UNIVERSAL ROBOTS

C-API

Functional Specification

Motion Control Commands

1) void robotinterface_command_position_velocity_acceleration

(
const double *q,
const double *qd,
const double *qdd
)

Purpose : *Send a set of joint angle values(q) to the controller along with speed(qd) and acceleration(qdd) values on how to get to said configuration.*

Parameters:

q - list of 6 joint angle values [j0, j1, j2, j3, j4, j5] in radians
qd – list of 6 joint speed values in radians/sec
qdd – list of 6 joint accel values in radians/sec/sec

Return:

No return value

2) void robotinterface_command_velocity_acceleration

(
const double *q,
const double *qd,
const double *qdd
)

Purpose : *Send a set of joint speeds(qd) and accels(qdd) to the controller to have the joints accelerate to and achieve said speed.*

Parameters:

q - returns list of 6 joint angle values [j0, j1, j2, j3, j4, j5] in radians, configuration
qd – list of 6 joint speed values in radians/sec
qdd – list of 6 joint accel values in radians/sec/sec

Return:

No return value

3) void robotinterface_command_velocity

(

const double *qd,

)

Purpose : *Send a set of joint speeds(qd) to the controller to have the joints move with set speeds. Sent every 8ms to maintain set speed.*

Parameters:

qd – list of 6 joint speed values [j0, j1, j2, j3, j4, j5] in radians/sec

Return:

No return value

4) void robotinterface_command_joint_position_velocity_acceleration(

int joint,
double q,
double qd,
double qdd

)

Purpose : *Send joint ID(joint), position(q), speed(qd) and accel(qdd)to the controller to have the specified joint move with set speed and accel.*

Parameters:

joint – joint ID [0-5 => base to wrist3]
q - joint angle value in radians
qd –joint speed value in radians/sec
qdd –joint accel value in radians/sec/sec

Return:

No return value

5) void robotinterface_command_velocity_security_torque_control_torque
(
const double *qd,
const double *security_torque,
const double *control_torque,
const double *softness
)

Purpose : *Send set(6 values for each joint) of speeds(qd), security torques, control torques and softness factors to the controller to have all the joints move with set speed and torque values.*

Parameters:

qd –joint speed values in radians/sec [j0, j1, j2, j3, j4, j5]
security torque – reference torques for safety limits
control_torque – target torques
softness – [0 to 1] gain values (stiff to soft)

Return:

No return value

I/O interface commands

6) void robotinterface_command_digital_out_port(int port, int value)

Purpose : *Set any digital output between 0 and 9 to high or low*

Parameters:

port – 0 - 9

value – 0(low) or 1(high)

Return:

No return value

7) void robotinterface_command_digital_out_bits(unsigned short bits)

Purpose : *Set multiple digital outputs together using bit masks*

Parameters:

bits – value between 0 and 1023

e.g 1 => digital_output[0] is high.

1023 => outputs 0 through 9 are high

Return:

No return value

8) void robotinterface_command_analog_output_domain (int port, int type)

Purpose : *Set output for any of the 2 analog outs to current or voltage output*

Parameters:

port – 0 or 1

type – 0 => current,

1=> voltage

Return:

No return value

9) void robotinterface_command_analog_out_port(int port, double value)

Purpose : *Set output for any of the 2 analog outs to a value*

Parameters:

port – 0 or 1

value – current (4-20mA) or voltage(0-10v)

Return:

No return value

10) void robotinterface_command_analog_input_range_port(int port, int range);

Purpose : *Set analog input and input range for selected input*

Parameters:

port – 0 - 3

range– 0 => 0 to 5v

1=> 0 to 10v

2=> -5 to 5v

3=> -10 to 10v

Return:

No return value

11) void robotinterface_command_tool_output_voltage(unsigned char value)

Purpose : *Set output voltage at tool connector to 0,12 or 24v*

Parameters:

value – 0, 12 or 24v

Return:

No return value

Robot state commands

12) **void robotinterface_set_tcp**(const double *tcp_pose)

Purpose : *Set TCP pose*

Parameters:

tcp_pose – list of 6 values => [x, y, z, Rx, Ry, Rz] w.r.t base

Return:

No return value

13) **void robotinterface_set_tcp_payload_cog**(const double *tcp_payload_cog)

Purpose : *Set center of gravity of payload*

Parameters:

tcp_payload_cog – list of 3 values => [x, y, z] w.r.t tool frame

Return:

No return value

14) **void robotinterface_set_tcp_payload**(double tcp_payload)

Purpose : *Set actual payload at load end of robot*

Parameters:

tcp_payload – payload in kilograms

Return:

No return value

15) **void robotinterface_get_tcp** (double *tcp_pose)

Purpose : *Retrieve current TCP pose*

Parameters:

tcp_pose – returns TCP pose as a list of 6 values

Return:

No return value

16) void robotinterface_get_tcp_payload_cog(double *tcp_pose)

Purpose : *Retrieve current settings for center of gravity co-ordinates*

Parameters:

tcp_pose – returns a list of 3 points [x, y, z]

Return:

No return value

17) double robotinterface_get_tcp_payload()

Purpose : *Retrieve current settings for robot payload*

Parameters:

No parameters to be passed

Return:

Current payload settings in kilograms

18) void robotinterface_set_tcp_wrench(

const double *new_tcp_wrench
const int in_base_coord
)

Purpose : *set required TCP wrench settings*

Parameters:

new_tcp_wrench –list of 6 points force along x, y, z and twist about x,y and z

in_base_coord – specifies reference frame as base

Return:

No return value

19) void robotinterface_get_tcp_wrench(
double *gotten_tcp_wrench
)

Purpose : *get current TCP wrench settings*
Parameters:
gotten_tcp_wrench – returns list of 6 points, force along x, y, z and twist about x,y and z
Return:
No return value

20) void robotinterface_get_actual_position(
double *q
)

Purpose : *get current joint angles*
Parameters:
q – returns list of 6 joint angles
Return:
No return value

21) void robotinterface_get_actual_velocity(
double *qd
)

Purpose : *get current joint speeds*
Parameters:
qd – returns list of 6 joint speeds
Return:
No return value

22) void robotinterface_get_actual_current(
double *I
)

Purpose : *get joint current drawn*
Parameters:
I – returns list of 6 joint current
Return:
No return value

23) void robotinterface_get_tool_accelerometer_readings(
double *ax,
double *ay,
double *az,
)

Purpose : *get TCP acceleration along x y and z axes*
Parameters:
ax – acceleration along x axes
ay – acceleration along y axes
az – acceleration along z axes

Return:
No return value

Interface Handshake Commands

24) **int robotinterface_open**(
int open_simulated
)

Purpose : *open C-API interface*
Parameters:
open_simulated = communicate with real(0) or simulated robot

Return:
Result = 1(if simulation) else Ethernet connection message

25) **void robotinterface_send**()

Purpose : *after every control command*
Parameters:
n/a
Return:
n/a

26) **void robotinterface_security_stop**(
char joint_code
int error_state
int error_argument
)

Purpose : *to trigger a security stop*
Parameters:
joint_code – [0 to 6] base to tool
error_state – definition of error (string)
error_argument – [-1 to 10] set robotmode

Return:
n/a

RobotModes

-1	**NO_CONTROLLER_MODE**
0	**ROBOT_RUNNING_MODE (normal state)**
1	**ROBOT_FREEDRIVE_MODE**
2	**ROBOT_READY_MODE**
3	**ROBOT_INITIALIZING_MODE**
4	**ROBOT_SECURITY_STOPPED_MODE**
5	**ROBOT_EMERGENCY_STOPPED_MODE**
6	**ROBOT_FAULT_MODE**
7	**ROBOT_NO_POWER_MODE**
8	**ROBOT_NOT_CONNECTED_MODE**
9	**ROBOT_SHUTDOWN_MODE**
10	**ROBOT_SAFEGUARD_STOP_MODE**

Miscellaneous

```
uint8_t robotinterface_get_joint_mode(int joint);                pass joint ID to get joint mode as return value
void robotinterface_get_motor_temperature(float *T);             returns a list of 6 motor temperatures
float robotinterface_get_master_temperature();                   returns temperature of MASTERBOARD
void robotinterface_get_actual(double *q, double *qd);           returns list of current joint position and joint speed
void robotinterface_get_actual_current(double *I);               returns list of current joint currents
uint64_t robotinterface_get_step();              returns step size for motor in radians (same for all motors)
double robotinterface_get_time();                returns current timestamp
double robotinterface_get_time_per_step();       returns time taken for each step (changes with speed)
uint8_t robotinterface_get_robot_mode();         returns current robot mode [-1 to 10 in previous page]
uint8_t robotinterface_get_tool_mode();          returns current tool state
int robotinterface_is_power_on_robot();          returns 1 if power is on
int robotinterface_is_security_stopped();        returns 1 if robot is security stopped
int robotinterface_is_emergency_stopped();       returns 1 if robot is e-stopped
int robotinterface_is_power_button_pressed(); returns 1 if power button is pressed on pendant
void robotinterface_get_tcp_force(double *F); returns current force along x,y,z and twist about x,y,z (list of 6 values)
void robotinterface_get_tcp_speed(double *V); returns lnear and angular velocities at TCP (list of 6 values)
float robotinterface_get_robot_voltage_48V(); returns voltage on the powerbus of the robot
float robotinterface_get_robot_current();        returns current being drawn by the robot
float robotinterface_get_master_io_current();  returns current drawn by the I/Os on the MASTERBOARD
unsigned char robotinterface_get_master_safety_state(); returns state of the safety inputs on the MASTERBOARD
```

unsigned char robotinterface_get_master_on_off_state();	returns the ON/OFF state off the MASTERBOARD
void robotinterface_get_micro_temperature(float *T);	returns processor temperature on all joints (list of 6)
void robotinterface_get_joint_voltage(float *V);	returns joint voltages (list of 6)
float robotinterface_get_tool_temperature();	returns temperature of tool PCB
float robotinterface_get_tool_voltage_48V();	returns voltage across tool PCB
unsigned char robotinterface_get_tool_output_voltage();	returns voltage at tool output
float robotinterface_get_tool_current();	returns current drawn at tool