XArm7 Controlling API Instructions (C++)

Concept Explanations: (confirm before reading)

u8: 8-bit unsigned integer.
u16: 16-bit unsigned integer.
fp32: 32-bit floating point.
TCP: Tool Center Point.
[Reset System]: after using this instruction, command buffer will be cleared and system will go to STOPPED state.

Part I System States:

I-1. motion_en(u8 id, u8 value): *[Reset System]*
parameters (data type):
id (u8):
1-7 for target joint number,
or 8 for all joints.
value (u8):
0 – disable servo motor,
1 – enable servo motor.
function:
enable or disable the servo motor associated with the specified joint.
return value:
(int): execution state. (listed in Appendix I)

I-2. set_state(u8 value):
parameters (data type):
value (u8):
0 – ready for new motion command,
3 – pause on the current motion,
4 – terminate and not responsive for any motion command. *[Reset System]*
function:
setting the state of robot arm.
return value:
(int): execution state. (listed in Appendix I)

I-3. get_state(u8 *rx_data):
parameters (data type):
rx_data (u8*): [output] pointer to the space for result, dimension 1.
result: system state (u8)
1 – RUNNING, executing motion command.
2 – SLEEPING, not in execution, but ready to move.
3 – PAUSED, paused in the middle of unfinished motion.
4 – STOPPED, not ready for any motion commands.
function:

get the current state of robot arm.

return value:

(int): execution state. (listed in Appendix I)

Remarks:

if in STOPPED state, the robot can receive command but will not execute, the reason could be system environment not ready or errors encountered. Users have to set the state back to 0 (ready) for execution.

I-4. get_cmdnum(u16 *rx_data):

parameters (data type):

rx_data (u16*): [output] pointer to the space for result, dimension 1.

result: number of instructions. (u16)

function:

get the number of instructions stored in buffer.

return value:

(int): execution state.

I-5. get_err_code(u8 *rx_data):

parameters (data type):

rx_data (u8*): [output] pointer to the space for result, dimension 2.

rx_data[0]: error code. (u8)

rx_data[1]: warning code. (u8)

function:

get the error code as well as warning code.

return value:

(int): execution state.

Remarks:

meaning of the error and warning code is listed in Appendix II .

I-6. clean_err(): *[Reset System]*

parameters (data type):

None

function:

clear the error code.

return value:

(int): execution state.

I-7. clean_war():

parameters (data type):

None

function:

clear the warning code.

return value:

(int): execution state.

I-8. set_brake(u8 axis, u8 en): *[Reset System]*

parameters (data type):

axis (u8):

1-7 for target joint number,

or 8 for all joints.

en (u8):

0 – disable the joint brake,

1 – enable the joint brake.

function:

enable(lock) or disable(unlock) the motor brake associated with the specified joint.

return value:

(int): execution state.

Remarks:

use this instruction with absolute **CAUTION**, unlocking some joints will cause the robot part falling down!

Part II Basic Motions:

Reminder: If any error encountered, system will go to STOPPED state.

II-1. move_line(fp32 mvpose[6], fp32 mvvelo, fp32 mvacc, fp32 mvtime):

parameters (data type):

mvpose (fp32*): [X Y Z A B C] dimension 6.

X Y Z – Cartesian coordinate of TCP target position, unit is mm.

A B C – Cartesian coordinate of TCP target orientation, unit is radian.

Expressed in X-Y-Z fixed frame rotation.

mvvelo (fp32):

motion execution speed, unit is mm/s or rad/s.

mvacc (fp32):

motion execution acceleration, unit is mm/s^2 or rad/s^2.

mvtime (fp32):

execution time. [currently not meaningful, just leave it to 0]

function:

move TCP to target coordinate in a straight line trajectory.

return value:

(int): execution state.

II-2. move_lineb(fp32 mvpose[6], fp32 mvvelo, fp32 mvacc, fp32 mvtime, fp32 mvradii):

parameters (data type):

mvpose (fp32*): [X Y Z A B C] dimension 6.

X Y Z – Cartesian coordinate of TCP target position, unit is mm.

A B C – Cartesian coordinate of TCP target orientation, unit is radian.

Expressed in X-Y-Z fixed frame rotation.

mvvelo (fp32):

motion execution speed, unit is mm/s or rad/s.

mvacc (fp32):

motion execution acceleration, unit is mm/s^2 or rad/s^2.

mvtime (fp32):

execution time. [currently not meaningful, just leave it to 0]

mvradii (fp32):

blending (between two lines) radius length, unit is mm.

function:

move TCP to target coordinate in a straight line and ensure a smooth transition with the next motion command, by parabolic blending.

return value:

(int): execution state.

II-3. move_joint(fp32 mvjoint[7], fp32 mvvelo, fp32 mvacc, fp32 mvtime):

parameters (data type):

mvpose (fp32*): dimension 7.

7 target joint positions, expressed in radian.

mvvelo (fp32):

motion execution speed, unit is rad/s.

mvacc (fp32):

motion execution acceleration, unit is rad/s^2.

mvtime (fp32):

execution time. [currently not meaningful, just leave it to 0]

function:

move to a target joint space position. Not ensuring any trajectory type.

return value:

(int): execution state.

II-4. move_gohome(fp32 mvvelo, fp32 mvacc, fp32 mvtime):

parameters (data type):

mvvelo (fp32):

motion execution speed, unit is rad/s.

mvacc (fp32):

motion execution acceleration, unit is rad/s^2.

mvtime (fp32):

execution time. [currently not meaningful, just leave it to 0]

function:

move to home position in joint space. Not ensuring any trajectory type.

return value:

(int): execution state.

II-5. sleep_instruction(fp32 sltime):

parameters (data type):

sltime (fp32): time duration for holding, unit is second.

function:

sleep / hold on for some time before moving on to next instruction.

return value:

(int): execution state.

Part III Motion Parameter Configurations:

Reminder: the following parameter configurations can not be done when system is in STOPPED state!

III-1. set_tcp_jerk(fp32 jerk):

parameters (data type):

jerk (fp32):

TCP motion jerk, unit is mm/s^3.

function:

setting the TCP Cartesian jerk (derivative of acceleration).

return value:

(int): execution state.

III-2. set_tcp_maxacc(fp32 maxacc):

parameters (data type):

maxacc (fp32):

TCP motion acceleration, unit is mm/s^2.

function:

setting the TCP Cartesian acceleration.

return value:

(int): execution state.

III-3. set_joint_jerk(fp32 jerk):

parameters (data type):

jerk (fp32):

joint space motion jerk, unit is rad/s^3.

function:

setting the joint motion jerk (derivative of acceleration).

return value:

(int): execution state.

III-4. set_joint_maxacc(fp32 maxacc):

parameters (data type):

maxacc (fp32):

joint space motion acceleration, unit is rad/s^2.

function:

setting the joint motion acceleration.

return value:

(int): execution state.

III-5. set_tcp_offset(fp32 pose_offset[6]):

parameters (data type):

pose_offset (fp32*): [X Y Z A B C] dimension 6.

X Y Z – Cartesian coordinate of TCP offset position, unit is mm.

A B C – Cartesian coordinate of TCP offset orientation, unit is radian.

Expressed in X-Y-Z fixed frame rotation.

function:

setting Tool Center Point offset with respect to the center of robot end flange.

return value:

(int): execution state.

III-6. clean_conf():

parameters (data type):

None.

function:

delete the ini configuration file.

return value:

(int): execution state.

III-7. save_conf():

parameters (data type):

None.

function:

save current configurations and overwrite to the ini file.

return value:

(int): execution state.

Remarks:

For example, you can save the TCP offset you have confirmed, and no need to set again after controller reboot.

Part IV Motion Parameter Configurations:

IV-1. get_tcp_pose(fp32 pose[6]):

parameters (data type):

pose (fp32*): [output] Cartesian coordinate of TCP: [X Y Z A B C] dimension 6.

X Y Z – Cartesian coordinate of TCP position, unit is mm.

A B C – Cartesian coordinate of TCP orientation, unit is radian.

Expressed in X-Y-Z fixed frame rotation.

function:

Obtain current TCP Cartesian coordinate.

return value:

(int): execution state.

IV-2. get_joint_pose(fp32 angles[7]):

parameters (data type):

angles (fp32*): [output] current joint angles from joint1 to joint7, unit is radian.

function:

Obtain current joint angles.

return value:

(int): execution state.

IV-3. get_fk(fp32 angles[7], fp32 pose[6]):

parameters (data type):

angles (fp32*): [input] list of joint angles for inquiry, dimension 7, unit is radian.

pose (fp32): [output] Cartesian coordinate of TCP: [X Y Z A B C] dimension 6.

X Y Z – Cartesian coordinate of TCP position, unit is mm.

A B C – Cartesian coordinate of TCP orientation, unit is radian.

Expressed in X-Y-Z fixed frame rotation.

function:

Obtain the corresponding TCP Cartesian coordinate for specified joint angles.

return value:

(int): execution state.

IV-4. is_joint_limit(fp32 joint[7], int *value):

parameters (data type):

joint (fp32*): [input] list of joint angles for inquiry, dimension 7, unit is radian.

value (int*): [output] inquiry result.

0 – not exceed limit.

1 – exceed joint limit.

function:

Inquire if the specified joint angles would violate joint limits.

return value:

(int): execution state.

Part V Servo Information Related:

V-1. servo_get_dbmsg(u8 rx_data[16]):

parameters (data type):

rx_data (u8*): [output] servo state info, dimension 16.

rx_data[0,1]: robot joint 1 [state, error_message]

rx_data[2, 3]: robot joint 2 [state, error_message]

rx_data[4~13]: robot joint 3~7 [state, error_message]

rx_data[14,15]: gripper [state, error_message]

function:

Obtain servo motor state information.

return value:

(int): execution state.

Remarks:

[state]:

0 – normal,

1 – error detected,

3 – communication failure.

[error_message]:

If there is any non-zero error code returned, please provide to us for diagnose.

Appendix I. Execution State Explanation

Code	Description
0	Normal
1	Xarm has error code
2	Xarm has warning code
3	Communication time out
4	Instruction parsing error
5	Instruction parsing error
6	Instruction parsing error
7	Instruction parsing error
8	Network problem
11	Unknown error

Appendix II. Error and Warning Code Explanation

Error Code	Description
10, 11~17	Servo error, 11~17 representing joint1 ~ joint7
21	Inverse Kinematics solution error
22	Collision detection true
23	Joint angle limit violation
24	Joint speed limit violation
25	Trajectory planning error
26	Linix real-time timing error
27	Communication reply failure
28	Gripper error
29	Others

Warning Code	Description
11	Command buffer is full
12	Parameter incorrect
13	Instruction not exist
14	Command has no solution