set_interface_eip_m2r_process()`, `app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`, `app_weld_set_interface_eip_r2m_condition()`) in the communication data sent to the robot controller from the welder can be set with the corresponding command. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

(i) Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

`app_weld_set_interface_eip_r2m_process()`, `app_weld_set_interface_eip_r2m_mode()`,
`app_weld_set_interface_eip_r2m_test()`, `app_weld_set_interface_eip_r2m_condition()`,
`app_weld_set_interface_eip_r2m_option()`, `app_weld_set_interface_eip_m2r_process()`,
`app_weld_set_interface_eip_m2r_monitoring()`, `app_weld_set_interface_eip_m2r_other()`

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|---------------|---------------|-------------|
| opt1 | Specification | | |
| opt2 | Specification | | |
| opt3 | Specification | | |
| opt4 | Specification | | |
| opt5 | Specification | | |
| opt6 | Specification | | |
| opt7 | Specification | | |
| opt8 | Specification | | |
| opt9 | Specification | | |
| opt10 | Specification | | |

The data type, default value and description are identical to the below

| Parameter Name | Data Type | Default Value | Description |
|----------------|--------------|----------------|---|
| | list(int[7]) | 0 | Not Used: 0 Used: 1 |
| | | 0 | Data Type (on/off: 0, Select: 1, Value: 2) |
| | | 0 | Data Digits (1: 0, 0.1: 1, 0.01: 2) |
| | | 0 | Communication Data Point (byte): 1~255 |
| | | 0 | Communication Data Point (bit): 1~255 |
| | | 0 | Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7 |
| | | 0 | Valid Data Size Value (bit) |
| | | list(float[2]) | Minimum Data Value |
| | | 0 | Maximum Data Value |

 **Note**

For examples of data (0~2) interface settings, refer to the `app_weld_set_interface_eip_r2m_process()` section.

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1   app_weld_set_interface_eip_m2r_other(opt1 = [ 1 , 2 , 1 , 12 , 0 ,
6 , 15 , 0.0 , 25.5 ], opt2 = [ 1 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 ],
, opt3 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], opt4 = [ 0 , 0 , 0 ,
, opt5 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], opt6 = [ 0 , 0 , 0 ,
, opt7 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], opt8 = [ 0 , 0 , 0 ,
, opt9 = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ], opt10 = [ 0 , 0 , 0 ,
)

```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0\]\)](#)(p. 469)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)](#)(p. 474)

- `app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0], ...)(p. 477)`
- `app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0], voltage_corret=[0,0,0,0,0,0,0,0], dynamic_correct=[0,0,0,0,0,0,0,0])(p. 480)`
- `app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], ...)(p. 483)`
- `app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0])(p. 487)`
- `app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0], ...)(p. 491)`
- `app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0], ...)(p. 494)`

12.3.11 `app_weld_set_weld_cond_digital(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...)`

Feature

This sets the welding condition of the remote control welders. It is only valid within the welding section defined with the Enable Welding (`app_weld_enable_digital()`) and Disable Welding (`app_weld_disable_digital()`) commands, and any operations starting at a point outside the welding section will generate an error. Items that can be set as welding conditions are welders corresponding to the following commands (`app_weld_set_interface_eip_r2m_mode()`, `app_weld_set_interface_eip_r2m_condition()`, `app_weld_set_interface_eip_r2m_option()`) and items that completed the communication interface setting.

Only one welding condition is allowed in a single welding section. This welding condition can be adjusted during welding with the `app_weld_adj_welding_cond_digital()` command while the voltage correction/dynamic correction/feeding speed/speed (and weaving offset) can be also adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with `app_weld_set_weld_cond_digital()`).

(i) Note

1. Voltage Correction: Adjusts the length of the ark.
2. Dynamic Correction: Adjusts the ark property.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|-------------------|-----------|---------------|--|
| flag_dry_run | int | 0 | Dry-Run Mode Actual Welding (0) Dry-Run (1): Motion/Weaving/Offset only |
| vel_target | float | 0 | Target Speed (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| vel_min | float | 0 | Minimum Target Speed Correction (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| vel_max | float | 0 | Maximum Target Speed Correction (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| welding_mode | Int | 0 | Welding Mode Setting |
| s_2t | Int | 0 | 2T, 2T Special Setting |
| pulse_mode | Int | 0 | Pulse Mode Setting |
| wm_opt1 | Int | 0 | Welding Mode Option 1 Setting |
| simulation | int | 0 | Simulation Mode Setting |
| ts_opt1 | Int | 0 | Test Signal Option 1 Setting |
| ts_opt2 | Int | 0 | Test Signal Option 2 Setting |
| Job_num | Int | 0 | Job Number Setting |
| synergic_id | Int | 0 | Synergic ID Setting |
| r_wire_feed_speed | float | 0 | Wire Feeding Speed Setting |

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|----------------------------|
| voltage_correct | float | 0 | Voltage Correction Setting |
| dynamic_correct | float | 0 | Dynamic Correction Setting |
| r_opt1 | float | 0 | Option 1 Setting |
| r_opt2 | float | 0 | Option 2 Setting |
| r_opt3 | float | 0 | Option 3 Setting |
| r_opt4 | float | 0 | Option 4 Setting |
| r_opt5 | float | 0 | Option 5 Setting |
| r_opt6 | float | 0 | Option 6 Setting |
| r_opt7 | float | 0 | Option 7 Setting |
| r_opt8 | float | 0 | Option 8 Setting |
| r_opt9 | float | 0 | Option 9 Setting |
| r_opt10 | float | 0 | Option 10 Setting |
| r_opt11 | float | 0 | Option 11 Setting |
| r_opt12 | float | 0 | Option 12 Setting |
| r_opt13 | float | 0 | Option 13 Setting |
| r_opt14 | float | 0 | Option 14 Setting |
| r_opt15 | float | 0 | Option 15 Setting |

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 app_weld_enable_digital()
2 app_weld_set_weld_cond_digital(flag_dry_run = 0 , vel_target = 16 ,
3   vel_min = 0.00 , vel_max = 16.67 , welding_mode = 3 , s_2t = 0 ,
4   pulse_mode = 0 , wm_opt1 = 0 , simulation = 0 , ts_opt1 = 0 ,
5   ts_opt2 = 0 , job_num = 4 , svnergic_id = 0 , r_wire_feed_speed = 1
6   0 , voltage_correct = 0 , dynamic_correct = 0 , r_opt1 = 0 ,
7   r_opt2 = 0 , r_opt3 = 0 , r_opt4 = 0 , r_opt5 = 0 , r_opt6 = 0 ,
8   r_opt7 = 0 , r_opt8 = 0 , r_opt9 = 0 , r_opt10 = 0 , r_opt11 = 0 ,
9   r_opt12 = 0 , r_opt13 = 0 , r_opt14 = 0 , r_opt15 = 0 )
#Welding Speed=60 mm/sec (=1 cm/min), Welding Mode=3, Job
Number=4, Wire Feeding Speed=10 m/min
4
5 app_weld_disable_digital()
```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)\(p. 491\)](#)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\], ...\)\(p. 494\)](#)
- [app_weld_enable_digital\(\)\(p. 465\)](#)
- [app_weld_set_weld_cond_digital\(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...\)\(p. 498\)](#)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)\(p. 503\)](#)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)\(p. 503\)](#)
- [app_weld_disable_digital\(\)\(p. 467\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.12 app_weld_adj_welding_cond_digital(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None)

Features

This adjusts the welding condition and weaving condition during welding with a remote control welder. It is used to change the welding condition of each section in a series of sections before calling the motion command (movel(), movec(), moveb(), movesx()). If an adjustment factor is entered using this command, the corresponding welding condition and weaving condition are adjusted, and real-time adjustment of the welding/weaving condition from the welding monitoring information screen of the TP becomes unavailable. Execute flag_reset=1 to reset the adjusted condition to the main condition set using app_weld_set_weld_cond_digital() and app_weld_weave_cond_trapezoidal(). Setting flag_reset=1 will reset to the final condition adjusted in real-time using the TP (the weaving width ratio (wv_width_ratio), which cannot be adjusted in real-time, is changed to 1), and the welding condition can be adjusted in real-time from the TP.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| flag_reset | int | 0 | 0: Apply Adjustment 1 : Default Target (app_weld_set_weld_cond_digital()) Apply Value |
| f_target | float | - | Feeding Speed (m/min) |
| vel_target | float | - | Target Speed (mm/sec) <ul style="list-style-type: none"> • Take note that the unit is different from that of the teaching pendant input (cm/min) |
| wv_offset | float[2] | - | Weaving Coordinate-Y Direction Offset (mm) |
| | | - | Weaving Coordinate-Z Direction Offset (mm) |
| wv_width_ratio | float | - | Changed Weaving Width/Set Weaving Width Ratio (0-2) |
| dynamic_cor | float | - | Dynamic Correction |
| voltage_cor | float | - | Voltage Correction |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|-------------|
| job_number | float | - | Job Number |
| synergic_id | float | - | Synergic ID |

 **Note**

Conditions which do not designate a value in factors vel_target/wv_offset/wv_width_ratio/dynamic_cor/voltage_cor/job_number/synergic_id will maintain the current condition (including conditions with real-time adjustments), so only set factors that require adjustment. However, in the case of wv_offset, even if an adjustment is made only to the Y direction or Z direction, values for both sequences must be entered.

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ),v = 30 ,a = 60 )
2
3
4
5 pt1 = posx( 559 , 434.5 , 651.5 , 45 , 180 , 45 )
6
7 pt2 = posx( 559 , 434.5 , 151.5 , 45 , 180 , 45 )
8
9 pt3 = posx( 559 , 0.0 , 151.5 , 45 , 180 , 45 )
10
11
12 app_weld_enable_digital()
13
14
15
16
17 app_weld_set_weld_cond_digital(flag_dry_run = 0 , vel_target = 16 ,
18 vel_min = 0.00 , vel_max = 16.67 , welding_mode = 3 , s_2t = 0 ,
19 pulse_mode = 0 , wm_opt1 = 0 , simulation = 0 , ts_opt1 = 0 ,
20 ts_opt2 = 0 , job_num = 4 , synergic_id = 0 , r_wire_feed_speed = 1
21 5 , voltage_correct = 0 , dynamic_correct = 0 , r_opt1 = 0 ,
22 r_opt2 = 0 , r_opt3 = 0 , r_opt4 = 0 , r_opt5 = 0 , r_opt6 = 0 ,
23 r_opt7 = 0 , r_opt8 = 0 , r_opt9 = 0 , r_opt10 = 0 , r_opt11 = 0 ,
24 r_opt12 = 0 , r_opt13 = 0 , r_opt14 = 0 , r_opt15 = 0 )
25
26
27 movel(pt1, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
28
29 app_weld_adi_welding_cond_digital(flag_reset = 0 , f_target = 10 ,
30 vel_target = 16 , wv_offset = [ 20 , 30 ] , wv_width_ratio = 0.5 ,
31 dynamic_cor = 0 , voltage_cor = 0 , job_number = 5 , synergic_id = 4
32
33 movel(pt2, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
34
35 app_weld_adj_welding_cond_digital(flag_reset = 1 )
36
37 movel(pt3, v = 5 , a = 5 , app_type = DR_MV_APP_WELD)
# Start Point → pt1: Apply Initial Welding Condition Setting (Job
# Number: 4, Synergic ID: 0, Feeding Speed: 15 m/min)
# pt1 → pt2: Apply Correction Condition (Job Number: 5, Synergic
# ID: 4, Feeding Speed: 15 m/min)
# pt2 → pt3: Apply Initial Setting Apply Initial Welding
# Condition Setting (Job Number: 4, Synergic ID: 0, Feeding Speed:
# 15 m/min)

```

```

38
39
40
41    app_weld_disable_digital()

```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\], wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)
- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)\(p. 480\)](#)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], ...\)\(p. 483\)](#)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)\(p. 487\)](#)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0\], ...\)\(p. 491\)](#)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0\], ...\)\(p. 494\)](#)
- [app_weld_enable_digital\(\)\(p. 465\)](#)
- [app_weld_set_weld_cond_digital\(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0, ...\)\(p. 498\)](#)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)\(p. 503\)](#)
- [app_weld_disable_digital\(\)\(p. 467\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [moveL\(\)\(p. 59\)](#)
- [amoveL\(\)\(p. 98\)](#)
- [moveC\(\)\(p. 68\)](#)

- [amovec\(\)](#)(p. 104)
- [moveb\(\)](#)(p. 81)
- [amoveb\(\)](#)(p. 114)
- [movesx\(\)](#)(p. 77)
- [amovesx\(\)](#)(p. 111)

12.3.13 app_weld_get_welding_cond_digital()

Features

This monitors the welding status during welding with a remote control welder. Items available for monitoring are voltage correction/dynamic correction/feeding speed/speed/weaving offset/error number/error status/wire tip fusion/option and measured voltage/current/feeding speed and welding status. Signals other than the basic monitoring items can be added. The corresponding signal must be preset using the interface `app_weld_set_interface_eip_m2r_other()`. In addition, it is possible to check the fail status using the welding status factor.

Parameter

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

Return

| Value | Description |
|-------------|--|
| voltage_cor | Current Target Voltage Correction (V) (target with adjustment applied) |
| dynamic_cor | Current Target Dynamic Correction (target with adjustment applied) |
| f_target | Current Target Feeding Speed (m/min) (target with adjustment applied) |
| vel_target | Current Target Speed (mm/sec) (target with adjustment applied) <ul style="list-style-type: none"> Take note that the unit is different from that of the monitoring output unit (cm/min) of the teaching pendant |
| v_meas | Current Measured Voltage (V) |
| c_meas | Current Measured Current (A) |

| Value | Description |
|----------------|---|
| wv_offset[2] | Current Target Offset (Y and Z directions, mm) (target with adjustment applied) |
| status | Non-weld:0, Weld (Normal): 1, Weld (Abnormal): 9, Dry-run: 99 |
| f_meas | Current Measured Feeding Speed (mm/sec) |
| error_num | Error Number |
| wire_stick | Wire Tip Fusion Status (0: Normal, 1: Abnormal) |
| error | Error Status (0: Normal, 1: Abnormal) |
| option1 | Option 1 Information |
| option2 | Option 2 Information |
| option3 | Option 3 Information |
| option4 | Option 4 Information |
| option5 | Option 5 Information |
| option6 | Option 6 Information |
| option7 | Option 7 Information |
| option8 | Option 8 Information |
| option9 | Option 9 Information |
| option10 | Option 10 Information |
| current_flow | Current Flow Status (0: Normal, 1: Abnormal) |
| process_active | Process Activation Status (0: Normal, 1: Abnormal) |
| machine_ready | Welding Preparation Status (0: Normal, 1: Abnormal) |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ),v = 30 ,a = 60 )
2
3
4
5 pt1 = posx( 559 , 434.5 , 651.5 , 45 , 180 , 45 )
6
7 pt2 = posx( 559 , 434.5 , 151.5 , 45 , 180 , 45 )
8
9 pt3 = posx( 559 , 0.0 , 151.5 , 45 , 180 , 45 )
10
11
12 app_weld_enable_digital()
13
14
15
16 app_weld_set_weld_cond_digital(flag_dry_run = 1 , vel_target = 16 ,
17 vel_min = 0.00 , vel_max = 16.67 , welding_mode = 3 , s2t = 0 ,
pulse_mode = 0 , wm_opt1 = 0 , simulation = 0 , ts_opt1 = 0 ,
ts_opt2 = 0 , job_num = 4 , synergic_id = 0 , r_wire_feed_speed = 1
5 , voltage_correct = 0 , dynamic_correct = 0 , r_opt1 = 0 ,
r_opt2 = 0 , r_opt3 = 0 , r_opt4 = 0 , r_opt5 = 0 , r_opt6 = 0 ,
r_opt7 = 0 , r_opt8 = 0 , r_opt9 = 0 , r_opt10 = 0 , r_opt11 = 0 ,
r_opt12 = 0 , r_opt13 = 0 , r_opt14 = 0 , r_opt15 = 0 )
18
19
20
21 movel(pt1, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)

```

```

22     app_weld_adi_welding_cond_digital(flag_reset = 0 , f_target = 10 ,
23                                         vel_target = 16 , wv_offset = [ 20 , 30 ] , wv_width_ratio = 0.5 ,
24                                         dynamic_cor = 0 , voltage_cor = 0 , job_number = 5 , synergic_id = 4
25                                         movel(pt2, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
26                                         app_weld_adj_welding_cond_digital(flag_reset = 1 )
27                                         amovel(pt3, v = 5 , a = 5 , app_type = DR_MV_APP_WELD)
28                                         ...
29                                         ...
30                                         ...
31                                         ...
32                                         ...
33                                         ...
34                                         ...
35                                         [voltage_cor, dynamic_cor, f_target, vel_target, v_meas, c_meas,
36                                         wv_offset, status, f_meas, error_num, wire_stick,
37                                         error, opt1, opt2, opt3, opt4, opt5, opt6, opt7, opt8, opt9,
38                                         opt10, current_flow, process_active, machine_ready] = app_weld_get
39                                         _welding_cond_digital();
40                                         ...
41                                         ...
42                                         while True :
43                                         if status == 9 :
44                                         tp_popup( "welding error!! " , DR_PM_ALARM, 1 )
45                                         # An alarm is generated if abnormal welding occurs (status=9)
46                                         else :
47                                         if check_motion() == 0 :
48                                         break
49                                         ...
50                                         ...
51                                         ...
52                                         ...
53                                         ...
54                                         ...
55                                         ...
56                                         ...
57                                         app_weld_disable_digital()

```

Related commands

- [app_weld_set_interface_eip_r2m_process\(welding_start=\[0,0,0,0,0,0,0,0\], robot_ready=\[0,0,0,0,0,0,0,0\], error_reset=\[0,0,0,0,0,0,0,0\]\)\(p. 469\)](#)
- [app_weld_set_interface_eip_r2m_mode\(welding_mode=\[0,0,0,0,0,0,0,0\], s_2t=\[0,0,0,0,0,0,0,0\], pulse_mode=\[0,0,0,0,0,0,0,0\],wm_opt1=\[0,0,0,0,0,0,0,0\]\)\(p. 474\)](#)
- [app_weld_set_interface_eip_r2m_test\(gas_test=\[0,0,0,0,0,0,0,0\], inching_plus=\[0,0,0,0,0,0,0,0\], inching_minus=\[0,0,0,0,0,0,0,0\], blow_out_torch=\[0,0,0,0,0,0,0,0\], simulation=\[0,0,0,0,0,0,0,0\], ts_opt1=\[0,0,0,0,0,0,0,0\], ...\)\(p. 477\)](#)

- [app_weld_set_interface_eip_r2m_condition\(job_num=\[0,0,0,0,0,0,0,0\], synergic_id=\[0,0,0,0,0,0,0,0\], r_wire_feed_speed=\[0,0,0,0,0,0,0,0\], voltage_corret=\[0,0,0,0,0,0,0,0,0\], dynamic_correct=\[0,0,0,0,0,0,0,0\]\)](#)(p. 480)
- [app_weld_set_interface_eip_r2m_option\(opt1=\[0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0,...\]\)](#)(p. 483)
- [app_weld_set_interface_eip_m2r_process\(current_flow=\[0,0,0,0,0,0,0,0\], process_active=\[0,0,0,0,0,0,0,0,0\], main_current=\[0,0,0,0,0,0,0,0,0\], machine_ready=\[0,0,0,0,0,0,0,0,0\], comm_ready=\[0,0,0,0,0,0,0,0\]\)](#)(p. 487)
- [app_weld_set_interface_eip_m2r_monitoring\(welding_voltage=\[0,0,0,0,0,0,0,0\], welding_current=\[0,0,0,0,0,0,0,0\], wire_feed_speed=\[0,0,0,0,0,0,0,0\], wire_stick=\[0,0,0,0,0,0,0,0,0\], error=\[0,0,0,0,0,0,0,0,...\]\)](#)(p. 491)
- [app_weld_set_interface_eip_m2r_other\(opt1=\[0,0,0,0,0,0,0,0,0\], opt2=\[0,0,0,0,0,0,0,0,0\], opt3=\[0,0,0,0,0,0,0,0,0\], opt4=\[0,0,0,0,0,0,0,0,0\], opt5=\[0,0,0,0,0,0,0,0,0\], opt6=\[0,0,0,0,0,0,0,0,0\], opt7=\[0,0,0,0,0,0,0,0,...\]\)](#)(p. 494)
- [app_weld_enable_digital\(\)](#)(p. 465)
- [app_weld_set_weld_cond_digital\(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...\)](#)(p. 498)
- [app_weld_adj_welding_cond_digital\(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None\)](#)(p. 503)
- [app_weld_disable_digital\(\)](#)(p. 467)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)](#)(p. 530)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)](#)(p. 532)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)](#)(p. 535)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)](#)(p. 537)
- [movel\(\)](#)(p. 59)
- [amovel\(\)](#)(p. 98)
- [movec\(\)](#)(p. 68)
- [amovec\(\)](#)(p. 104)
- [moveb\(\)](#)(p. 81)
- [amoveb\(\)](#)(p. 114)
- [movesx\(\)](#)(p. 77)
- [amovesx\(\)](#)(p. 111)

12.3.14 app_weld_enable_analog(ch_v_out=[1,0], spec_v_out=[0,0,0,0], ch_f_out=[2,0], spec_f_out=[0,0,0,0], ch_v_in=[1,0], spec_v_in=[0,0,0,0], ch_c_in=[2,0], spec_c_in=[0,0,0,0], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4, ...)

Features

This enables the analog welding function. It enters the connection and environment information of the available welding machine's analog I/O and digital signal output as input factors.

The target welding machine must support an analog interface so it can receive target current and target voltage inputs from the analog output channel of the connected controller. Set the channel number (1 or 2) and output mode (current/voltage) of the physically connected analog channel in ch_v_out and ch_f_out. The analog input/output range of the controller is 0-10 V for voltage mode and 4-20 mA for current mode. Make sure to set the mode and output range of each channel to be compatible with the input specification and range of the welding machine. For example, if the target input range of the welding machine is 0-10 V, it is ideal to set the output channel of the controller as voltage mode (0-10 V output range). Another example would be if the input channel specification of the welding machine is 2-15 V. In this case, set the analog channel current mode (4-20 mA output range) on the corresponding controller and connect a 75 ohm resistance to output a 3-15 V voltage. (In this case, the 2-3 V range, which cannot be set using the controller, cannot set a target.) It is recommended that the welding machine be set with as large an input range as possible.

Set the maximum and minimum range of the controller analog output in spec_v_out and spec_f_out.

First item of spec_v_out/spec_f_out = WO_min

Second item of spec_v_out/spec_f_out = CO_min

Third item of spec_v_out/spec_f_out = WO_max

Fourth item of spec_v_out/spec_f_out = CO_max

Where, WO_min and WO_max are the minimum and maximum output of the welder, and CO_min and CO_max are the controller analog output corresponding to WO_min and WO_max.

Note

The welding current varies according to the wire feeding speed, basic material, material/type/stick-out of the welding wire, and welding voltage, and this must be monitored with the welding machine or a separate current sensor.

In order to monitor the voltage/current measurements during welding, it is necessary to connect an analog output welding machine or a separate sensor. Set the analog input channel number and input mode of the corresponding controller in ch_v_in and ch_c_in.

Set the maximum and minimum input range of sensor measured in spec_v_in and spec_f_in.

First item of spec_v_in/spec_c_in = SO_min
 Second item of spec_v_in/spec_c_in = CI_min
 Third item of spec_v_in/spec_c_in = SO_max
 Fourth item of spec_v_in/spec_c_in = CI_max

Where, SO_min and SO_max are the minimum and maximum sensor, and CI_min and CI_max are the controller input corresponding to SO_min and SO_max.

Set the channel numbers for ARC-ON/OFF (gas output signal - start/end), GAS-ON/OFF (gas output signal - start/end), INCHING-Forward-ON/OFF (forward wire feed signal - start/end), INCHING-Backward-ON/OFF (backward wire feed signal - start/end), and BlowOut-ON/OFF (torch cleaning gas output signal - start/end), which connect to the welding machine using digital contact method. In the case of signal outputs other than the ARC-ON/OFF signal, enter them selectively, depending on whether the welding machine supports the corresponding function.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|--|
| ch_v_out | list(int[2]) | 1 | Target Voltage Analog Output Channel (1-2) If no designation is made: 0 |
| | | 0 | 0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V) |
| spec_v_out | list(float[4]) | 0 | Welder Output Voltage (V) Minimum (a) |
| | | 0 | Controller Output corresponding to (a) |
| | | 0 | Welder Output Voltage (V) Maximum (b) |
| | | 0 | Controller Output corresponding to (b) |

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|---|
| ch_f_out | list(int[2]) | 2 | Feeding Speed Command Analog Output Channel (1-2) If no designation is made: 0 |
| | | 0 | 0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V) |
| spec_f_out | list(float[4]) | 0 | Feeding Speed (m/min) Minimum (c) |
| | | 0 | Controller Output corresponding to (c) |
| | | 0 | Feeding Speed (m/min) Maximum (d) |
| | | 0 | Controller Output corresponding to (d) |
| ch_v_in | list(int[2]) | 1 | Voltage Sensor Analog Input Channel (1-2) If no sensor is present: 0 |
| | | 0 | 0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V) |
| spec_v_in | list(float[4]) | 0 | Voltage Sensor Input (V) Minimum (e) |
| | | 0 | Controller Input corresponding to (e) |
| | | 0 | Voltage Sensor Input (V) Maximum (f) |
| | | 0 | Controller Input corresponding to (f) |
| ch_c_in | list(int[2]) | 2 | Current Sensor Analog Input Channel (1-2) If no sensor is present: 0 |
| | | 0 | 0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V) |
| spec_c_in | list(float[4]) | 0 | Current Sensor Input (A) Minimum (g) |
| | | 0 | Controller Input corresponding to (g) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| | | 0 | Current Sensor Input (A) Maximum (h) |
| | | 0 | Controller Input corresponding to (h) |
| ch_arc_on | int | 1 | Welding Output Digital Output Channel (1~16) |
| ch_gas_on | int | 2 | Protective Gas Digital Output Channel (1~16) If no connection is present: 0 |
| ch_inching_fwd | int | 3 | Welding Wire Forward Stick-Out Digital Output Channel (1~16) If no connection is present: 0 |
| ch_inching_bwd | int | 4 | Welding Wire Reverse Stick-Out Digital Output Channel (1~16) If no connection is present: 0 |
| ch_blow_out | int | 5 | Torch Cleaning Gas Output Digital Channel (1~16) If no connection is present: 0 |

Return

| Value | Description |
|----------------|------------------------|
| 0 | Enable Welding Success |
| Negative Value | Enable Welding Failure |

Exception

| Exception | Description |
|------------------------------|-------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |

| Exception | Description |
|--------------------------------|-----------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```
1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
3 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
4
5
6
7
8
9 # Voltage Output (Channel 1, Voltage Mode), Welder Voltage
Specification (Min/Max)=(0-300)
10
11 # Feeding Speed Output (Channel 2, Voltage Mode), Feeding Speed
Specification (Min/Max)=(0-40)
12
13 # Voltage Sensing (Channel 1, Voltage Mode), Sensor Measurement
Specification (Min/Max)=(0-300)
14
15 # Current Sensing (Channel 2, Voltage Mode), Sensor Specification
(Min/Max)=(0-40)
16
17 # Start Welding Signal (Channel 1), Gas Output Signal (Channel 2),
Wire Forward Stick-Out Signal (Channel 3),
18
19 # Wire Reverse Stick-Out Signal (Channel 4), Torch Cleaning Gas
Output Signal (Channel 5)
20
21 app_weld_disable_analog()
```

Related commands

- `app_weld_enable_analog(ch_v_out=[1,0], spec_v_out=[0,0,0,0], ch_f_out=[2,0], spec_f_out=[0,0,0,0], ch_v_in=[1,0], spec_v_in=[0,0,0,0], ch_c_in=[2,0], spec_c_in=[0,0,0,0], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4, ...)`(p. 512)
 - `app_weld_set_weld_cond_analog(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])`(p. 519)

- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [app_weld_adj_welding_cond_analog\(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None\)\(p. 523\)](#)
- [app_weld_get_welding_cond_analog\(\)\(p. 526\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.15 app_weld_disable_analog()

Features

This disables the analog welding function.

Parameter

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

Return

| Value | Description |
|----------------|-------------|
| 0 | Success |
| Negative Value | Failure |

Exception

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

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
3 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
4
5 app_weld_disable_analog()

```

Related commands

- [app_weld_enable_analog\(ch_v_out=\[1,0\],spec_v_out=\[0,0,0\],ch_f_out=\[2,0\],spec_f_out=\[0,0,0\],ch_v_in=\[1,0\],spec_v_in=\[0,0,0\],ch_c_in=\[2,0\],spec_c_in=\[0,0,0,0\],ch_arc_on=1,ch_gas_on=2,ch_inching_fwd=3,ch_inching_bwd=4,...\)\(p. 512\)](#)
- [app_weld_set_weld_cond_analog\(flag_dry_run=0,v_target=0,f_target=0,vel_target=0,vel_min=0,vel_max=0,weld_proc_param=\[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5\]\)\(p. 519\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [app_weld_adj_welding_cond_analog\(flag_reset=0,v_target=None,f_target=None,vel_target=None,wv_offset=None,wv_width_ratio=None\)\(p. 523\)](#)
- [app_weld_get_welding_cond_analog\(\)\(p. 526\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\],wv_ang=0,wv_param=\[3,0.6\]\)\(p. 537\)](#)

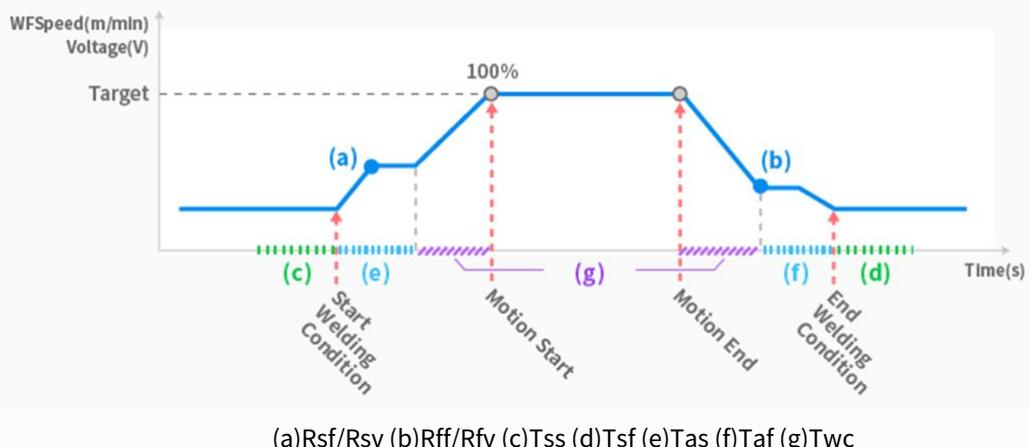
12.3.16 app_weld_set_weld_cond_analog(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])

Features

This sets the analog welding condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()) commands, and any operations starting at a point outside the welding section will generate an error. The Welding Parameter (weld_proc_param) of the welding condition displays detailed conditions, including gas during start/end welding and condition maintenance time. Refer to the figure below for the values to enter. Only one welding condition is allowed in a single welding section. This welding condition can be adjusted during welding with the app_weld_adj_welding_cond_analog() command while the voltage/feeding speed/speed (and weaving offset) can be also adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()).

Note

The welding current varies according to the wire feeding speed, basic material, material/type/stick-out of the welding wire, and welding voltage, and this must be monitored with the welding machine or a separate current sensor.



Parameter

| Parameter Name | Data Type | Default Value | Description |
|-----------------|----------------|---------------|--|
| flag_dry_run | int | 0 | Dry-Run Mode Actual Welding (0) Dry-Run (1): Motion/Weaving/Offset only |
| v_target | float | 0 | Target Voltage (V) |
| f_target | float | 0 | Target Feeding Speed (m/min) |
| vel_target | float | 0 | Target Speed (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| vel_min | float | 0 | Minimum Target Speed Correction (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| vel_max | float | 0 | Maximum Target Speed Correction (mm/sec) <ul style="list-style-type: none"> Take note that the unit is different from that of the teaching pendant input (cm/min) |
| weld_proc_param | list(float[9]) | 0.2 | Rsf (Feeding Speed Start Condition/Target Condition Ratio) ($0 < Rsf \leq 1$) |
| | | 0.2 | Rsv (Voltage Start Condition/Target Condition Ratio) ($0 < Rsv \leq 1$) |
| | | 0.5 | Tss (Protective Gas Discharge Time Before Welding, sec) ($0 \leq Tss$) |
| | | 0.5 | Tas (Start Welding Condition Maintenance Time, sec) ($0 \leq Tas$) |
| | | 0.5 | Twc (Change Welding Condition Time, sec) ($0 \leq Twc$) |

| Parameter Name | Data Type | Default Value | Description |
|-----------------------|------------------|----------------------|---|
| | | 0.2 | Rff (Feeding Speed End Condition/Target Condition Ratio) (0< Rff <= 1) |
| | | 0.2 | Rfv (Voltage End Condition/Target Condition Ratio) (0< Rfv <= 1) |
| | | 0.5 | Taf (End Welding Condition Maintenance Time, sec) (0<= Taf) |
| | | 0.5 | Tsf (Protective Gas Discharge Time After Welding, sec) (0<= Tsf) |

Return

| Value | Description |
|----------------|--------------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

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

Example

```

1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
3 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
4
5
6
7
8
9 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 24 ,
f_target = 20 , vel_target = 60 , vel_min = 10 ,
10 vel_max = 100 , weld_proc_param = [ 0.2 , 0.2 , 0.5 , 0.5 , 0.5 , 0
.2 , 0.2 , 0.5 , 0.5 ])
11
12 # Target Voltage/Feeding Speed = 24 V, 20 m/min, Welding Speed=60
13 mm/sec (=1 cm/min), Actual Welding, Use Default Welding Parameter
14
15 app_weld_disable_analog()

```

Related commands

- [app_weld_enable_analog\(ch_v_out=\[1,0\], spec_v_out=\[0,0,0,0\], ch_f_out=\[2,0\], spec_f_out=\[0,0,0,0\], ch_v_in=\[1,0\], spec_v_in=\[0,0,0,0\], ch_c_in=\[2,0\], spec_c_in=\[0,0,0,0\], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4, ...\)\(p. 512\)](#)
- [app_weld_set_weld_cond_analog\(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=\[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5\]\)\(p. 519\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [app_weld_adj_welding_cond_analog\(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None\)\(p. 523\)](#)
- [app_weld_get_welding_cond_analog\(\)\(p. 526\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.17 app_weld_adj_welding_cond_analog(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None)

Features

This adjusts the welding condition and weaving condition during analog welding. It is used to change the welding condition of each section in a series of sections before calling the motion command (movel(), movec(), moveb(), movesx()). If an adjustment factor is entered using this command, the corresponding welding condition and weaving condition are adjusted, and real-time adjustment of the welding/weaving condition from the welding monitoring information screen of the TP becomes unavailable. Execute flag_reset=1 to reset the adjusted condition to the main condition set using app_weld_set_weld_cond_analog() and app_weld_weave_cond_trapezoidal(). Setting flag_reset=1 will reset to the final condition adjusted in real-time using the TP (the weaving width ratio (wv_width_ratio), which cannot be adjusted in real-time, is changed to 1), and the welding condition can be adjusted in real-time from the TP.

Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| flag_reset | int | 0 | 0: Apply Adjustment 1: Default Target (app_weld_set_weld_cond_analog()) Apply Value |
| v_target | float | - | Target Voltage (V) |
| f_target | float | - | Feeding Speed (m/min)) |
| vel_target | float | - | Target Speed (mm/sec) <ul style="list-style-type: none"> • Take note that the unit is different from that of the teaching pendant input (cm/min) |
| wv_offset | float[2] | - | Weaving Coordinate-Y Direction Offset (mm) |
| | | - | Weaving Coordinate-Z Direction Offset (mm) |
| wv_width_ratio | float | - | Changed Weaving Width/Set Weaving Width Ratio (0-2) |

(i) Note

Conditions which do not designate a value in factors v_target/f_target/vel_target/wv_offset/wv_width_ratio will maintain the current condition (including conditions with real-time adjustments), so only set factors that require adjustment. However, in the case of wv_offset, even if an adjustment is made only to the Y direction or Z direction, values for both sequences must be entered.

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ),v = 30 ,a = 60 )
2
3
4
5 pt1 = posx( 559 , 434.5 , 651.5 , 45 , 180 , 45 )
6
7 pt2 = posx( 559 , 434.5 , 151.5 , 45 , 180 , 45 )
8
9 pt3 = posx( 559 , 0.0 , 151.5 , 45 , 180 , 45 )
10
11
12

```

```

13 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
14 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
15 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 ,
16 ch_inching_bwd = 4 , ch_blow_out = 5 )
17
18
19
20
21
22
23 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 24 ,
f_target = 20 , vel_target = 60 , vel_min = 10 ,
24 vel_max = 100 , weld_proc_param = [ 0.2 , 0.2 , 0.5 , 0.5 , 0.5 , 0
.2 , 0.2 , 0.5 , 0.5 ])
25
26
27
28
29 movel(pt1, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
30
31 app_weld_adi_welding_cond_analog(flag_reset = 0 , v_target = 20 ,
f_target = 10 , vel_target = 30 , wv_offset = [ 20 , 10 ],
wv_width_ratio = 0.5 )
32
33 movel(pt2, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
34
35 app_weld_adj_welding_cond_analog(flag_reset = 1 )
36
37 movel(pt3, v = 5 , a = 5 , app_type = DR_MV_APP_WELD)
38
39 # Start Point → pt1: Apply Origin Welding Condition (24V, 20 m/
min)
40
41 # pt1 → pt2: Apply Adjusted Condition (20 V, 10 m/min)
42
43 # pt2 → pt3: Apply Origin Condition (24 V, 20 m/min)
44
45
46
47 app_weld_disable_analog()

```

Related commands

- `app_weld_enable_analog(ch_v_out=[1,0], spec_v_out=[0,0,0,0], ch_f_out=[2,0], spec_f_out=[0,0,0,0], ch_v_in=[1,0], spec_v_in=[0,0,0,0], ch_c_in=[2,0], spec_c_in=[0,0,0,0], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4, ...)(p. 512)`
- `app_weld_set_weld_cond_analog(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])(p. 519)`

- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [app_weld_adj_welding_cond_analog\(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None\)\(p. 523\)](#)
- [app_weld_get_welding_cond_analog\(\)\(p. 526\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [movel\(\)\(p. 59\)](#)
- [amovel\(\)\(p. 98\)](#)
- [movec\(\)\(p. 68\)](#)
- [amovec\(\)\(p. 104\)](#)
- [moveb\(\)\(p. 81\)](#)
- [amoveb\(\)\(p. 114\)](#)
- [movesx\(\)\(p. 77\)](#)
- [amovesx\(\)\(p. 111\)](#)

12.3.18 app_weld_get_welding_cond_analog()

Features

This monitors the welding status during analog welding. Values available for monitoring are the current target voltage/current/speed/weaving offset/digital output signal and measured voltage/current and welding status. If measured voltage/current is not set (if ch_v_in and ch_c_in are not set during app_weld_enable()), the output of the corresponding values are set equal to the target voltage (v_target)/current (c_target). In addition, it is possible to check the fail status using the welding status factor.

Parameter

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

Return

| Value | Description |
|--------------|---|
| v_target | Current Target Voltage (V) (target with adjustment applied) |
| c_target | Current Target Current (A) (target with adjustment applied) |
| f_target | Current Target Feeding Speed (m/min) (target with adjustment applied) |
| vel_target | Current Target Speed (mm/sec) (target with adjustment applied) * Take note that the unit is different from that of the monitoring output unit (cm/min) of the teaching pendant |
| v_meas | Current Measured Voltage (V) |
| c_meas | Current Measured Current (A) |
| wv_offset[2] | Current Target Offset (Y and Z directions, mm) (target with adjustment applied) |
| sig_out[4] | Digital Output Signal (arc_on, gas_on, inching_fwd, inching_bwd) |
| status | Non-weld:0, Weld (Normal): 1, Weld (Abnormal): 9, Dry-run: 99 |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 movej(posj( 0 , 0 , 90 , 0 , 90 , 0 ),v = 30 ,a = 60 )
2
3
4
5 pt1 = posx( 559 , 434.5 , 651.5 , 45 , 180 , 45 )
6
7 pt2 = posx( 559 , 434.5 , 151.5 , 45 , 180 , 45 )
8
9 pt3 = posx( 559 , 0.0 , 151.5 , 45 , 180 , 45 )
10
11
12
13 app_weld_enable_analog(ch_v_out = [ 1 , 1 ] , spec_v_out = [ 0 , 0 ,
14 00 , 10 ] , ch_f_out = [ 2 , 1 ] ,
15 spec_f_out = [ 0 , 0 , 40 , 10 ] , ch_v_in = [ 1 , 1 ] , spec_v_in
16 = [ 0 , 0 , 300 , 10 ] , ch_c_in = [ 2 , 1 ] ,
17 spec_c_in = [ 0 , 0 , 40 , 10 ] , ch_arc_on = 1 , ch_gas_on = 2 ,
18 ch_inching_fwd = 3 ,
19 ch_inching_bwd = 4 , ch_blow_out = 5 )
20
21
22
23 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 24 ,
24 f_target = 20 , vel_target = 60 , vel_min = 10 ,
25 vel_max = 100 , weld_proc_param = [ 0.2 , 0.2 , 0.5 , 0.5 , 0.5 , 0
.2 , 0.2 , 0.5 , 0.5 ] )
26
27
28
29 movel(pt1, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
30
31 app_weld_adi_welding_cond_analog(flag_reset = 0 , v_target = 20 ,
32 f_target = 10 , vel_target = 30 , wv_offset = [ 20 , 10 ] ,
33 wv_width_ratio = 0.5 )
34
35 movel(pt2, v = 5 , a = 5 , r = 30 , app_type = DR_MV_APP_WELD)
36
37 app_weld_adj_welding_cond_analog(flag_reset = 1 )
38
39 amovel(pt3, v = 5 , a = 5 , app_type = DR_MV_APP_WELD)
40
41
42 while True :

```

```

43     Vt, Ct, Ft, velt, Vm, Cm, Off, Dout, status = app_weld_get_welding_cond_analog()
44
45     if status == 9 :
46
47         tp_popup( "welding error!! " , DR_PM_ALARM, 1 )
48
49 # An alarm is generated if abnormal welding occurs (status=9)
50
51     else :
52
53         if check_motion() == 0 :
54
55             break
56
57
58
59     app_weld_disable_analog()

```

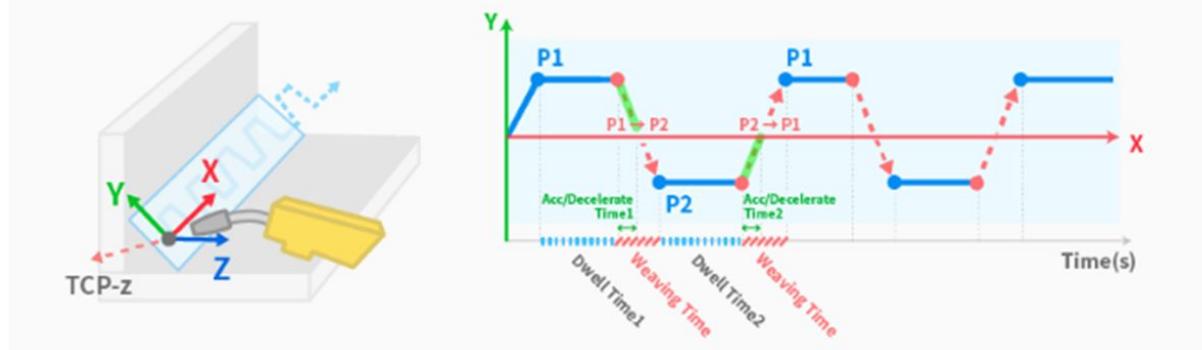
Related commands

- [app_weld_enable_analog\(ch_v_out=\[1,0\], spec_v_out=\[0,0,0,0\], ch_f_out=\[2,0\], spec_f_out=\[0,0,0,0\], ch_v_in=\[1,0\], spec_v_in=\[0,0,0,0\], ch_c_in=\[2,0\], spec_c_in=\[0,0,0,0\], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4, ...\)\(p. 512\)](#)
- [app_weld_set_weld_cond_analog\(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=\[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5\]\)\(p. 519\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)
- [app_weld_adj_welding_cond_analog\(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None\)\(p. 523\)](#)
- [app_weld_get_welding_cond_analog\(\)\(p. 526\)](#)
- [app_weld_weave_cond_trapezoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3\]\)\(p. 530\)](#)
- [app_weld_weave_cond_zigzag\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 532\)](#)
- [app_weld_weave_cond_circular\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,3,0.3,0.3\]\)\(p. 535\)](#)
- [app_weld_weave_cond_sinusoidal\(wv_offset=\[0,0\], wv_ang=0, wv_param=\[3,0.6\]\)\(p. 537\)](#)

12.3.19 app_weld_weave_cond_trapezoidal(wv_offset=[0,0], wv_ang=0, wv_param=[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3])

Features

This sets the trapezoidal weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/app_weld_set_weld_cond_digital()).



Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|---|
| wv_offset | float[2] | 0 | Weaving Coordinate-Y Direction Offset (mm) |
| | | 0 | Weaving Coordinate-Z Direction Offset (mm) |
| wv_angle | float | 0 | Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------------|---------------|--|
| wv_param | list(float[10]) | 0 | Weaving Point 1-x (mm) |
| | | 1.5 | Weaving Point 1-y (mm) |
| | | 0 | Weaving Point 2-x (mm) |
| | | -1.5 | Weaving Point 2-y (mm) |
| | | 0.3 | Weaving Point 1 → 2 hrs (sec) |
| | | 0.1 | Weaving Point 1 → 2 Dec/Acc Time (sec) |
| | | 0.3 | Weaving Point 1 Dwell Time (sec) |
| | | 0.3 | Weaving Point 2 → 1 hr (sec) |
| | | 0.1 | Weaving Point 2 → 1 Dec/Acc Time (sec) |
| | | 0.3 | Weaving Point 2 Dwell Time (sec) |

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

| Exception | Description |
|------------------------------|-------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |

| Exception | Description |
|--------------------------------|-----------------------------|
| DR_Error (DR_ERROR_RUNTIME) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

```

1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
3 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
4
5 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 200 ,
f_target = 150 , vel_target = 10 , vel_min = 10 ,
6 vel_max = 100 , weld_proc_param = [ 0.5 , 0.3 , 2 , 1 , 0.7 , 0.4 ,
.7 , 0.6 , 1.5 ])
7
8 app_weld_weave_cond_trapezoidal(wv_offset = [ 0 , 0 ], wv_ang = 0 ,
wv_param = [ 0 , 5 , 0 , -5 , 0.7 , 0.2 , 0.5 , 0.7 , 0.2 , 0.5 ]
)
9
10 # Trapezoidal Weaving Pattern, Offset=0.0, Tilt Angle=0, Weaving
Point 1=(0.5), Weaving Point 2=(0,-5), Weaving Time=0.7 (sec)
(same in both directions), Weaving Dec/Acc Time=0.2 (sec) (same in
both directions), Weaving Point 1 Dwell Time=0.5 sec, Weaving
Point 2 Dwell Time=0.5 sec
11 app_weld_disable_analog ()

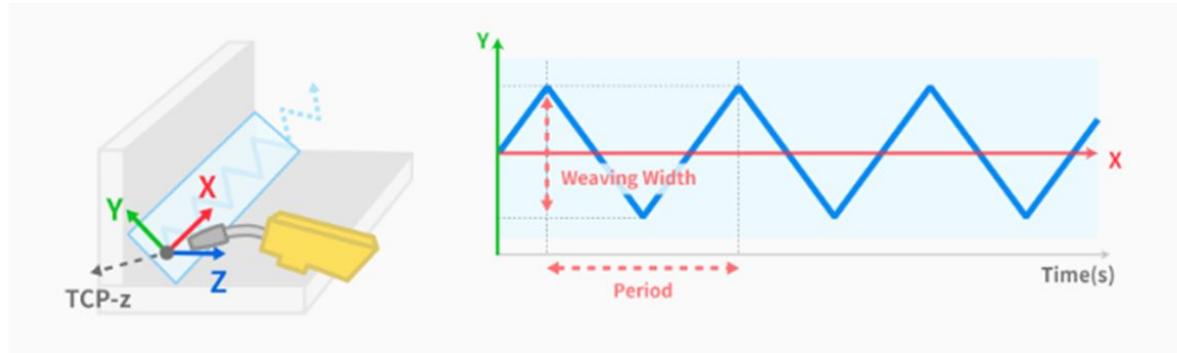
```

12.3.20 app_weld_weave_cond_zigzag(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6])

Features

This sets the zigzag weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The

offset or weaving width can be adjusted during welding with the `app_weld_adj_welding_cond_analog()`/
`app_weld_set_weld_cond_digital()` command or the voltage/current/speed and the offset can be adjusted from
the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant,
welding condition adjustments are only available if the welding condition adjustment status is set to RESET
using a command (the welding condition setting is designated with `app_weld_set_weld_cond_analog()`/
`app_weld_set_weld_cond_digital()`).



Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| wv_offset | float[2] | 0 | Weaving Coordinate-Y Direction Offset (mm) |
| | | 0 | Weaving Coordinate-Z Direction Offset (mm) |
| wv_angle | float | 0 | Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg) |
| wv_param | list(float[2]) | 3 | Weaving Width (mm) |
| | | 0.6 | Weaving Period (sec) |

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

| Exception | Description |
|---------------------------------|-----------------------------|
| DR_Error (DR_ERROR_TYPE) | Parameter data error |
| DR_Error (DR_ERROR_VALUE) | Invalid parameter value |
| DR_Error (DR_ERROR_RUNTIM E) | C Extension module error |
| DR_Error (DR_ERROR_STOP) | Program terminated forcibly |

Example

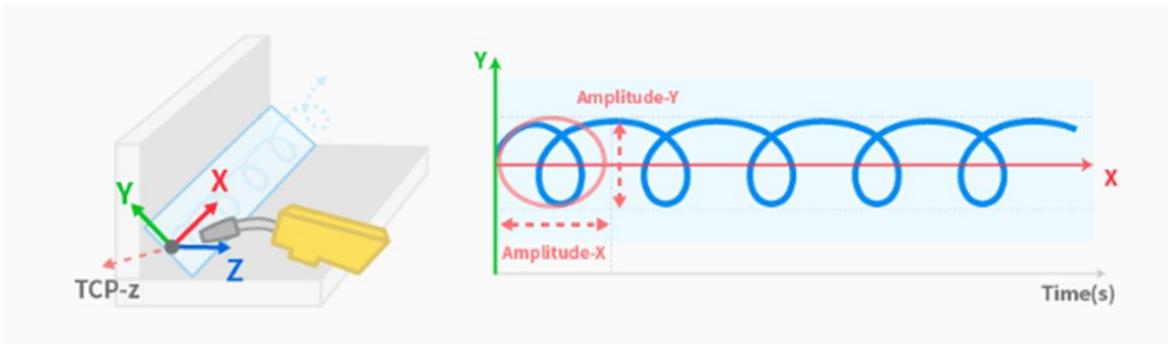
```

1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
3 = [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
4 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
5
6
7
8
9 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 200 ,
f_target = 150 , vel_target = 10 , vel_min = 10 ,
10 vel_max = 100 , weld_proc_param = [ 0.5 , 0.3 , 2 , 1 , 0.7 , 0.4 ,
.7 , 0.6 , 1.5 ])
11
12
13
14
15 app_weld_weave_cond_zigzag(wv_offset = [ 0 , 0 ], wv_ang = 0 ,
wv_param = [ 10 , 0.5 ])
16
17 # Zigzag Weaving Pattern, Offset=0.0 Tilt Angle=0, Weaving
18 Width=10 (mm), Weaving Period=0.5 (sec)
19 app_weld_disable_analog()
```

12.3.21 app_weld_weave_cond_circular(wv_offset=[0,0], wv_ang=0, wv_param=[3,3,0.3,0.3])

Features

This sets the circular weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog())/ app_weld_disable_digital() commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| wv_offset | float[2] | 0 | Weaving Coordinate-Y Direction Offset (mm) |
| | | 0 | Weaving Coordinate-Z Direction Offset (mm) |
| wv_angle | float | 0 | Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg) |
| wv_param | list(float[4]) | 3 | X Direction Weaving Width (mm) |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|----------------------------------|
| | | 3 | Y Direction Weaving Width (mm) |
| | | 0.3 | X Direction Weaving Period (sec) |
| | | 0.3 | Y Direction Weaving Period (sec) |

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

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

Example

```

1 app_weld enable analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ], spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ], spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )

```

```

4   app_weld.set_weld_cond_analog(flag_dry_run = 1 , v_target = 200 ,
5     f_target = 150 , vel_target = 10 , vel_min = 10 ,
6     vel_max = 100 , weld_proc_param = [ 0.5 , 0.3 , 2 , 1 , 0.7 , 0.4 ,
7     .7 , 0.6 , 1.5 ])
8   app_weld.weave_cond_circular(wv_offset = [ 0 , 0 ] , wv_ang = 0 ,
9     wv_param = [ 3 , 3 , 0.3 , 0.3 ])
# Circular Weaving Pattern, Offset=0,0 Tilt Angle=0, X Direction
Weaving Width=3 (mm), Y Direction Weaving=3 (mm), X Direction
Weaving Period=0.3 (s), Y Direction Weaving Period=0.3 (s)
10  app_weld_disable_analog()

```

12.3.22 app_weld_weave_cond_sinusoidal(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6])

Features

This sets the sinusoidal weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog() / app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog() / app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog() / app_weld_set_weld_cond_digital()).



Parameter

| Parameter Name | Data Type | Default Value | Description |
|----------------|----------------|---------------|---|
| wv_offset | float[2] | 0 | Weaving Coordinate-Y Direction Offset (mm) |
| | | 0 | Weaving Coordinate-Z Direction Offset (mm) |
| wv_angle | float | 0 | Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg) |
| wv_param | list(float[2]) | 3 | Weaving Width (mm) |
| | | 0.6 | Weaving Period (sec) |

Return

| Value | Description |
|----------------|-----------------|
| 0 | Setting Success |
| Negative value | Setting Failure |

Exception

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

Example

```
1 app_weld_enable_analog(ch_v_out = [ 1 , 1 ], spec_v_out = [ 0 , 0 ,
00 , 10 ], ch_f_out = [ 2 , 1 ],
2 spec_f_out = [ 0 , 0 , 40 , 10 ], ch_v_in = [ 1 , 1 ], spec_v_in
= [ 0 , 0 , 300 , 10 ], ch_c_in = [ 2 , 1 ],
3 spec_c_in = [ 0 , 0 , 40 , 10 ], ch_arc_on = 1 , ch_gas_on = 2 ,
ch_inching_fwd = 3 , ch_inching_bwd = 4 , ch_blow_out = 5 )
4
5
6
7
8
9 app_weld_set_weld_cond_analog(flag_dry_run = 1 , v_target = 200 ,
f_target = 150 , vel_target = 10 , vel_min = 10 ,
10 vel_max = 100 , weld_proc_param = [ 0.5 , 0.3 , 2 , 1 , 0.7 , 0.4 ,
.7 , 0.6 , 1.5 ])
11
12
13
14
15 app_weld_weave_cond_sinusoidal(wv_offset = [ 0 , 0 ] , wv_ang = 0 ,
wv_param = [ 10 , 0.5 ])
16
17 # Sinusoidal Weaving Pattern, Offset=0,0 Tilt Angle=0, Weaving
Width=10 (mm), Weaving Period=0.5 (s)
18
19 app_weld_disable_analog()
```

13 A-Series Command

13.1 Controller

13.1.1 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 |

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

Example

```

1  in1 = get_function_input( 1 )           # Reads the no. 1
function button input
2  in8 = get_function_input( 4 )           # Reads the no. 4
function button input

```

13.2 Flange I/O

13.2.1 flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)

Features

A command used for opening the Pseudo Flange Serial communication port.

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 2400, 4800, 9600, 19200, 38400, 57600, 115200 etc |
| bytesize | int | 8 | Number of data bits <ul style="list-style-type: none"> • DR_FIVEBITS : 5 • DR_SIXBITS : 6 • DR_SEVENBITS : 7 • DR_EIGHTBITS : 8 |

| Parameter Name | Data Type | Default Value | Description |
|----------------|-----------|---------------|--|
| parity | str | "N" | Parity checking <ul style="list-style-type: none"> • DR_PARITY_NONE: "N" • DR_PARITY_EVEN: "E" • DR_PARITY_ODD: "O" |
| stopbits | int | 1 | Number of stop bits <ul style="list-style-type: none"> • DR_STOPBITS_ONE =1 • DR_STOPBITS_TWO =2 |

Return

| Value | Description |
|----------------|-----------------------|
| 0 | Successful connection |
| Negative value | Fail to connection |

Exception

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

13.2.2 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 |

13.2.3 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"> • The data type must be a byte. • Refer to the example below. |

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_RUNTIM E) | C extension module error occurred |
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

13.2.4 flange_serial_read(timeout=None)

Features

This function reads the data from a flange serial port.

Return

| Value | 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_RUNTIM E) | C extension module error occurred |

| Exception | Description |
|-----------------------------|-------------------------------|
| DR_Error (DR_ERROR_STOP) | Program terminated forcefully |

13.2.5 Integrated example - A-Series Command

This example controls robotiq 2f using Pseudo Flange Serial.

Example : robotiq 2f

```
35 flange_serial_write(modbus_send_make(b
36 "\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\xFF\xFF\xFF" ))
37 res, data = flange_serial_read()
38
39 #Step 5: Read Gripper status until the grasp is completed
40 flange_serial_write(modbus_send_make(b "\x09\x03\x07\xD0\x00\x03"))
41 res, data = flange_serial_read()
42
43 #Step 6: Move the robot to the release location
44 wait( 1 )
45
46 #Step 7: Open the Gripper at full speed and full force
47 flange_serial_write(modbus_send_make(b
48 "\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\x00\xFF\xFF" ))
49 res, data = flange_serial_read()
50
51 #Step 8: Read Gripper status until the opening is completed
52 flange_serial_write(modbus_send_make(b "\x09\x03\x07\xD0\x00\x03"))
53 res, data = flange_serial_read()
54
55 flange_serial_close()
```