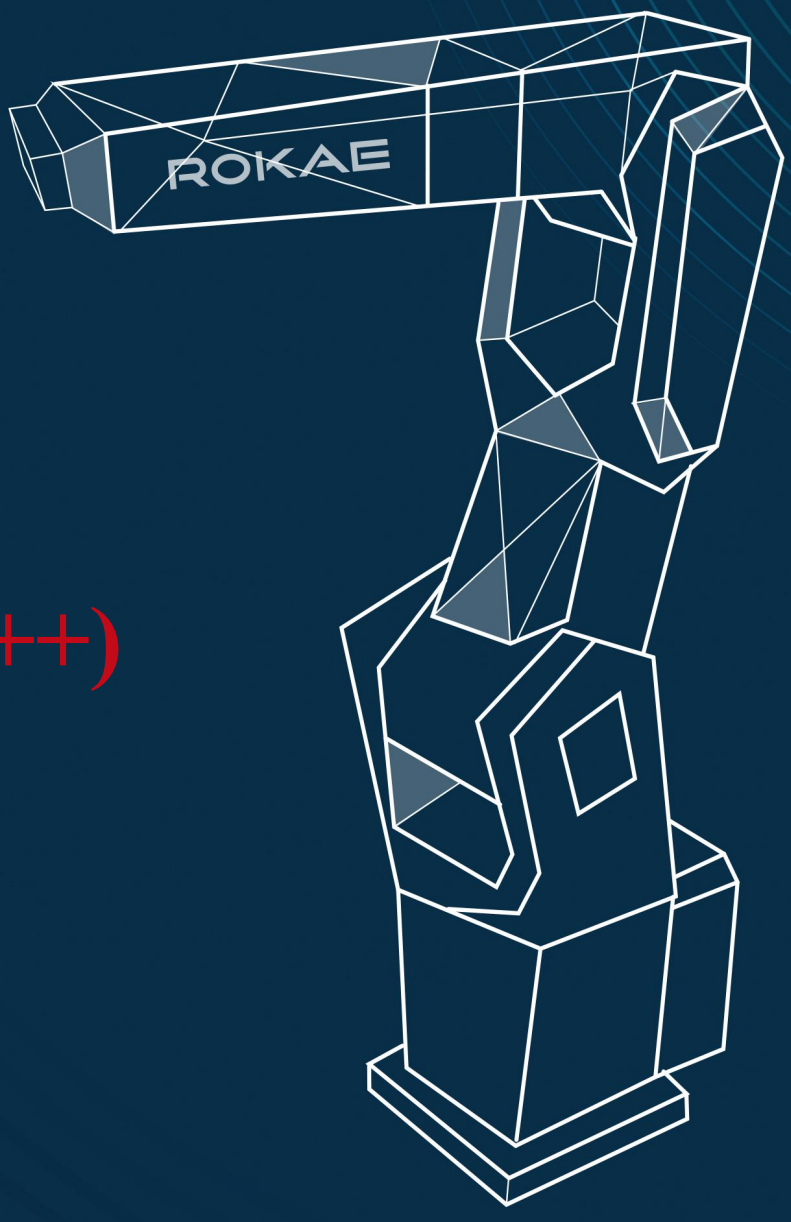
ROKAE

Light-Weight Robot Expert



xCore SDK(C++)

User Manual

Take the Efficiency of Intelligent Manufacturing to the Next Level

xCore SDK(C++) User Manual

[Type]

[Remarks]
Control System Version: V3.1.1
Document Version: A

Contents in the Manual are subject to change without notice. We assume no responsibility for any errors that may appear in the Manual.
We hope you can understand that in no event shall we be liable for incidental or consequential damages arising from the use of the Manual and the products described herein.
We cannot foresee all possible dangers and consequences. Therefore, the Manual cannot warn the user of all possible hazards.
No part of the Manual may be reproduced in any form.
If you find the contents of the Manual wrong or in need of improvement or supplement, please contact us for correction.
The original language of the Manual is Chinese, and all other language versions are translated from the Chinese version.

Contents

1 Manual Overview

1.1 About the Manual

Thank you for choosing our ROKAE robot system.

The Manual contains the following instructions for the correct installation and use of the robot:

- Use of SDK (C++) for secondary development of robots.

Please read the Manual and other related manuals carefully before installing and using the robot system.

After reading, keep it properly for future reference.

1.2 Target group

The Manual is intended for:

- Robot application development engineers.

Please ensure that the above personnel have the necessary knowledge of robot operation and C++ programming, and have received our training.

1.3 How to read the Manual

The Manual includes a separate safety section that must be read through before proceeding with any installation or maintenance procedures.

1.4 Revision History

Version No.	Date	Explanation
V1.5	June 2022	Initial version;
V1.6	September 2022	ROKAE SDK initial version for xCore V1.6.2;
V1.7	February 2023	ROKAE SDK official version for xCore V1.7;
V1.8	June 2023	Add interface, and optimize and correct some problems for xCore v2.0;
V1.9	October 2023	Add interface, and correct some problems for xCore v2.1
V2.0	April 2024	Add several interfaces, and correct some problems for xCore v2.2
v0.4.1	July 2024	Add interfaces for emergency stop reset for xCore version v2.2.2
v0.5.0	December 2024	Add several interfaces, and correct some problems for xCore v3.0.1
v0.6.0	June 2025	Add several interfaces, and correct some problems for xCore v3.1

2 Overview

The xCore SDK API library is a software product designed for secondary development of ROKAE robots. It enables users to control and operate robots equipped with the xCore system, including real-time and non-real-time motion control, robot communication-related read/write, query and execution of RL projects, and so on. This manual describes how to use the API library and the features of API functions. Users can develop their applications and integrate them into external software and hardware modules.

2.1 Compatibility

2.1.1 Controller Version and Robot Model

- Controller version: xCore v3.1.1 and later.
- Robot model: All robot models. The callable APIs may vary according to the features of collaborative robots and industrial robots.

2.1.2 Compiler Platform and Language

Operating system	Compiler	Platform	Language
Ubuntu 18.04/20.04/22.04	build-essential	x86_64 aarch64	C++, Python
Windows 10	MSVC 14.1+	x86_64	C++, Python, C#
Android		armeabi-v7a， arm64-v8a， x86，x86_64	Java

2.2 Non-Real-Time Control

The xCore SDK provides non-real time control for robots, primarily by sending movecommands to the robots. It utilizes the trajectory planning within the controller to complete path planning and move execution. Operations available in the non-real-time mode include:

- Moves: Joint space move (MoveAbsJ, MoveJ), Cartesian space move (MoveL, MoveC, MoveCF, MoveSP), supporting for coordinated move with rail, reachability verification, acceleration setting, etc.
- Force control commands
- Robot communication: Digital and analog I/O, register read/write, 485 communication at end-effector for XMS/XMC models
- Query and execution of RL projects
- Direct teaching control and path playback (only for xMate collaborative robots)
- Other operations: Jog, collision detection setup, soft limit configuration, alarm clearance, controller log query, etc.

2.3 Real-Time Control

The real-time control offered by the xCore SDK involves a series of underlying control interfaces. It allows real-time control of up to 1000 Hz for algorithm verification and development of new applications undertaken by researchers or secondary development users.

The xMate collaborative robot supports 5 control modes:

- Joint space position control
- Cartesian space position control
- Impedance control of joint space
- Cartesian impedance control
- Direct torque control

Two position control modes are available for axis 6 industrial robots:

- Joint space position control
- Cartesian space position control

The modes are not available for 3-axis and 4-axis industrial robots.

3 Operations Guide

This section describes how to configure and run an xCore SDK C++ program.
For other programming languages such as Python, Java and C#, please refer to the relevant manual.

3.1 Setting

3.1.1 Hardware Configurations

For the settings of the robot body, control cab and other hardware, please refer to the *xCore Control System User Manual V3.1.1*. Except for network configurations, no additional hardware configurations are required to use the xCore SDK.

3.1.2 Network Configurations

The xCore SDK is connected to the robot via Ethernet (TCP/IP). The user PC and the robot are connected to the same LAN using a wired or wireless connection. A wireless connection can be used for non-real-time control that does not require high network performance.
For real-time control, a wired connection to the robot is recommended. The robot has two network interfaces, including one external network interface and one direct connect network interface. The default static IP of the direct connect network interface is 192.168.0.160. There are two ways to connect to the robot:

- Method 1: Direct cable connection between the robot and the user PC. The IP address of the user IPC and the static IP address of the robot should be in the same network segment, e.g. 192.168.0.22.
- Method 2: The robot's external network interface is connected to the router or switch, and the user PC is also connected to the router or switch with both of them in the same LAN.

Note: Method 1 is recommended for connection, as Method 2 may cause unstable robot motion in case of poor network communication.

3.1.3 Robot Function Settings

The xCore SDK can be used directly to control the robot without any settings in RobotAssist.
Switch to the real-time control mode and then restart the robot, and the real-time control mode will stay on and the robot will automatically switch to the auto mode.

3.2 Compilation

3.2.1 Description of xCore SDK

```
xCoreSDK
├── doc: Documentation and User Manual
├── example: Example Program
├── external: Eigen
├── include: Header Files
└── lib: Library Files for Operating Systems and Architectures
```

3.2.2 C++ Compilation

The xCore SDK requires CMake version 3.12 or higher for project building.

3.2.2.1 Compilation in Linux

Take compilation of the sample program sdk_example as an example. The installation directory is "out" under the root directory:

```
cd xCoreSDK-v0.5.0
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=../out
cmake --build . --target sdk_example
cmake --build . --target install
```

3.2.2.2 Compilation in Windows

1. Download and install Microsoft Visual Studio 2017 or later versions. Select and install "Desktop development with C++";
2. Open a CMake project. Select CMakeLists.txt in the root directory;
3. Select "Release" or "Debug" to compile the sample program.

3.2.2.3 Compilation in QT

1. Download and install Qt 5.15.2 or later versions. Select MSVC2019 as the compiler;
2. Save the SDK project package locally (without a Chinese path);
3. Create a new project file and select the MSVC2019 compiler;
4. Enter the configuration file (.pro file) and enter the following configuration statements:

```
DEFINES += _USE_MATH_DEFINES
LIBS += -L<path-to-library-directory> -lxCoreSDK

INCLUDEPATH += <path-to-include-directory>

#Eigen configuration
INCLUDEPATH += <path-to-external-directory>
```

Static library compilation example:

```
DEFINES += _USE_MATH_DEFINES

#SDK
LIBS += -L$$PWD/xCoreSDK-v0.6.0/lib/Windows/Debug/64bit -lxCoreSDK
INCLUDEPATH += xCoreSDK-v0.6.0/include

#Eigen
INCLUDEPATH += xCoreSDK-v0.6.0/external
```

3.3 Locale configuration files

3.3.1 Operation logs

After running the SDK program and connecting to the robot, the SDK will print logs to a file when calling interfaces such as setting data and executing operations, including call parameters and return results. When encountering problems, the logs can be provided to ROKAE's technical personnel for analysis and troubleshooting.
The logs are set by default as follows:

- File path:<current working directory>/_rokae_log_/<Robot uid>/xCoreSDK1<YYYY-MM-DD>.log
- Retain logs for up to 7 days

Users can now customize the log output location and retention period by placing a file named user_log_settings.json in the program's runtime directory. The file structure is as follows:

```
{
    "_log_storage_directory_": "user_defined_log_storage_directory",
    "_retention_days_": 5
}
```

_log_storage_directory_is the user-defined directory path for log storage, replacing the default directory rokae_log. It is recommended to write with a relative path method; _retention_days_ is the number of days to retain log files.
Global logs are also provided, with the output location fixed in the folder named logs under the directory where the running program is located, with a maximum retention of 7 days.

3.3.2 Real-time mode waiting timeout setting

The default waiting timeout time for receiving status data in real-time mode is 1ms. Based on the network reality on site, a configuration file (xcoresdk_config.json) can be added to the executable files in the same directory to set the timeout time, in order to adapt to the actual situation and optimize the real-time effect. If the device has good network connectivity but experiences delayed execution of motion commands after a period of control, the timeout value can be appropriately increased within the range of 1-4ms. The specific JSON file format is as follows:

```
{
    "rt":{
        "_timeout_": 1
    }
}
```

3.4 Language

The error code information and exception information of xCore SDK support both Chinese and English, depending on the system language set by the user's PC. Return Chinese error messages when the display language is set to Chinese; return English error messages for all other languages

3.5 Linux Real-time Environment Configuration (Optional)

The real-time mode of the xCore controller receives motion commands in 1ms, the client should ensure a sending cycle of at least 1kHz. A real-time kernel is recommended for large computational load.

1. Install the dependencies

```
apt-get install build-essential bc curl ca-certificates fakeroot gnupg2 libssl-dev lsb-release libelf-dev bison flex cmake libeigen3-dev
```

2. Download real-time kernel patches

The uname -r command can be used to learn about the kernel being used locally;

The kernel closest to the current kernel version can be found on https://www.kernel.org/pub/linux/kernel/projects/rt/;
Download files:

```
$ curl -SLO https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.14.12.tar.xz
$ curl -SLO https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.14.12.tar.sign
$ curl -SLO https://www.kernel.org/pub/linux/kernel/projects/rt/4.14/older/patch-4.14.12-rt10.patch.xz
$ curl –SLO https://www.kernel.org/pub/linux/kernel/projects/rt/4.14/older/patch-4.14.12-rt10.patch.sign
```

If the domestic connection is too slow, you can download directly from the website or find another mirror source;
Decompress:

```
$ xz -d linux-4.14.12.tar.xz
$ xz -d patch-4.14.12-rt10.patch.xz
```

Check the integrity of the sign file

```
$ gpg2 --verify linux-4.14.12.tar.sign
```

You will get information similar to the following:

```
$ gpg2 --verify linux-4.14.12.tar.sign gpg: assuming signed data in 'linux-4.14.12.tar
gpg: Signature made Fr 05 Jan 2018 06:49:11 PST using RSA key ID 6092693E
gpg: Can't check signature: No public key
```

Write down ID 6092693E and execute:

```
$ gpg2 --keyserver hkp://keys.gnupg.net --recv-keys 0x6092693E
```

Perform the same on the patch files.
Verify again the downloaded server key. It is correct if the following information is displayed.

```
$ gpg2 --verify linux-4.14.12.tar.sign
gpg: assuming signed data in 'linux-4.14.12.tar'
gpg: Signature made Fr 05 Jan 2018 06:49:11 PST using RSA key ID 6092693E
gpg: Good signature from "Greg Kroah-Hartman <gregkh@linuxfoundation.org>"[unknown]
gpg: aka "Greg Kroah-Hartman <gregkh@kernel.org>" [unknown]
gpg: aka "Greg Kroah-Hartman (Linux kernel stable release signing key)
<greg@kroah.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner. Primary key
fingerprint: 647F 2865 4894 E3BD 4571 99BE 38DB BDC8 6092 693E
```

Verify the patch files as well.

3. Build the kernel

Decompress:

```
$ tar xf linux-4.14.12.tar
$ cd linux-4.14.12
$ patch -p1 < ../patch-4.14.12-rt10.patch
```

Configure the kernel:

```
$ make oldconfig
```

The following information displays:

```
Preemption Model
  1. No Forced Preemption (Server) (PREEMPT_NONE)
  2. Voluntary Kernel Preemption (Desktop) (PREEMPT_VOLUNTARY)
  3. Preemptible Kernel (Low-Latency Desktop) (PREEMPT__LL) (NEW)
  4. Preemptible Kernel (Basic RT) (PREEMPT_RTB) (NEW)
  > 5. Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL) (NEW)
```

Select 5 and keep entering.
Start building:

```
$ fakeroot make -j4 deb-pkg
```

Install dpkg:

```
$ sudo dpkg -i ../linux-headers-4.14.12-rt10_*.deb ../linux-image-4.14.12-rt10_*.deb
```

4. Verify if the installation is successful

Reboot and access ubuntu advanced options to check the installed kernel. Select the newly installed kernel and check the corresponding kernel version with the uname -r command. If the version is correct, /sys/kernel/realtime will display 1.

4 Interface description

This section contains the APIs supported by different versions of xCore SDK and the description of their features. The functional definitions of interfaces are generally consistent across different development language versions, but there may be differences in parameters, return values, and calling methods.

4.1 API Support

The following table is an overview of APIs supported by different languages.

Module	API Features	C++	Python & C#	Android
rokae::Robot	Basic operations	All supported	All supported	All supported
	Non-real-time motion	All supported	All supported	All supported
	Jog robot	All supported	All supported	All supported
	Communication	All supported	All supported	All supported
	RL Project	All supported	All supported	All supported
	Collaboration-related	All supported	Partially supported	All supported
rokae::Model	Kinematics calculation	All supported	All supported	All supported
rokae::ForceControl	Force control commands	All supported	All supported	All supported
rokae::RtMotionControl	Real-time mode	All supported	Not supported	Not supported
rokae::Planner	Host computer path planning	All supported	Not supported	Not supported
rokae::xMateModel	Kinematics and Dynamics Calculation	All supported	Not supported	Not supported

4.2 C++: Instantiation ROKAE::Robot Class

The C++ SDK offers the following robot classes for instantiation according to different robot configurations and axes. During initialization, the selected configurations and axes will be checked to see if they match the connected robot:

Class	Applicable model
xMateRobot	6-axis collaborative robot
xMateErProRobot	7-axis collaborative robot
StandardRobot	6-axis industrial robot
xMateCr5Robot	CR5-axis collaborative robot
PCB4Robot	4-axis industrial robot
PCB3Robot	3-axis industrial robot

4.3 Basic Robot Operations and Information Query

Description	Interface	Parameter	Return
Connecting to the robot	connectToRobot()		
	connectToRobot(remoteIP, localIP)	RemoteIP - robot IP address LocalIP - local IP address. For data transmission and reception in real-time mode	
Disconnect from the robot	disconnectFromRobot()		
Set up a disconnection callback function	setConnectionHandler(handler)	Handler - callback function with bool parameter	
Query basic robot information	robotInfo()		Controller version, robot model, number of axes
Robot power on/off and emergency stop statuses	powerState()		on/off/Estop/Gstop
Robot power on/off	setPowerState(state)	state - on/off	
Query current operating mode	operateMode()		auto/manual
Switch between automatic/manual mode	setOperateMode(mode)	mode - auto/manual	
Query robot status	operationState()		Robot status, such as idle/jog/RLprogram/moving
Get current end-effector/flange posture	posture(ct)	ct - frame type	[X, Y, Z, Rx, Ry, Rz]
Get current end-effector/flange posture	cartPosture(ct)	ct - frame type	[X, Y, Z, Rx, Ry, Rz] and joint configuration parameters
Get current joint angle	jointPos()		Axis angle (rad)
Get current joint velocity	jointVel()		Axis velocity (rad/s)
Get joint torque	jointTorque()		Axis torque (Nm)
Query multiple statuses	getStateList()		Current position, IO signal, operating mode, speed coverage value
Query base frame	baseFrame()		[X, Y, Z, Rx, Ry, Rz]
Set up a base frame	setBaseFrame(frame)	Frame	

Query current tool & work object	toolset()		End-effector frame, reference frame, load information
Set tools & work objects	setToolset(toolset) setToolset(toolName, wobjName)	toolset - tool & work object information toolName - tool name wobjName - work object name	
Inverse kinematics	calcIk(posture) calcIk(posture,toolset)	posture - end-effector posture relative to external reference frame toolset - tool & work object information	Joint angle
Forward kinematics	calcFk(joints) calcFk(joints,toolset)	joints - joint angle toolset - tool & work object information	End-effector posture relative to external reference frame
Clear servo alarms	clearServoAlarm()		
Query controller logs	queryControllerLog(count, level)	count - number of queries level - log level	List of controller logs
Set collision detection parameters and enable collision detection.	enableCollisionDetection (sensitivity, behaviour, fallback)	sensitivity – collision detection sensitivity behavior – behavior after collision fallback – fallback distance/compliance	
Disable collision detection	disableCollisionDetection()		
Frame calibration	calibrateFrame(type, points, is_held, base_aux)	type - frame type points - calibration axis angle list is_held - handheld/external tool work object base_aux - base frame calibration auxiliary point	Calibration results: frame and deviation
Get the current soft limit value	getSoftLimit(limits)	limits - Soft limit of each axis	On/Off
Set soft limitation	setSoftLimit(enable, limits)	enable – On/Off limits - Soft limit of each axis	
Restore state	recoverState(item)	item - recovery option 1: Emergency stop recovery	
Restart IPC	rebootSystem()		
Set rail parameters	setRailParameter(name, value)	name - parameter name value - parameter value	
Read rail parameter	getRailParameter(name, value)	name - parameter name value - parameter value	
Configure NTP	configNtp(server_ip)	server_ip - NTP server IP	
Manually synchronize one-time NTP time	syncTimeWithServer()		
Query SDK version	sdkVersion()		Version No.
Teach pendant hot plug	setTeachPendantMode(enable)	enable - use teach pendant or not	

4.4 Motion control

APIs for non-real-time motion control.

Description	Interface	Parameter	Return
Set motion control mode	setMotionControlMode(mode)	mode - NRT/RT/RL project	
Start/continue moving	moveStart()		
Motion reset	moveReset()		
Pause robot motion	stop()		
Add motion command	moveAppend(command, id)	command – one or multiple MoveL/MoveJ/MoveAbsJ/MoveC/MoveCF/MoveSP command (s) id – command ID, used to execute information feedback	
Set default speed	setDefaultSpeed(speed)	speed - maximum linear speed of the robot end-effector	
Set default turning zone	setDefaultZone(zone)	zone - radius of turning zone	
Set whether to use conf	setDefaultConfOpt(forced)	forced – use or not	
Set the maximum number of cache commands	setMaxCacheSize(number)	number	
Set whether motion commands automatically cancel the turning zone	setAutoIgnoreZone(enable)	enable - whether to automatically cancel the turning zone	

Start jogging the robot	startJog(space, rate, step, index, direction)	space - reference frame rate - rate step - step length index - XYZABC/J1-7 direction - direction	
Dynamically adjust the robot's motion speed	adjustSpeedOnline(scale)	scale - speed	
Set the callback function for receiving events	setEventWatcher(eventType, callback)	eventType - event type callback - callback function of handling an event	
Query event information	queryEventInfo(eventType)	eventType - event type	Event information
Execute motion commands	executeCommand(command)	command – one or multiple MoveL/MoveJ/MoveAbsJ/MoveC/MoveCF/MoveSP command (s)	
Read the current acceleration	getAcceleration(acc, jerk)	acc - acceleration jerk - jerk	
Set motion acceleration	adjustAcceleration(acc, jerk)	acc - acceleration jerk - jerk	
Enable singularity avoidance function	setAvoidSingularity(method, enable, threshold)	method - singularity avoidance method enable – On/Off threshold - threshold parameter	
Query whether to enable the singularity avoidance function	getAvoidSingularity(method)	method - singularity avoidance method	On/Off
Query whether Cartesian trajectory is reachable	checkPath(start, start_joint, target)	start - start point start_joint - start joint angle target - target point	Calculated target axial angle

4.5 Real-time motion control

Description	Interface	Parameter	Return
Reconnect to real-time control server	reconnectNetwork()		
Disconnect from the robot	disconnectNetwork()		
Set periodic scheduling	setControlLoop(callback, priority, useStateDataInLoop)	callback - callback function priority - thread priority useStateDataInLoop – confirm whether to use real-time state data in loop	
Start executing scheduling task	startLoop(blocking)	blocking - blocking or non-blocking	
Stop scheduling task	stopLoop()		
Start motion	startMove(mode)	mode - control mode	
Stop motion	stopMove()		
Start receiving real-time status data	startReceiveRobotState(timeout , fields)	timeout - timeout waiting time fields - received data list	
Stop receiving real-time status data	stopReceiveRobotState()		
Update current robot status data	updateRobotState(timeout)	timeout - timeout waiting time	
Get robot status data	getStateData(name, data)	name - data name data - data value	
Joint space PTP motion planning	MoveJ(speed, start, target)	speed - velocity factor start - start joint angle target - target joint angle	
Cartesian space PTP linear motion planning	MoveL(speed, start, target)	speed - velocity factor start - start posture target - target posture	
3-point arc motion planning	MoveC(speed, start, aux, target)	speed - velocity factor start - start posture aux - auxiliary point posture target - target posture	
Set clipping and filtering parameters	setFilterLimit(limit, frequency)	limit - whether to enable clipping frequency - cut-off frequency	
Set cartesian space motion area	setCartesianLimit(length, frame)	length - dimensions of motion area frame - regional center frame	
Set joint impedance factor	setJointImpedance(factor)	factor - joint impedance factor	
Set Cartesian space impedance control factor	setCartesianImpedance(factor)	factor - impedance factor	
Set collision detection threshold	setCollisionBehaviour(threshold)	threshold - joint threshold	

Set end-effector posture	setEndEffectorFrame(frame)	frame - end-effector posture relative to the flange	
Set load	setLoad(load)	load - load information	
Set filter cut-off frequency	setFilterFrequency(joint, cart, torque)	joint - joint position cut-off frequency cart - Cartesian space position cut-off frequency torque - joint torque cut-off frequency	
Set end-effector desired force of Cartesian impedance control	setCartesianImpedanceDesiredTorque(torque)	torque - end-effector desired force	
Set filtering parameters	setTorqueFilterCutoffFrequency(frequency)	frequency - frequency	
Set force control frame	setFcCoor(frame, type)	Frame type - frame type for force control tasks	
Automatically recover the robot after an error occurs	automaticErrorRecovery()		
Set network delay threshold	setNetworkTolerance(percent)	percent - threshold percentage	
Switch to RCI client	useGen1RciClient(use)	use - whether to use RCI client	
Whether there is any motion error in real-time mode	hasMotionError()		true - error report

Path planning related

Description	category
Cartesian spatial motion in S-speed planning	CartMotionGenerator
Joint space motion of S-speed planning	JointMotionGenerator
Follow position, which can be Cartesian posture or joint angle	FollowPosition

4.5.1 Parameter setting

Set parameters before the motion starts. All parameters are only used for real-time motion and independent of non-real-time motion and motion controlled by RobotAssist.

4.5.2 Send motion commands

Set the callback function via setControlLoop(), and calculate the moving command for each cycle in the function and return it as the function return value. The SDK filters the returned commands and sends them to the controller. The data type of the command (joint angle/Cartesian posture/joint torque) should match the control mode. The control cycle of the controller is 1ms. The monitoring window of the controller is 2 seconds. According to the network delay threshold, if no enough commands are received within the monitoring window, an error "unstable network connection" will be returned. Then the motion will be stopped, and the robot will be powered off.

4.5.3 Get real-time data

Before the moving starts, set the receiving interval of data and the sending interval of controller via startReceiveRobotState(). Depending on whether the state data is read in the callback function described above, the update mode is different. If you need to read the state data in the callback function, in order to ensure a control cycle of 1 ms, xCore-SDK will update the state data before each callback execution, and directly call getStateDate() inside the callback function to read.

If not, you need to call updateStateData() according to the set sending cycle, and then read it through the getStateDate() interface.

4.5.4 Kinematics and Dynamics Calculation

The SDK provides users with a library for xMate kinematic and dynamics computation, facilitating the computation of robot forward/inverse kinematics, Jacobian matrices, etc. Currently, supported models include all models in the xMateER series, XMC7/12, and XMS3/4/5. (Note: XMS5 is a new model. The model library can only be used after upgrading to a special version of the model file, otherwise the calculation results will be incorrect. Contact ROKAE's technical personnel for model file upgrades). In terms of compatibility, it supports Linux x86_64 and Windows 64-bit. Main API functions include: forward kinematics getCartPose(), inverse kinematics getJointPos(), Jacobian matrix calculation jacobian(), and forward dynamics getTorqueNoFriction().

For details, please refer to Appendix A - C++ API.

4.5.4.1 Compilation and use

To use the kinematics and dynamics computation library, the following options should be added during compilation:

```
cmake .. -DXCORE_USE_XMATE_MODEL=ON
```

It can then be obtained using robot.model().

4.5.5 S-curve trajectory planning API

S-curve trajectory planning ensures velocity and acceleration to be continuously differentiable and motion to be efficient and smooth. S-curve planning can be done for joint space or Cartesian space as required by the users. As it is an offline planning method, it should be made clear that this method does not involve path planning, and the path should be defined and generated by users.

Take S-curve planning in joint space as an example:

```
JointMotionGenerator joint_s(0.2, q_end);    //Create a planner with a target joint angle of q_ded, with a velocity coefficient of 0.2;
joint_s.calculateSynchronizedValues(q_start);  //Specify the initial joint angle q_start and perform synchronization processing;
joint_s.calculateDesiredValues(t, q_delta);    //Calculate the increment q_delta of the joint angle relative to q_start at time t;
```

For details, please refer to planner.h in SDK.

4.5.6 Error exception

The robot will check its status during motion, and report detected errors to the SDK. Users can determine the type of errors via the following 20 error bits.

Error bit	Error name	Cause	Solution
0	kActualJointPositionLimitsViolation	Actual axis angle exceeds limit	Check whether the planned trajectory is continuously differentiable. Generally, the planned trajectory needs to be continuously differentiable in terms of velocity and angular velocity. Check whether the trajectory is smooth when the robot is running unsteadily or with strange noises. An extra filtering process is needed when necessary.
1	kActualCartesianPositionLimitsViolation	Actual end-effector posture exceeds limit	
2	kActualCartesianMotionGeneratorElbowLimitViolation	Actual arm angle exceeds limit	
3	kActualJointVelocityLimitsViolation	Actual axis velocity exceeds limit	
4	kActualCartesianVelocityLimitsViolation	Actual end-effector velocity exceeds limit	
5	kActualJointAccelerationLimitsViolation	Actual axis acceleration exceeds limit	
6	kActualCartesianAccelerationLimitsViolation	Actual end-effector acceleration exceeds limit	
7	kCommandJointPositionLimitsViolation	Command axis angle exceeds limit	
8	kCommandCartesianPositionLimitsViolation	Command end-effector posture exceeds limit	
9	kCommandCartesianMotionGeneratorElbowLimitViolation	Command arm angle exceeds limit	
10	kCommandJointVelocityLimitsViolation	Command axis velocity exceeds limit	
11	kCommandCartesianVelocityLimitsViolation	Command end-effector velocity exceeds limit	
12	kCommandJointAccelerationLimitsViolation	Command axis acceleration exceeds limit	
13	kCommandCartesianAccelerationLimitsViolation	Command end-effector acceleration exceeds limit	
14	kCommandJointAccelerationDiscontinuity	Command axis acceleration is not continuous	
17	kCommandTorqueDiscontinuity	Command torque is not continuous	
18	kCommandTorqueRangeViolation	Command torque exceeds limit	
15	kCollision	Collision detected	If collision detection is triggered frequently, the collision detection threshold should be raised appropriately. This error may also be triggered by an emergency stop
16	kCartesianPositionMotionGeneratorInvalidFrame	Robot singularity	The robot should not be in a singular posture during Cartesian space motion
19	kInstabilityDetection	Unstable operation detected	1. Check whether the packet loss threshold is too low. The range is usually set to 10–20; 2. Servo error. Usually it is necessary to restart the robot; 3. This error may also be triggered by pressing the emergency stop switch.

4.6 Communication

Description	Interface	Parameter	Return value
Query DI signal value	getDI(board, port)	board – IO board serial number port – signal port number	on | off
Set DI signal value	setDI(board, port, state)	board – IO board serial number port – signal port number state – signal value	
Query DO signal value	getDO(board, port)	board – IO board serial number port – signal port number	on | off
Set DO signal value	setDO(board, port, state)	board – IO board serial number port – signal port number state – signal value	
Query AI signal value	getAI(board, port)	board – IO board serial number port – signal port number	Signal value
Set AO signal	setAO(board, port, value)	board – IO board serial number port – signal port number value - signal value	
Set simulation mode	setSimulationMode(state)	state – on/off	
Read register value	readRegister(name, index, value)	name - register name index - register array index value - read value	
Write register value	writeRegister(name, index, value)	name - register name index - register array index value - write value	
Set xPanel external power supply mode	setxPanelVout(opt)	opt - mode	
Get the state of end-effector keypad	getKeypadState()		State of end-effector keypad
End-effector 485 communication switch	setxPanelRS485(Vopt,if_rs485)	Vopt - output voltage if_rs485 - whether to enable 485 communication switch	
485 communication read-write register	XPRWModbusRTUReg(slave_addr, fun_cmd, reg_addr data_type, num, data_array, if_crc_reverse)	slave_addr - slave address fun_cmd - function code reg_addr - register address data_type - data type num - data length data_array - data array if_crc_reverse - whether CRC reversal is required	
485 communication read-write coil or discrete input	XPRWModbusRTUCoil(slave_addr, fun_cmd, coil_addr, int num, data_array, if_crc_reverse)	slave_addr - slave address fun_cmd - function code coil_addr - coil address num - data length data_array - data array if_crc_reverse - whether CRC reversal is required	
485 communication bare data transmission	XPRS485SendData(send_byte, rev_byte, send_data, rev_data,)	send_byte - data transmission length rev_byte - data receiving length send_data - data transmission rev_data - data receiving	

4.7 RL Project

An RL project that allows information query and execution needs to be created in the controller.

Description	Interface	Parameter	Return value
Query RL project list	projectInfo()		Project name and task name
Load project	loadProject(name, tasks)	name - project name tasks - task list	
pp-to-main	ppToMain()		
Start running project	runProject()		
Pause running project	pauseProject()		
Set running rate and loop mode	setProjectRunningOpt(rate, loop)	rate - running rate loop - loop/single cycle	
Query tool information	toolsInfo()		Tool name, posture, load, etc.

Query work object information	wobjsInfo()		Work object name, posture, load, etc.
Import the local RL project package into the controller	importProject(file_path, overwrite)	file_path - local.zip package path overwrite - whether to overwrite files with the same name	Project name
Remove the RL project from the controller	removeProject(project_name, remove_all)	project_name - project name remove_all whether to remove all	
Import local files to the controller	importFile(src_file_path, dest, overwrite)	src_file_path - local file path dest - target path overwrite - overwrite files with the same name	File name after successfully imported
Remove files from the controller	removeFiles(file_path_list)	file_path_list - list of file paths	
Set global tool information	setToolInfo(tool_info)	tool_info - tool information	
Set global work object information	setWobjInfo(wobj_infoc)	wobj_info - work object information	

4.8 Collaboration-related

Relevant features include direct teaching control and path recording.

Description	Interface	Parameter	Return value
Enable Drag mode	enableDrag(space, type, enable_drag_button)	space - drag space type - drag type enable_drag_button - no button drag	
Disable Drag mode	disableDrag()		
Start path recording	startRecordPath(duration)	duration - recording duration	
Stop path recording	stopRecordPath()		
Cancel recording	cancelRecordPath()		
Save path	saveRecordPath(name, saveAs)	name - path name saveAs - rename as	
Replay path	replayPath(name, rate)	name - path name rate - playback speed	
Delete saved path	removePath(name, all)	name - path name all - whether to delete all paths	
Query path list	queryPathLists()		Path name list
Force sensor calibration	calibrateForceSensor(all_axes, axis_index)	all_axes - calibrate all axes axis_index - single axis calibration subscript	

4.9 Force control commands

Description	Interface	Parameter	Return value
Get the current torque information	getEndTorque(ref_type, joint, external, cart_torque, cart_force)	ref_type - reference frame joint for relative torque - measurement of each axis external - external force on each axis cart_torque - Cartesian space torque cart_force - Cartesian space force	
Initialize the force control	fcInit(frame_type)	frame_type - force control frame	
Start force control	fcStart()		
Stop force control	fcStop()		
Set impedance control type	setControlType(type)	type - impedance type	
Set the load used by the force control module	setLoad(load)	load - load	
It is used to set the joint impedance stiffness	setJointStiffness(stiffness)	stiffness - stiffness	
It is used to set the Cartesian impedance stiffness	setCartesianStiffness(stiffness)	stiffness - stiffness	
Set the Cartesian null-space impedance stiffness	setCartesianNullspaceStiffness(stiffness)	stiffness - stiffness	
Set the desired torque of the joint	setJointDesiredTorque(torque)	torque - torque	
Set the desired Cartesian force/torque	setCartesianDesiredForce (value)	value - expected force/torque	
Set the sine overlay rotating around a single axis	setSineOverlay(line_dir, amplify, frequency, phase, bias)	line_dir - reference axis amplify - amplitude frequency - frequency phase - phase	

		bias - bias	
Set the Lissajous overlay within a plane	setLissajousOverlay (int plane, double amplify_one, double frequency_one, double amplify_two, double frequency_two, double phase_diff, error_code &ec)	plane - reference plane amplify_one - direction I amplitude frequency_one - direction I frequency amplify_two - direction II amplitude frequency_two - direction II frequency phase_diff phase difference	
Start the overlays	startOverlay()		
Stop the overlays	stopOverlay()		
Pause the overlay	pauseOverlay()		
Restart the paused overlays	restartOverlay()		
Define termination conditions related to contact force	setForceCondition (range, isInside, timeout)	range - force limitation isInside - stop waiting when exceeding/meeting the limiting conditions timeout - time-out period	
Define termination conditions related to contact torque	setTorqueCondition (range, isInside, timeout)	range - torque limitation isInside - stop waiting when exceeding/meeting the limiting conditions timeout - time-out period	
Define termination conditions related to contact location	setPoseBoxCondition(supervising_frame, box, isInside, timeout)	supervising_frame - reference frame where the cuboid is located box - cuboid isInside - stop waiting when exceeding/meeting the limiting conditions timeout - time-out period	
Activate the termination conditions set and wait	waitCondition()		
Activate/deactivate force control module protection monitoring	fcMonitor(enable)	enable - On|Off	
Set the maximum axial velocity in force control mode	setJointMaxVel(velocity)	velocity - axial velocity	
Set the maximum velocity of robotic arm end-effector relative to base frame	setCartesianMaxVel(velocity)	velocity - end-effector velocity	
Set the maximum axial momentum in force control mode	setJointMaxMomentum(momentum)	momentum - momentum	
Set the maximum axial energy in force control mode	setJointMaxEnergy(energy)	energy - energy	

4.10 Error code and exception

4.10.1 Error code

The call results of non-real-time interfaces are fed back through the error code, and each interface passes a parameter of std::error_code. The information corresponding to the error code can be obtained through error_code::message().

Value	Error message	Cause and solutions
0	Completed successfully	N/A
-1	Error	It may be caused by unrecognized error code. Please provide feedback to technical personnel
-3	The emergency stop button of the robot is pressed, please restore it first	Restore emergency stop state
-16	This operation is not allowed to be performed in the current mode (manual/automatic), please switch to another mode	Switch between manual and automatic modes
-17	This operation is not allowed to be performed in the current power on/off state	Switch power on/off state
-18	This operation is not allowed to be performed while the robot is running	The robot may be in drag/real-time mode control/recognition mode rather than idling
-19	This operation is not allowed to be performed in the current control mode (position/force control)	Switch position control/force control
-20	In robot motion	Stop robot motion
-32	Inverse kinematics calculation error	Input correct posture

-33	The inverse kinematics is at a singularity point	Avoid singularity
-34	Inverse kinematics exceeds the robot's soft limit	Check if the soft limit setting is appropriate, and input the posture within the limit
-35	Target point exceeds range of motion	Input reachable point position
-36	Target point exceeds range of motion	Input reachable point position
-37	The single step distance is too large to calculate the inverse kinematics	Input reachable point position
-38	Configuration data CFX error, inverse kinematics is not found	Check conf data
-39	Input data error, inverse kinematics cannot be calculated	Input reachable point position
-40	The inverse kinematics reference point is a singularity	Input reachable point position
-41	Algorithm fails, inverse kinematics cannot be calculated	It may be caused by the controller calculation issues, please provide feedback to the technical personnel
-257	Communication protocol parsing failed	Please provide feedback to the technical personnel
-258	No matching data name found	Possible incompatibility between controller and SDK version
-259	No matching command name found	Possible incompatibility between controller and SDK version
-260	Data value error, possibly due to communication protocol mismatch	Possible incompatibility between controller and SDK version
-272	No data name to be queried found	Possible incompatibility between controller and SDK version
-273	Data unreadability	Possible incompatibility between controller and SDK version
-277	Non-corresponding function codes before and after file transfer	Simultaneous transfer of multiple files. Only one file is transferred at a time
-278	Project file package error - requiring .mod and .xml files	RL project does not meet the requirements
-279	The file path does not exist	The local file does not exist, or there is no access permission to files
-280	The files cannot be opened	Local file cannot be opened
-281	Input file type error	File transfer accepts files of zip/json/mod/xml type
-283	The file size exceeds the transferable size	RL project package requires 10M or less
-284	File data transmission failed	Check the network connections
-285	File data transmission timeout	Check the network connections
-288	Data non-writable	Possible incompatibility between controller and SDK version
-289	Data setting failed	The data set is unreasonable
-290	Data setting failed	The data set is unreasonable
-304	Monitoring data failed	Never
-305	The data has been monitored, duplicate monitoring is not supported	Never
-320	Command execution not completed	Never
-337	The recorded path is too short, there are insufficient data points. Please record again	Increase recording time and record again
-338	The recorded path contains instances where joint soft limits are exceeded Please record again	Drag within the soft limit
-339	The recorded path has joints moving at excessive speeds. Please record again	Slow down the drag action
-340	The path recording did not start from the robot's stationary state	Start recording when the robot is stationary
-341	The robot did not stop moving when stopping recording the path	Stop recording when the robot is stationary
-342	The robot is in a power-off state, the path was not saved	Robot power-on
-352	There is no valid path in the buffer, please record	Record the path first and then save it
-353	Failed to save path to disk	Record the drag path again or restart the robot
-513	Switch between manual/automatic operation mode failed	- Stop robot motion - Resume emergency stop - Disable Drag mode
-514	Failed to power on/off	- Resume emergency stop - Clear servo alarms
-515	Opening/closing drag failed, please check if the robot is powered off	When the robot is powered off in manual mode, enable the dragging function

-10005	Failed to reset the load during calibration or in calibration	- Wait for calibration to complete - Set the load correctly
-10030	Simulation mode is not turned on or signal is not present	Open simulation mode and then set DI/AI
-10040	Starting drag failed, correctly set load and calibrate torque sensor	Correctly set the tool load mass and center of mass. Perform torque sensor calibration after setting
-10051	Soft limit exceeds mechanical limit	Set the soft limit within the mechanical limit
-10065	Failed to synchronize NTP time	- Install NTP function properly - Confirm that the NTP function on the server is available and that the IP address is correct
-10101	IP address is illegal	Check if it conforms to the IPv4 address format
-10079	The force control module is in an error state	The force control protection is triggered. Please check if the robot status is normal in force control mode and set reasonable force control protection parameters
-10141	The load mass exceeds the rated load of the robot	Use and set the load within the rated range
-14010	Bound to system IO	- Cancel binding and change signal to system IO - Use other signals
-14501	DO signal does not exist or is a system output	Check if the DO signal has been created
-17001	Failed to read register	- Check if the register has been created - Check for array index out-of-bounds - Read with matching data types
-17002	Failed to write register	- Check if the register has been created - Check for array index out-of-bounds - Or write with matching data types
-17320	After running an RL program, the motion cache should be reset before starting movement	Call moveReset to reset
-17407	Opening the rail requires disabling the safe region	Disable the safe region function
-28672	Real-time mode network error	- The local IP address cannot be localhost or a robot address - There is a port occupancy situation, please check RCI settings - port
-28688	Switching motion control mode failed	- Stop robot motion - Restart the controller
-28689	This frequency is not supported	Status data transmission frequency supports 1kHz, 500Hz, 250Hz, 125Hz
-28705	Motion is in progress	Stop motion before starting again
-28706	The operation cannot be performed, possibly because the robot is not idle	Stop robot motion
-28707	The starting position is a singularity point	Move the robot to a non-singularity point
-28708	Parameter error	Reset data
-28709	The robot is at collision stop	Power on to restore collision state
-28710	The robot is in an emergency stop state	Resume emergency stop
-28711	Request rejected	During impedance control, force control model deviations often occur. Re-calibrate the torque sensor and set the correct load parameters
-41400	There are still force control commands pending execution	Wait for the execution of the preceding force control command to be completed
-41419	TCP length exceeds the limit, force control initialization failed	The length limit of the end-effector is 0.3 m
-41420	No force control initialization has been performed; or the load is not set correctly; or there is a significant deviation in the force control model	- Calibrate zero point and torque sensors; - Set the correct load mass and center of mass; - Check if the base frame is set correctly; - When dragging, the robot did not receive any external force
-41421	Stop force control failed, currently not in force control operation state	The robot is not in force control state
-41422	Failed to restart force control	Pause or stop the force control first, and then restart
-41425	In non-impedance control mode, or overlay in operation	- Ensure that it is in the impedance control mode and that the overlay is stopped - Please check the parameter settings
-41426	In non-impedance control mode, or overlay in operation	- Ensure that it is in the impedance control mode and that the overlay is stopped - Please check the parameter settings
-41427	Not in Cartesian impedance control mode or no overlay set	Check the control mode command, set the overlay parameters before another attempt
-41428	Not in Cartesian impedance control mode or the overlay not been initiated	Please ensure that the overlay is started and it is in the Cartesian impedance control mode

-41429	Not in Cartesian impedance mode or overlay not paused	Pause the current force control task and check the control mode, switch to Cartesian impedance control mode and try again
-41430	Unsupported impedance control type or force control frame	Stop the current force control task and reinitialize the parameters
-41431	Not in joint impedance control mode or not initialized	Set the joint space impedance control mode before operating force control
-41432	Not in Cartesian impedance control mode or not initialized	Set the Cartesian impedance control mode before operating force control
-41433	Not in Cartesian impedance control mode or not initialized	Check the current impedance control mode and try again. Ensure correct input of force value and set Cartesian impedance control mode
-41434	Joint expected force over limit	Check the current impedance control mode and try again. Ensure correct input of expected force value and set joint space impedance control mode
-41435	Expected force of Cartesian space over limit	Check the current impedance control mode and try again. Ensure correct input of expected force value and set Cartesian impedance control mode
-41442	The robot failure to meet the soft limit requirements and failure to activate the force control	Please ensure that the robot is within the software limitation when the impedance mode is enabled for the robot
-41444	Impedance stiffness setting failed, with unreasonable value or impossibility of stiffness setting in the current state	Please reset the impedance stiffness value within a reasonable range
-41448	Power-off during force control process leading to initialization failure	Power on again and initialize force control
-41449	Initialization failure caused by the force control command not been executed yet	The force control command is sent too frequently. Please increase the time between exiting and restarting the force control
-41457	Force control has been stopped and needs to be restarted	Initialize the force control
-41459	Operation is not allowed in force control mode	Stop force control and replay the path
-50000	The angle between the two trajectories is too small, or the length is too short, the turning area cannot be generated	- Increase the angle between the two trajectories; - Increase the length of the two trajectories; - Increase the value of the turning area
-50001	Controller status error, unable to generate trajectory	Call moveReset() to reset
-50002	The target point exceeds the range of motion or is a singularity	- Check the location of the target point; - Move the robot with joints - Check the confdata configuration
-50003	Two adjacent target points are too close	Check whether two adjacent move commands use the same target point
-50004	An arc cannot be generated, as the distance between the starting point and the target point is too close	Adjust the distance between the point positions of the start point and the end point of the arc
-50005	An arc cannot be generated, as the distance between the starting point and the auxiliary point is too close	Adjust the distance between the point positions of the start point and the auxiliary point of the arc
-50006	An arc cannot be generated, as the distance between the auxiliary point and the target point is too close	Adjust the distance between the point positions of the end point and the auxiliary point of the arc
-50007	An arc cannot be generated, as the distance between the starting point, auxiliary point and the target point is too close	Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
-50008	An arc cannot be generated, as the point positions are on a straight line	Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
-50009	An arc cannot be generated, as the radius is too small	Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
-50010	An arc cannot be generated	Adjust the distance between or the orientation of the point positions of the start point, the end point, and the auxiliary point of the arc
-50019	Failed to generate trajectories	Re-adjust the target point position, pose, and arm angle. Note that the arm angle only needs to be considered for 7-axis robots
-50021	There is no solution for the target point under the specified conf parameter. Please check the values or do not set confData	- Call setDefaultConfOpt (false) to cancel confdata - Teach the points again and pass in the correct confdata
-50027	The trajectory is shorter than the minimum turning radius, and the path is automatically stitched	- This trajectory needs to be connected to the turning areas at both the front and the rear, but the trajectory length is less than twice the minimum turning radius; - The trajectory is set to connect to a turning zone, but its length is less than the minimum turning radius. This function can make the movement smoother. To turn off this function, set the minimum turning radius to 0
-50033	The robot is in a locked-axis state, and the locked-axis angle of the target point has deviated. Please adjust the target point or turn off the locked-axis state	Adjust the trajectory target point or disable the axis locking command. When axis-locking is enabled, the angle of 4-axis should be 0°, 180°, or -180°

-50034	The target point cannot be reached in axis-locking mode or with a 5-axis robot configuration	For a 6-axis robot in axis-locking mode or a 5-axis robot, the target point is unreachable. Please adjust the target position or disable lock axis
-50101	The axis angle exceeds the motion range. Try to cancel the soft limit and restore each axis to the allowable range	- Cancel the soft limit - Manually move the robot's axes to the normal working range
-50102	There are trajectories across singularities	- Please avoid singularity - Teach points again and replace target points; - Change Cartesian space motion command to joint space motion command
-50103	The end point of Cartesian path cannot match the given ConfData	- Call setDefaultConfOpt (false) to cancel confdata - Change to MoveJ or MoveAbsJ - Change target point confdata
-50104	Joint torque over limit	- Check if the load value matches the actual load - Check the friction coefficient, motor overload coefficient, transmission overload coefficient and other parameters of the robot - Try to change the command format, such as replacing Cartesian space command with joint space command
-50112	The posture is incorrect, resulting in inverse kinematics calculation failure	- Teach posture again - If the current model is 3-axis or 4-axis robot, please check whether the entered posture matches the features of the current model - If setAvoidSingularity (lock4axis) is used, please check if the target meets the lock axis requirements
-50113	The changed orientation exceeds the set threshold	- Please avoid singularity - Reset the threshold for singular avoidance pose change - Teach points again and replace target points - Avoid singularity in other ways
-50114	The lookahead central axis movement exceeds the motion range, the movement stops in advance	- Cancel the soft limit - Manually move the robot's axes to the normal working range
-50115	During the lookahead of the trajectory, encounter singularities	- Please avoid singularity - Teach points again and replace target points - Replace the Jog method and try the joint space Jog - Change Cartesian space motion command to joint space motion command
-50118	The endpoint solved after changing the pose is inconsistent with the required endpoint	- Please avoid singularity - Teach points again and replace target points - Replace the singularity avoidance method
-50120	Search path endpoint angle failed under singularity avoidance	- Please avoid singularity - Teach points again and replace target points - Replace the singularity avoidance method - If Conf is turned on, it can be turned off
-50121	Search path endpoint angle failed under singularity avoidance	- Please avoid singularity - Teach points again and replace target points - Replace the singularity avoidance method
-50204	The sampling points are insufficient in the planning process, please rerun	The IPC status is unstable. Please try to run it again or restart it
-50205	The difference between adjacent position commands is too large	- Teach the trajectory target point position again - Try to change the command format, such as changing MoveL to MoveJ
-50208	Internal trajectory error	- Please call moveReset() to reset the motion command cache - Increase or decrease the turning area
-50401	Collision detected	- Check the robot operating environment, confirm that the staff and devices are safe, and power on the robot, before running again - Check if the current tool settings are consistent with reality, and if the set tool mass and center of mass are reasonable
-50501	The setting of the tool work object frame fails, possibly due to the absence of the tool or the work object	Set the created tool work object, which is handheld and external respectively

-50512	Number of joints mismatches	The subscripted parameter of the joint space Jog cannot exceed the number of joints and the number of external joints
-50513	The speed setting is invalid, causing the robot not to move	Input speed parameters ranging from 0.01 to 1
-50514	The step length setting is invalid, causing the robot not to move	Input step length parameters greater than 0
-50515	The reference frame setting is invalid, causing the robot not to move	Input supported frame type parameters
-50516	The axis of motion setting is invalid, causing the robot not to move	Input index parameter according to the instructions
-50518	Motions failed. The current point may be the target point position	The current point position may be the target point position
-50519	Failed to generate trajectory, the target point may exceed the robot's working range	Teach point positions again within the normal working range of robots
-50525	This model does not support motion in the singularity avoidance mode, or the robot fails to lock the axis. The motion will not run if the 4-axis angle is not 0. Please adjust the 4-axis angle	Adjust the 4-axis angle to 0° or ±180°
-60005	RL project or designated task does not exist	Run the created project and task
-60014	The controller in the current state does not allow motion	Make sure the robot is powered on and is not running
-60200	Load information error, setting failed	- Load information error, please check if the load weight exceeds the rated load. The center of mass does not exceed 0.3 m - Force control does not support external tools or handheld work object
-60511	The path does not exist	Use the saved playback path
-60611	Diagnostic data is being generated, so the motion is disabled. Please wait	Wait for 10s and try to start again
-60702	The current 4-axis angle is not 0, it is not allowed to switch to 4-axis fixed mode, or the model does not support it	Please check if the current angle of the 4-axis is 0° or 180°
-60704	The enabling conditions for sacrificing pose singularity avoidance are not met	Please check whether the current status meets the enabling conditions for sacrificing pose singularity avoidance
-60706	The enabling conditions for joint space interpolation singularity avoidance are not met	Please check whether the current status meets the enabling conditions for joint space interpolation singularity avoidance
256	Network connection error	- The IPC is not turned on - The robot's address is incorrect, or it is not on the same local area network - The instance type is incorrect or SDK is not authorized, leading to connection rejection
255	The connection to the robot has not been initialized	This generally does not occur. Just create an instance of the Robot class normally
257	The robot message cannot be parsed, possibly due to a mismatch between the SDK version and the controller version	Possible incompatibility between controller and SDK version
258	Parameter error: The value exceeds the range	Check the parameter value range according to the function description
259	Parameter error: The type or number of parameters is incorrect	Check the parameter type or array length according to the function description
260	It is not a valid transformation matrix	Check if the input parameters meet the requirements of the secondary transformation matrix
261	The number of array elements does not match the number of robot axes	Input an array length that matches the number of robot axes
262	The motion control mode is incorrect, please switch to the correct mode	Call setMotionControlMode to switch mode according to the actual situation
263	No response is received from the robot before timeout, possibly due to network communication issues	Check the network connection status or provide feedback to technical personnel
264	Redo	Collision detector has been turned on, it should be turned off first
265	Failed to receive data through UDP port, please check network and firewall configuration	- Check if the local address is set correctly - Check firewall settings to see if UDP connections are allowed
266	Client verification failed, please check the controller version, authorization status, and robot model	- Upgrade the controller to a matching version according to the manual or README - Create matching robot instances - Contact technical personnel to authorize SDK functionality
272	Event is not monitored	Call setEventWatcher first to start monitoring data

273	The point positions are too close, so the frame is not calibrated successfully	Refer to the tool work object calibration method in the *xCore Control System User Manual* for recalibration
512	IP address or port is set incorrectly, or the port is occupied	- The local IP address cannot be localhost or a robot address - There is a port occupancy situation, please check RCI settings - port
513	Unsupported fields are set, or the total length exceeds the limit	The supported fields can be found in RtSupportedFields, with a total length limit of 1024 bytes
768	There are no executable motion commands	First call moveAppend to issue motion commands, then start the motion
769	The robot is stopped or cannot pause in its current state	Perhaps due to power-off in manual mode or emergency stop just before calling stop(), the controller is responding to the stopping process. Please wait

4.10.2 Exceptions

In real-time mode, interfaces for sending motion commands, periodic scheduling, reading status data, etc., may throw exceptions during calling. The exception types are defined in rokae/exception.h.

Exception information	Cause and solutions
The motion control mode is incorrectly, it is non-real-time mode	- Call setMotionControlMode (RtCommand) to switch mode - The previous impedance control error causes the controller to switch back to non-real-time mode, it should be enabled
Failed to create real-time control client	Network connection error, check the running status of IPC
Started receiving data	Stop receiving status data first, and then restart receiving
The controller cannot set the status data to be sent	Maybe the version does not match, please provide feedback to technical personnel
Failed to set control command	Maybe the version does not match, please provide feedback to technical personnel
Failed to open UDP socket	Port is occupied or failed to bind local address
The controller is unable to send robot status information.	Controller UDP port is set incorrectly, please report to technical personnel to check controller logs
Not started receiving data yet	Call startReceptRobotState to start receiving data
No robot status feedback is received before timeout, please check network configuration	- Check if the local address is set correctly - Check firewall settings to see if UDP connections are allowed
Error in receiving robot status data	State data parsing failed, possibly due to version mismatch. Please provide feedback to technical personnel
Unable to receive feedback on motion errors	Data parsing failed, possibly due to version mismatch. Please provide feedback to technical personnel
Minimum version requirement for xCore	Follow the prompts to upgrade the controller version
SDK functionality is not authorized	Contact technical personnel to authorize SDK functionality
The robot instance type does not match the connected model	Create corresponding robot instances according to the manual
Local IPv4 address is not set successfully, it is not 127.0.0.1 or robot address	Set the correct local address
6 or 16 parameters are received at posture initialization	To initialize trans&rpy, an array of length 6 should be input; to initialize pos (real-time Cartesian command), an array of length 16 should be input
Model loading failed	Unless there is a network connection issue, it generally does not occur
Models not currently supported by the xMate model library	Please refer to the xMateModel class annotations to view the supported models
Motion has already started, please do not call again	Only one motion callback is allowed to run at a time. Start the next motion after calling stopMove
Start motion before calling	First call startMove, then startLoop
Setting mode error, failed to start motion	Unless there is a network connection issue, it generally does not occur
Failed to start motion	- Ensure that the robot is idle and not in an emergency stop or other abnormal use state - This issue mostly occurs in impedance control. Please calibrate the zero point and torque sensor, set the load correctly, and try again - Or provide complete exception information feedback to technical personnel.
Motion stopping failed	Usually it's a network connection issue
Industrial models only support position control	Industrial models use position control
The elbow formats for the starting and ending points are different	The Cartesian command hasElbow has inconsistent values
The command is inconsistent with control mode	The data type returned by the callback function (JointPosition/CartesianPosition/Torque) should match the control type
Command sending failed	Command data serialization failed, please report to technical

	personnel
The starting/ending/auxiliary points of the arc coincide	Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
When solving the arc path, three points are collinear	Adjust the starting point, auxiliary point, and ending point of the arc
When solving the arc path, the angle between the three points is too small	Adjust the starting point, auxiliary point, and ending point of the arc
When solving the arc path, the arc radius is too small	Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
Not initialized	The default construction of FollowePosition requires calling the init() function before use
Filtering error, as input data is not positive	Check the command data to see whether the callback function returns an undefine command
Filter error: Filtered data is infinite	Check the command data to see whether the callback function returns an undefine command

5 Notes and troubleshooting

5.1 Use with RobotAssist

Currently, the use of the xCore SDK will not limit the control through RobotAssist. Some robot status changes made by the xCore SDK will be displayed on the RobotAssist interface, while some project running and motion control operations are completed separately. Here is a summary:

Components updated synchronously	● Bottom status bar: Automatic/manual, robot status, power on/off ● Status monitoring window; ● Log reporting;
Controlled by both SDK and RobotAssist, i.e. the components modified through RobotAssist will take effect	● RL project running rate and loop/single cycle; ● Pause motion in non-real-time mode; ● Tool work object group: Change the selected tool work object in the upper right corner of RobotAssist will change the toolset. By using setToolset() to set the tool work object, "toolx" and "wobjx" will display in the upper right corner
Components not updated synchronously, including but not limited to	● The issued motion command cannot enable the robot to execute by clicking the start button; ● Loaded RL projects, lookahead pointers, and motion pointers, will not be displayed synchronously; ● Most of the enabled functions and set values displayed on the robot setting interface;

It is recommended to use a single control source to avoid any confusion.

5.2 Compatibility with RCI client

Users of the original RCI client can directly use xCoreSDK after upgrading the controller to v3.1.1 (v1.7 and later versions are also available).

5.2.1 First use

1. Make sure that RobotAssist - Communication - RCI setting is disabled. It can also be confirmed via the robot status in the middle of the status bar;
2. Enable RCI by calling setMotionControlMode (RtCommand). If it is successful, the robot status will change to RCI, which will also be displayed in the status bar:

3. After that, RCI can be enabled/disabled using the above API or through RobotAssist. It should be noted that if the configuration has been erased, it will be considered as the first use, and RCI needs to be enabled using the API in the SDK.

5.2.2 Switch to RCI client

After using the SDK, the RCI client cannot be used simultaneously. To switch back to the RCI client, call useRciClient(true) after ensuring that RCI is disabled as instructed. Then RCI can be enabled/disabled through RobotAssist.

5.3 Real-time commands

The motion command issued by the user should meet the suggested and necessary conditions. The recommended conditions are for smoother robot motion and optimal performance. The required conditions must be met, otherwise, the robot will stop.

5.3.1 Joint space motion

1. Required conditions

The joint space trajectory is smooth. At least the speed is continuously differentiable:

$$q_{min} < q_c < q_{max}$$
$$-\dot{q}_{max} < \dot{q}_c < \dot{q}_{max}$$
$$-\ddot{q}_{max} < \ddot{q}_c < \ddot{q}_{max}$$
$$-\dddot{q}_{max} < \dddot{q}_c < \dddot{q}_{max}$$

2. Recommended conditions

Torque command is within limit:

$$-\tau_{jmax} < \tau_{jc} < \tau_{jmax}$$
$$-\dot{\tau}_{jmax} < \dot{\tau}_{jc} < \dot{\tau}_{jmax}$$

When the motion starts:

$$q = q_c$$
$$\dot{q}_c = 0$$
$$\ddot{q}_c = 0$$

When the motion finishes:

$$\dot{q}_c = 0$$
$$\ddot{q}_c = 0$$

5.3.2 Cartesian space motion

1. Required conditions

Not in a singular position and within the working space:

$$-\dot{p}_{max} < \dot{p}_c < \dot{p}_{max}$$
$$-\ddot{p}_{max} < \ddot{p}_c < \ddot{p}_{max}$$
$$-\dddot{p}_{max} < \dddot{p}_c < \dddot{p}_{max}$$

2. Recommended conditions

Torque command is within limit:

$$-\tau_{jmax} < \tau_{jc} < \tau_{jmax}$$
$$-\dot{\tau}_{jmax} < \dot{\tau}_{jc} < \dot{\tau}_{jmax}$$

When the motion starts:

$$T = T_c$$
$$\dot{p}_c = 0$$
$$\ddot{p}_c = 0$$

When the motion finishes:

$$\dot{p}_c = 0$$
$$\ddot{p}_c = 0$$

5.3.3 Direct torque control

1. Required conditions

Torque command is within limit and continuous:

$$-\dot{\tau}_{jmax} < \dot{\tau}_{jc} < \dot{\tau}_{jmax}$$
$$-\tau_{jmax} < \tau_{jc} < \tau_{jmax}$$

5.3.4 Motion limits

Cartesian space motion limits of xMateER3 Pro, xMateER7 Pro, xMateER3, and xMateER7:

Parameter	Translation	Rotation
Velocity	1.0 m/s	2.5 rad/s
Acceleration	10.0 m/s^2	10.0 rad/s^2
Jerk	5000.0 m/s^3	5000.0 rad/s^3

Joint space motion limits of xMateER3 Pro and xMateER7 Pro:

Parameter	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6	Axis 7	Unit
Limit upper	170	120	170	120	170	120	360	°
Limit lower	-170	-120	-170	-120	-170	-120	-360	°
Velocity upper limit	2.175	2.175	2.175	2.175	2.610	2.610	2.610	rad/s
Acceleration upper limit	15	7.5	10	10	15	15	20	rad/s^2
Jerk upper limit	5000	3500	5000	5000	7500	7500	7500	rad/s^3

Joint space motion limits of xMate3 and xMate7:

Parameter	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6	Unit
Limit upper	170	120	120	170	120	360	°
Limit lower	-170	-120	-120	-170	-120	-360	°
Velocity upper limit	2.175	2.175	2.175	2.610	2.610	2.610	rad/s
Acceleration upper limit	15	7.5	10	15	15	20	rad/s^2
Jerk upper limit	5000	3500	5000	7500	7500	7500	rad/s^3

Direct torque control limits of xMateER3 Pro and xMateER7 Pro:

Parameter	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6	Axis 7	Unit
Torque upper	85	85	85	85	36	36	36	Nm

limit								
Torque derivative upper limit	1500	1500	1500	1500	1000	1000	1000	Nm/s

Direct torque control limits of xMate3 and xMate7:

Parameter	**Axis 1**	**Axis 2**	**Axis 3**	**Axis 4**	**Axis 5**	**Axis 6**	**Unit**
Torque upper limit	85	85	85	36	36	36	Nm
Torque derivative upper limit	1500	1500	1500	1000	1000	1000	Nm/s

5.3.5 DH parameters

xMateER3 Pro DH parameters:

Joint	A(mm)	Alpha(rad)	D(mm)	Theta(rad)
1	0	-pi/2	341.5	0
2	0	pi/2	0	0
3	0	-pi/2	394.0	0
4	0	pi/2	0	0
5	0	-pi/2	366	0
6	0	pi/2	0	0
7	0	0	250.3	0

xMateER7 Pro DH parameters:

Joint	A(mm)	Alpha(rad)	D(mm)	Theta(rad)
1	0	-pi/2	404.0	0
2	0	pi/2	0	0
3	0	-pi/2	437.5	0
4	0	pi/2	0	0
5	0	-pi/2	412.5	0
6	0	pi/2	0	0
7	0	0	275.5	0

xMate3 DH parameters:

Joint	A(mm)	Alpha(rad)	D(mm)	Theta(rad)
1	0	-pi/2	341.5	0
2	394	0	0	0
3	0	pi/2	0	0
4	0	-pi/2	366	0
5	0	pi/2	0	0
6	0	0	250.3	0

xMate7 DH parameters:

Joint	A(mm)	Alpha(rad)	D(mm)	Theta(rad)
1	0	-pi/2	404	0
2	437.5	0	0	0
3	0	pi/2	0	0
4	0	-pi/2	412.5	0
5	0	pi/2	0	0
6	0	0	275.5	0

5.4 Troubleshooting

5.4.1 Network connection issues

Real-time motion command and status information are sent via UDP protocol unicast. When a real-time status error "no robot status data is received before timeout" occurs, check the firewall settings of Windows system to see if UDP inbound rules are enabled.

Linux (Debian distribution): Open a terminal, enter the command "nc –vul –p 1337", then run any sample program controlled in real-time mode, and check if a port from the robot's address is connected and the data is received.

5.4.2 Problem of load dropping of direct torque control in real-time mode

Force control is related to the hardware state of the robot itself. If no command is issued after enabling the direct torque control, but the robot falls or sways, it may be due to the inaccurate force control model. First, check whether the force control parameters are accurate. Secondly, adjust the moving command according to the actual situation.

6 Usage Examples

This section includes some C++ sample programs. More examples are shown in the software package.

6.1 Non-real-time APIs

6.1.1 Example I: Basic operation of robot, information query, jog, drag, etc.

```
/**
 * @brief Example - Basic information query, calculating forward/inverse kinematics results
 */
template <WorkType wt, unsigned short dof>
void example_basicOperation(Robot_T<wt, dof> *robot){
  error_code ec;
  // *** Query information ***
  auto robotinfo = robot->robotInfo(ec);
  print(os, "Controller version number:", robotinfo.version, "type:", robotinfo.type);
  print(os, "xCore-SDK version:", robot->sdkVersion());

  // *** Obtain robot information such as the current posture, axis angle, and base frame ***
  auto joint_pos = robot->jointPos(ec); // Axis angle [rad]
  auto joint_vel = robot->jointVel(ec); // Axis speed [rad/s]
  auto joint_torque = robot->jointTorque(ec); // Axis torque [N.m]
  auto tcp_xyzabc = robot->posture(CoordinateType::endInRef, ec);
  auto flan_cart = robot->cartPosture(CoordinateType::flangeInBase, ec);
  robot->baseFrame(ec); // Base frame
  print(os, "End-effector posture in the external reference frame", tcp_xyzabc);
  print(os, "Frame of flange in base -", flan_cart);

  // *** Calculate forward and inverse kinematics results ***
  auto model = robot->model();
  auto ik = model.calcIk(tcp_xyzabc, ec);
  auto fk_ret = model.calcFk(ik, ec);
}

/**
 * @brief Example - Enabling/Disabling the drag mode
 */
void example_drag(BaseCobot *robot) {
  error_code ec;
  robot->setOperateMode(rokae::OperateMode::manual, ec);
  robot->setPowerState(false, ec); // Before enabling the drag mode, the manipulator shall stay in the manual power-off state
  // In Cartesian space, conduct free drag
  robot->enableDrag(DragParameter::cartesianSpace, DragParameter::freely, ec);
  print(os, "Enable the drag mode", ec, "press enter to continue");
  std::this_thread::sleep_for(std::chrono::seconds(2)); //Wait to switch the control mode
  while(getchar() != '\n');
  robot->disableDrag(ec);
  std::this_thread::sleep_for(std::chrono::seconds(2)); //Wait to switch the control mode
}

/**
 * @brief Example - IO read-write, register
 */
void example_io_register(BaseRobot *robot) {
  error_code ec;
  print(os, "DO1_0 current signal value:", robot->getDO(1,0,ec));
  robot->setSimulationMode(true, ec); // DI can only be set when the input simulation mode is enabled
  robot->setDI(0, 2, true, ec);
  print(os, "DI0_2 current signal value:", robot->getDI(0, 2, ec));
  robot->setSimulationMode(false, ec); // Turn off the simulation mode

  // Read a single register, with the type of float
  // Assume "register0" is a register array with a length of 10
  float val_f;
  std::vector<float> val_af;
  // Read the 1st element, i.e., register0[1] in state monitoring, and assign the result to val_f
  robot->readRegister("register0", 0, val_f, ec);
  // Read the 10th element, i.e., register0[10] in state monitoring, and assign the result to val_f
  robot->readRegister("register0", 9, val_f, ec);
  // Read the entire array and assign it to val_af. The length of val_af also becomes 10. In this case, the index parameter is irrelevant.
  robot->readRegister("register0", 9, val_af, ec);

  // Read the int-type register array
  std::vector<int> val_ai;
  robot->readRegister("register1", 1, val_ai, ec);
```

```
  // Write bool/bit-type registers
  robot->writeRegister("register2", 0, true, ec);
}

/**
 * @brief Example - Jogging the robot
 * @param robot
 */
void example_jog(BaseRobot *robot) {
  error_code ec;
  robot->setMotionControlMode(rokae::MotionControlMode::NrtCommand, ec);
  robot->setOperateMode(rokae::OperateMode::manual, ec); // Jog under the manual mode
  print(os, "To prepare to jog the robot, manual mode power-on is required. Please confirm the robot is powered on, and then press enter to
continue");
  // For those with an external enable switch, it is required to hold the switch to manually power on
  robot->setPowerState(true, ec);

  print(os, "-- Start jogging the robot -- \n Move 50 mm along the Z+ direction in the world frame at 50% speed. Press enter after the robot
has stopped moving to continue");
  robot->startJog(JogOpt::world, 0.5, 50, 2, true, ec);
  while(getchar() != '\n');
  print(os, "In joint space, Axis 6 perform continuous rotation in the negative direction at 5% speed. Press enter to stop jogging");
  robot->startJog(JogOpt::jointSpace, 0.05, 5000, 5, false, ec);
  while(getchar() != '\n'); // Press enter to stop
  robot->stop(ec); // Call stop() to stop after jog completed
}

/**
 * @brief Example - Enable/Disable collision detection
 */
template <unsigned short dof>
void example_setCollisionDetection(Cobot<dof> *robot) {
  error_code ec;
  // Set the sensitivity for each axis, with the range of 0.01–2.0, equivalent to 1%–200% set in RobotAssist
  // Trigger behavior: safe stop; fallback distance 0.01 m
  robot->enableCollisionDetection({1.0, 1.0, 0.01, 2.0, 1.0, 1.0, 1.0}, StopLevel::stop1, 0.01, ec);
  std::this_thread::sleep_for(std::chrono::seconds(2));
  // Disable collision detection
  robot->disableCollisionDetection(ec);
}
```

6.1.2 Example II: Non-real-time motion command

```
/**
 * @brief Example - Setting axis configuration data (confData) to handle problems with multiple inverse kinematics results; applicable
model for point position: xMateCR7
 */
void multiplePosture(xMateRobot &robot) {
  error_code ec;
  std::string id;

  // This example uses the default tools and work objects
  Toolset defaultToolset;
  robot.setToolset(defaultToolset, ec);

  MoveJCommand moveJ({0.2434, -0.314, 0.591, 1.5456, 0.314, 2.173});
  // For the same end-effector posture, different confData leads to different axis angles
  moveJ.target.confData = {-67, 100, 88, -79, 90, -120, 0, 0};
  robot.moveAppend({moveJ}, id, ec);
  moveJ.target.confData = {-76, 8, -133, -106, 103, 108, 0, 0};
  robot.moveAppend({moveJ}, id, ec);
  moveJ.target.confData = {-70, 8, -88, 90, -105, -25, 0, 0};
  robot.moveAppend({moveJ}, id, ec);
  robot.moveStart(ec);
  waitForFinish(robot, id, 0);
}
```

```cpp
/**
 * @brief Example - Seven-axis redundant motion & motion restoration after collision detection; applicable model for point position: xMateER3 Pro
 */
void redundantMove(xMateErProRobot &robot) {
  error_code ec;
  std::string id;

  // This example uses the default tool and work object, with a speed of v500 and turning zone fine
  Toolset defaultToolset;
  robot.setToolset(defaultToolset, ec);
  robot.setDefaultSpeed(500, ec);
  robot.setDefaultZone(0, ec);

  // Optional: Set a callback function for collision detection events
  robot.setEventWatcher(Event::safety, [&](const EventInfo &info){
    recoverFromCollision(robot, info);
  }, ec);


  MoveAbsJCommand moveAbsj({0, M_PI/6, 0, M_PI/3, 0, M_PI_2, 0});
  // ** 1) Variable elbow motion **
  MoveLCommand moveL1({0.562, 0, 0.432, M_PI, 0, -M_PI});
  moveL1.target.elbow = 1.45;
  robot.moveAppend({moveAbsj}, id, ec);
  robot.moveAppend({moveL1}, id, ec);
  moveL1.target.elbow = -1.51;
  robot.moveAppend({moveL1}, id, ec);
  robot.moveStart(ec);
  // The last moveAppend() sends one command, so index = 0
  waitForFinish(robot, id, 0);

  // ** 2) 60° elbow arc **
  CartesianPosition circle_p1({0.472, 0, 0.342, M_PI, 0, -M_PI}),
  circle_p2({0.602, 0, 0.342, M_PI, 0, -M_PI}),
  circle_a1({0.537, 0.065, 0.342, M_PI, 0, -M_PI}),
  circle_a2({0.537, -0.065, 0.342, M_PI, 0, -M_PI});
  // All elbows are set to 60°
  circle_p1.elbow = M_PI/3;
  circle_p2.elbow = M_PI/3;
  circle_a1.elbow = M_PI/3;
  circle_a2.elbow = M_PI/3;

  MoveLCommand moveL2(circle_p1);
  robot.moveAppend({moveL2}, id, ec);
  MoveCCommand moveC1(circle_p2, circle_a1), moveC2(circle_p1, circle_a2);
  std::vector<MoveCCommand> movec_cmds = {moveC1, moveC2};
  robot.moveAppend(movec_cmds, id, ec);
  robot.moveStart(ec);
  // The last moveAppend() sends 2 commands, so it is necessary to wait for the second point to finish and return; index refers to the subscript index of the second point
  waitForFinish(robot, id, (int)movec_cmds.size() - 1);
}

/**
* @brief Example - Motion with rail; applicable model for point position: xMateSR4
 */
template <WorkType wt, unsigned short dof>
void moveWithRail(Robot_T<wt, dof> *robot) {
  error_code ec;
  bool is_rail_enabled;
  robot->getRailParameter("enable", is_rail_enabled, ec);
  if(!is_rail_enabled) {
    print(os, "Disabled rail");
    return;
  }

  // Enable/Disable the rail and set rail parameters
  // Setting rail parameters and base frame requires controller restart to take effect. This section only demonstrates how to call the APIs
  robot->setRailParameter("enable", true, ec); // Enable rail functions
  robot->setRailParameter("maxSpeed", 1, ec); // Set the maximum speed to 1 m/s
  robot->setRailParameter("softLimit", std::vector<double>({-0.8, 0.8}), ec); // Set the soft limit to ±0.8 m
  robot->setRailParameter("reductionRatio", 1.0, ec); // Set the reduction ratio
```

```
  auto curr = robot->BaseRobot::jointPos(ec);
  print(os, "Current axis angle", robot->BaseRobot::jointPos(ec));

  // *** Jog rail example ***
  // Power on in manual mode
  robot->setOperateMode(OperateMode::manual, ec);
  robot->setPowerState(true, ec);
  std::vector<double> soft_limit;
  robot->getRailParameter("softLimit", soft_limit, ec);
  // Jog within soft limits
  double step = (curr.back() - soft_limit[0] > 0.1 ? 0.1 : (curr.back() - soft_limit[0])) * 1000.0;
  // Taking jogging in joint space of a six-axis model as an example, the index 0–5 represent axes 1–6, and index=6 represents the first
external axis
  int ex_jnt_index = robot->robotInfo(ec).joint_num;
  // The rail performs the negative direction motion of 100 mm in joint space
  robot->startJog(JogOpt::jointSpace, 0.6, step, ex_jnt_index, false, ec);
  // Wait for jogging to finish
  while(true) {
    std::this_thread::sleep_for(std::chrono::milliseconds(50));
    if(robot->operationState(ec) != OperationState::jogging) break;
  }
  robot->stop(ec);

  // *** Example of motion commands with rail ***
  CartesianPosition p0({0.56, 0.136, 0.416, M_PI, 0, M_PI}), p1({0.56, 0.136, 0.3, M_PI, 0, M_PI});
  p0.external = { 0.02 }; // The rail moves to 0.02 m, similar for others below
  p1.external = { -0.04 };
  MoveAbsJCommand abs_j_command({0, M_PI/6, -M_PI_2,0, -M_PI/3, 0 });
  abs_j_command.target.external = { 0.1 }; // The rail moves to 0.1m
  MoveJCommand j_command(p0);
  MoveLCommand l_command(p1);
  MoveCCommand c_command(p1, p0);

  // Add custom information, which will be included in the motion feedback
  l_command.customInfo = "hello";

  std::string id;
  robot->moveAppend(abs_j_command, id, ec);
  robot->moveAppend(j_command, id, ec);
  robot->moveAppend(l_command, id, ec);
  robot->moveAppend(abs_j_command, id, ec);
  robot->moveAppend(c_command, id, ec);
  robot->moveStart(ec);
  waitForFinish(*robot, id, 0);
}
```

6.2 Real-time motion control

6.2.1 Example I: Cartesian impedance control

```
int main() {
  using namespace std;
  try {
    std::string ip = "192.168.0.160";
    std::error_code ec;
    rokae::xMateErProRobot robot(ip, "192.168.0.100"); // Local IP address: 192.168.0.100

    robot.setOperateMode(rokae::OperateMode::automatic,ec);
    robot.setMotionControlMode(MotionControlMode::RtCommand, ec);
    robot.setPowerState(true, ec);

    auto rtCon = robot.getRtMotionController().lock();

    // Set the data to be received. jointPos_m is used in this example
    rtCon->startReceiveRobotState({RtSupportedFields::jointPos_m, RtSupportedFields::tauVel_c,
                                   RtSupportedFields::tcpPose_m, RtSupportedFields::elbow_m});

    // Read real-time state data: torque differential
    rtCon->updateRobotState();
    std::array<double,7> array7 {};
    rtCon->getStateData(RtSupportedFields::tauVel_c, array7);

    std::array<double, 16> init_position {};
    static bool init = true;
    double time = 0;
```

```
    rtCon->getStateData(RtSupportedFields::jointPos_m, array7);
    std::array<double,7> q_drag_xm7p = {0, M_PI/6, 0, M_PI/3, 0, M_PI/2, 0};

    // Get the current axis angles of the robot (needing to set "jointPos_m" as the state data to be received)
    // Move from the current position to the drag posture using MoveJ
    rtCon->MoveJ(0.5, array7, q_drag_xm7p);

    // Set Cartesian space impedance factors and start Cartesian space impedance motion
    rtCon->setCartesianImpedance({1000, 1000, 1000, 100, 100, 100}, ec);
    rtCon->startMove(RtControllerMode::cartesianImpedance);
    std::atomic<bool> stopManually { true };

    std::function<CartesianPosition()> callback = [&, rtCon] {
        time += 0.001;
        if(init) {
          rtCon->getStateData(RtSupportedFields::tcpPose_m, init_position);
          init = false;
        }
        constexpr double kRadius = 0.2;
        double angle = M_PI / 4 * (1 - std::cos(M_PI / 2 * time));
        double delta_z = kRadius * (std::cos(angle) - 1);

        CartesianPosition output{};
        output.pos = init_position;
        output.pos[7] += delta_z;

        if(time > 20) {
          output.setFinished(); // End after 20s
          stopManually.store(false);  // The loop is non-blocking and stops the state synchronously with the main thread
        }
        return output;
    };

    rtCon->setControlLoop(callback);
    // Non-blocking loop
    rtCon->startLoop(false);
    while(stopManually.load());
      } catch (const std::exception &e) {
            std::cout << e.what();
        }
        return 0;
      }
```

6.2.2 Example II: Command combination for host computer trajectory planning

```
int main() {
  using namespace std;
  try {
    std::string ip = "192.168.0.160";
    std::error_code ec;
    rokae::xMateErProRobot robot(ip, "192.168.0.180"); // ****   XMate 7-axis
    robot.setOperateMode(rokae::OperateMode::automatic,ec);
    robot.setRtNetworkTolerance(20, ec);
    robot.setMotionControlMode(MotionControlMode::RtCommand, ec);
    robot.setPowerState(true, ec);

    auto rtCon = robot.getRtMotionController().lock();
    print(os, "Start receive");
    robot.startReceiveRobotState(std::chrono::milliseconds(1), {RtSupportedFields::jointPos_m, RtSupportedFields::elbow_m,
RtSupportedFields::tcpPose_m});

    // Robot model for example program: xMateER7 Pro
    // 1. Move from the current position to the drag position using MoveJ
    std::array<double,7> q_drag_xm7p = {0, M_PI/6, 0, M_PI/3, 0, M_PI/2, 0};
    robot.updateRobotState(std::chrono::milliseconds(1));
    rtCon->MoveJ(0.4,  robot.jointPos(ec), q_drag_xm7p);

    // 2. Circular motion (in the X-Y plane)
    CartesianPosition start, aux, target;
    robot.updateRobotState(std::chrono::milliseconds(1));
    Utils::postureToTransArray(robot.posture(rokae::CoordinateType::endInRef, ec), start.pos);
    Eigen::Matrix3d rot_start;
    Eigen::Vector3d trans_start, trans_aux, trans_end;
    Utils::arrayToTransMatrix(start.pos, rot_start, trans_start);
```

```
    trans_end = trans_start; trans_aux = trans_start;
    trans_aux[0] -= 0.28;
    trans_aux[1] -= 0.05;
    trans_end[1] -= 0.15;

    Utils::transMatrixToArray(rot_start, trans_aux, aux.pos);
    Utils::transMatrixToArray(rot_start, trans_end, target.pos);
    rtCon->MoveC(0.2, start, aux, target);

    // 3. Linear motion
    Utils::postureToTransArray(robot.posture(rokae::CoordinateType::endInRef, ec), start.pos);
    Utils::arrayToTransMatrix(start.pos, rot_start, trans_start);

    trans_end = trans_start;
    // Move along x -0.1m, y -0.3m, and z -0.25m
    trans_end[0] -= 0.1;
    trans_end[1] -= 0.3;
    trans_end[2] -= 0.25;
    Utils::transMatrixToArray(rot_start, trans_end, target.pos);

    print(os, "MoveL start position:", start.pos, "Target:", target.pos);
    rtCon->MoveL(0.3, start, target);

    // 4. Exit the real-time mode
    robot.setMotionControlMode(rokae::MotionControlMode::NrtCommand, ec);
    robot.setOperateMode(rokae::OperateMode::manual, ec);

  } catch (const std::exception &e) {
    std::cout << e.what();
  }
  return 0;
}
```

7 Feedback and Corrections

Feel free to let us know if there are inaccurate descriptions or mistakes in the manual. If you encounter any issues or have suggestions while reading this, please send an email to xuqiu@rokae.com. We will try to respond to each message as soon as possible.

8 Appendix A - C++ API

8.1 Enumeration type

8.1.1 Robot state rokae::OperationState

idle	Stationary
jog	Jog state (not moving)
rtControlling	Under real-time control
drag	Drag mode enabled
rlProgram	Running RL project
demo	Demo in progress
dynamicIdentify	Identifying dynamic parameters
frictionIdentify	Identifying friction force
loadIdentify	Identifying load
moving	In robot motion
jogging	Jog is in motion
unknown	Unknown

8.1.2 Model type rokae::WorkType

industrial	Industrial robot
collaborative	Collaborative robot

8.1.3 Robot operate mode rokae::OperateMode

manual	Manual
automatic	Automatic
unknown	Unknown (exception)

8.1.4 Robot power on/off and emergency stop state rokae::PowerState

on	Motor on
off	Motor off
estop	Emergency stop button pressed
gstop	Safety gate opened
unknown	Unknown (exception)

8.1.5 Posture frame type rokae:: CoordinateType

flangeInBase	Frame of flange in base
endInRef	End-effector in the external reference frame

8.1.6 Motion control mode rokae::MotionControlMode

Idle	Idle
NrtCommand	Motion command execution in the non-real-time mode
NrtRLTask	Running RL project in the non-real-time mode
RtCommand	Real-time control

8.1.7 Controller real-time control mode rokae::RtControllerMode

jointPosition	Real-time joint space position control
cartesianPosition	Real-time Cartesian space position control
jointImpedance	Real-time joint space impedance control
cartesianImpedance	Real-time Cartesian space impedance control
torque	Real-time toque control

8.1.8 Robot stop level rokae::StopLevel

stop0	Quickly stop robot motion and power off
stop1	Schedule the robot to stop on the original path and power off
stop2	Schedule the robot to stop on the original path and remain powered-on
suppleStop	Soft stop, only applicable to collaborative models

8.1.9 Robot drag mode parameters rokae::DragParameter

Space::jointSpace	Joint space drag
Space::cartesianSpace	Cartesian space drag
Type::translationOnly	Translational drag only
Type::rotationOnly	Rotational drag only
Type::freely	Free drag

8.1.10 Frame type rokae::FrameType

world	World frame

base	Base frame
flange	Flange frame
tool	Tool frame
wobj	Work object frame
path	Path frame — the necessary process of the frame for force control tasks tracking the trajectory variation
rail	Rail base frame

8.1.11 Jog options – frame rokae::JogOpt::Space

world	World frame
flange	Flange frame
baseFrame	Base frame
toolFrame	Tool frame
wobjFrame	Work object frame
jointSpace	Joint space
singularityAvoidMode	Singularity avoidance method, applicable to the industrial six-axis, xMateCR, and xMateSR six-axis models
baseParallelMode	Parallel base mode, applicable to the industrial six-axis, xMateCR, and xMateSR six-axis models

8.1.12 Singularity avoidance method rokae::AvoidSingularityMethod

lockAxis4	Four-axis locking
wrist	Sacrifice pose
jointWay	Short trajectory interpolation in joint space

8.1.13 MoveCF full-circle pose rotation type rokae::MoveCFCommand::RotType

constPose	Constant pose
rotAxis	Rotating axis rotation
fixedAxis	Fixed axis rotation

8.1.14 xPanel configuration: external power supply mode rokae::xPanelOpt::Vout

off	No output
reserve	Reserved
supply12v	Output 12 V
supply24v	Output 24 V

8.1.15 Torque type rokae::TorqueType

full	Joint torque calculated using dynamic models
inertia	Inertia force
coriolis	Coriolis force
friction	Friction force
gravity	Gravity

8.1.16 Event type rokae::Event

moveExecution	Non-real-time motion command execution information
safety	Safety (collision or not)
rlExecution	RL execution state
logReporter	Controller log reporting

8.2 Data structure

8.2.1 Basic robot information rokae::Info

std::string id	Robot uid used to distinguish connected robots
std::string mac;	Mac address
std::string version	Controller version
std::string type	Robot model
int joint_num	Number of axes

8.2.2 Robot state list rokae::StateList

std::vector<double> joint_pos	Joint
CartesianPosition cart_pos	Cartesian posture
std::vector<std::pair<std::string, bool>> digital_signals	Digital IO
std::vector<std::pair<std::string, double>> analog_signals;	Analog IO
OperateMode operation_mode	Operate mode
double speed_override	Speed coverage ratio

8.2.3 Frame rokae::Frame

std::array< double, 3 > trans	Translation, [X, Y, Z], in m
std::array< double, 3 > rpy	XYZ Euler angle, [A, B, C], in rad
std::array< double, 16 > pos	Row-major transformation matrix, only used for Cartesian position/impedance control under the real-time mode

8.2.4 Cartesian point position rokae::CartesianPosition

std::array< double, 3 > trans	Translation, [x, y, z], in m
std::array< double, 3 > rpy	XYZ Euler angle, [A, B, C], in rad
double elbow	Elbow, applicable to 7-axis robots, in rad
bool hasElbow	Existence of an elbow
std::vector< int > confData	Axis configuration data with the number of elements corresponding to that of robot axes
std::vector< double > external	External joint value, in rad/m Rail, in m

8.2.5 Cartesian point position offset rokae::CartesianPosition::Offset

Type type	Type: Offs/RelTool
Frame frame	Offset frame

8.2.6 Joint point position rokae::JointPosition

std::vector< double > joints	Joint angle, in rad
std::vector< double > external	External joint value, in rad/m Rail, in m

8.2.7 Joint torque, excluding gravity and friction force rokae::Torque

std::vector< double > tau	Desired joint torque, in N.m

8.2.8 Load information rokae::Load

double mass	Load mass, in kg
std::array< double, 3 > cog	Center of mass [x, y, z], in m
std::array< double, 3 > inertia	Inertia [ix, iy, iz], in kg·m²

8.2.9 Tool & work object information rokae::Toolset

Calculated based on a pair of tool & work object's coordinates, load, and handheld settings of the robot.

Load load	Robot end-effector handheld load
Frame end	Transformation of the robot end-effector frame in the flange frame
Frame ref	Transformation of the robot end-effector frame relative to the flange frame

8.2.10 Frame calibration result rokae::FrameCalibrationResult

Frame frame	Calibration result
std::array<double, 3> errors	Deviation between sample points and TCP calibration values, listed as minimum, average, and maximum, in m

8.2.11 RL project information rokae::RLProjectInfo

std::string name	Project name
std::vector< std::string > taskList	Task name list

8.2.12 Tool/Work object information rokae::WorkToolInfo

std::string name	Name
std::string alias	Description
bool robotHeld	Whether it is robot-handheld
Frame pos	Posture. **The work object frame has been transformed relative to its user frame**
Load load	Payload

8.2.13 Motion command MoveAbsJ rokae::MoveAbsJCommand

JointPosition target	Target joint point position
double speed	End-effector linear speed, in mm/s. The joint speed is divided into several ranges based on the magnitude of the end-effector linear speed. See setDefaultSpeed() for details
double jointSpeed	Joint Speed Percentage
double zone	Size of turning zone, in mm
std::string customInfo	Custom information, included in the motion feedback

8.2.14 Motion command MoveJ rokae::MoveJCommand

CartesianPosition target	Target Cartesian point position
double speed	End-effector linear speed, in mm/s. The joint speed is divided into

	several ranges based on the magnitude of the end-effector linear speed. See setDefaultSpeed() for details
double jointSpeed	Joint Speed Percentage
double zone	Size of turning zone, in mm
CartesianPosition::Offset offset	Offset
std::string customInfo	Custom information, included in the motion feedback

8.2.15 Motion command MoveL rokae::MoveLCommand

CartesianPosition target	Target Cartesian point position
double speed	End-effector linear speed, in mm/s
double rotSpeed	Orientation Speed
double zone	Size of turning zone, in mm
CartesianPosition ::Offset offset	Offset
std::string customInfo	Custom information, included in the motion feedback

8.2.16 Motion command MoveC rokae::MoveCCommand

CartesianPosition target	Target Cartesian point position
CartesianPosition aux	Auxiliary point position
double speed	End-effector linear speed, in mm/s
double rotSpeed	Orientation Speed
double zone	Size of turning zone, in mm
CartesianPosition ::Offset targetOffset	Target point offset
CartesianPosition ::Offset auxOffset	Auxiliary point offset
std::string customInfo	Custom information, included in the motion feedback

8.2.17 Motion command MoveCF rokae::MoveCFCommand

CartesianPosition target	Auxiliary point position 1
CartesianPosition aux	Auxiliary point position 2
double speed	End-effector linear speed, in mm/s
double rotSpeed	Orientation Speed
double zone	Size of turning zone, in mm
double angle	Full-circle execution angle, in rad
CartesianPosition ::Offset targetOffset	Target point offset
CartesianPosition ::Offset auxOffset	Auxiliary point offset
RotType rotType	Full-circle pose rotation mode
std::string customInfo	Custom information, included in the motion feedback

8.2.18 Motion command MoveSP rokae:: MoveSPCommand

CartesianPosition target	End Cartesian point position
CartesianPosition::Offset targetOffset	Offset option
double speed	End-effector linear speed, in mm/s
double radius	Initial radius, in m
double radius_step	Changes in radius per unit angle of rotation, in m/rad
double angle	Total rotation angle, in rad
bool direction	Rotation direction, true – clockwise | false – counterclockwise
std::string customInfo	Custom information, included in the motion feedback

8.2.19 Motion pause instruction rokae:: MoveWaitCommand

std::chrono::steady_clock::duration duration_	Pause duration, with a minimum effective duration of 1 ms

8.2.20 Controller log information rokae::LogInfo

const int id	Log ID
const std::string timestamp	Date and time
const std::string content	Log content
const std::string repair	Repair method

8.2.21 End-effector keypad state rokae::KeyPadState

bool key1_state	No. CR1
bool key2_state	No. CR2
bool key3_state	No. CR3
bool key4_state	No. CR4
bool key5_state	No. CR5
bool key6_state	No. CR6
bool key7_state	No. CR7

8.3 Method

8.3.1 Basic Robot Operations and Information Query

connectToRobot() [1/2]
template<WorkType Wt, unsigned short DoF> void rokae::Robot_T< Wt, DoF >::connectToRobot (error_code &ec) Establish a connection with the robot. The robot address is required when creating the robot instance
Parameter [out] ec Error code

connectToRobot() [2/2]
template<WorkType Wt, unsigned short DoF> void rokae::Robot_T< Wt, DoF >::connectToRobot (const std::string &remoteIP, const std::string &localIP = "") Connected to the Robot
Parameter RemoteIP robot IP address LocalIP local IP address. Used for sending and receiving interactive data in real-time mode; optional. Not supported on the PCB3/4-axis model

disconnectFromRobot()
void rokae::BaseRobot::disconnectFromRobot (error_code & ec) Disconnect from the robot. The robot motion will be stopped before disconnection. Please pay attention to safety
Parameter [out] ec Error code

setConnectionHandler()
void rokae::BaseRobot::setConnectionHandler(const std::function<void(bool)> &handler); Set up a disconnection callback function
Parameter [in] handler – callback function with bool parameter, where true indicates connected and false indicates disconnected

robotInfo()
Info rokae::BaseRobot::robotInfo (error_code &ec) const Query basic robot information
Parameter [out] ec Error code **Return** Basic robot information: controller version, robot model, number of axes

powerState()
PowerState rokae::BaseRobot::powerState(error_code &ec) const Robot power on/off and emergency stop statuses
Parameter [out] ec Error code **Return** on - powered on | off - powered off | estop - emergency stop | gstop - safety gate opened

setPowerState()
void rokae::BaseRobot::setPowerState (bool on, error_code & ec) Power on/off the robot. Note: Only robots without external enabling switches or teach pendants can be manually powered on.
Parameter [in] on true - power on | false - power off [out] ec Error code

operateMode()
OperateMode rokae::BaseRobot::operateMode (error_code & ec) const Query the robot's current operating mode
Parameter [out] ec Error code **Return** Manual | Automatic

setOperateMode()
void rokae::BaseRobot::setOperateMode (OperateMode mode, error_code & ec) Switch between automatic/manual mode
Parameter [in] mode Manual/Automatic

	[out] ec Error code

operationState()
OperationState rokae::BaseRobot::operationState (error_code & ec) const Query the robot's current operating state (idle, moving, drag enabled, etc.)
Parameter [out] ec Error code **Return** Robot operating state enumeration type

posture()
std::array< double, 6 > rokae::BaseRobot::posture(CoordinateType ct, error_code & ec) Get the current posture of the robot flange or end-effector
Parameter [in] ct Frame type 1) flangeInBase: Frame of flange in base; 2) endInRef: End-effector in the external reference frame For example, when a hand-held tool and an external work object are defined, this frame type returns the tool coordinate in the work object frame. [out] ec Error code **Return** double array, [X, Y, Z, Rx, Ry, Rz], where translation is in m and rotation in rad

cartPosture()
CartesianPosition rokae::BaseRobot::cartPosture(CoordinateType ct, error_code & ec) Get the current posture of the robot flange or end-effector
Parameter [in] ct Frame type [out] ec Error code **Return** Current Cartesian position

jointPos() [1/2]
template<WorkType Wt, unsigned short DoF> std::array< double, DoF > rokae::Robot_T< Wt, DoF >::jointPos (error_code & ec) Current robot axis angle, in rad
Parameter [out] ec Error code **Return** Axis angle

jointPos() [2/2]
std::vector<double> rokae::BaseRobot::jointPos (error_code & ec) Current axis angle of the robot, robot body axis + external axis, in rad; external axis rail, in m
Parameter [out] ec Error code **Return** Joint angle and external axis

jointVel() [1/2]
template<WorkType Wt, unsigned short DoF> std::array< double, DoF > rokae::Robot_T< Wt, DoF >::jointVel (error_code & ec) Current robot joint speed, in rad/s
Parameter [out] ec Error code **Return** Joint speed

jointVel() [2/2]
std::vector<double> rokae:: BaseRobot::jointVel (error_code & ec) Current joint speed of robot, robot body + external axis, in rad/s; external axis, in m/s
Parameter [out] ec Error code **Return** Joint speed

getStateList()
StateList rokae::BaseRobot::getStateList(error_code &ec) Query the current position, IO signal, operating mode, and speed coverage value

Parameter	[out] ec Error code
Return	Query results

jointTorque()	
template<WorkType Wt, unsigned short DoF> std::array< double, DoF > rokae::Robot_T< Wt, DoF >::jointTorque (error_code &ec) Value of joint force sensor, in N.m	
Parameter	[out] ec Error code
Return	Torque value

baseFrame()	
std::array< double, 6 > rokae::BaseRobot::baseFrame (error_code &ec) const Base frame defined by users, in the world frame	
Parameter	[out] ec Error code
Return	double array, [X, Y, Z, Rx, Ry, Rz], where translation is in m and rotation in rad

setBaseFrame()	
void rokae::BaseRobot::setBaseFrame(const Frame &frame, error_code &ec) Set the base frame, save only numerical values after setting, and take effect after restarting the controller	
Parameter	[in] frame Frame, default to using custom installation method [out] ec Error code

toolset()	
Toolset rokae:: BaseRobot::toolset (std::error_code & ec) const Query current tool & work object information	
Note	This pair of tool & work object is only used for SDK motion control and is independent of RL projects.
Parameter	[out] ec Error code
Return	See the Toolset data structure

setToolset()	
void rokae:: BaseRobot::setToolset (const Toolset & toolset, error_code & ec) Set tool & work object information	
Note	This pair of tool & work object is only used for SDK motion control and is independent of RL projects. After setting, the top right corner of RobotAssist will display "toolx" and "wobjx", and the end-effector coordinates displayed in the state monitoring will also change. In addition to this API, if the default tool and work object are changed through RobotAssist (option in the upper right corner), the tool & work object will also be changed accordingly
Parameter	[in] toolset tool & work object information [out] ec Error code

setToolset()	
void rokae:: BaseRobot::setToolset (const std::string &toolName, const std::string &wobjName, error_code &ec) Set the tool & work object information using the tools and work objects that have been created.	
Note	Setting prerequisite: An RL project has been loaded and tools and work objects have been created. Otherwise, only the default tool and work object are available, i.e. "tool0" and "wobj0". The tool and work object cannot be simultaneously handheld or external. If there is a conflict, the position of the tool prevails, for example, the tool and work object are simultaneously handheld, no error is returned, but the coordinate of the work object becomes external.
Parameter	[in] toolName: tool name [in] wobjName: work object name [out] ec Error code

calcIk() [1/2]

template<unsigned short DoF> JointArray rokae::Model_T< DoF >::calcIk (CartesianPosition posture, error_code &ec) Inverse kinematics based on posture
Parameter [in] posture Robot end-effector posture, in the external reference frame [out] ec Error code **Return** Axis angle, in rad

calcIk() [2/2]
template<unsigned short DoF> JointArray rokae::Model_T< DoF >::calcIk (CartesianPosition posture, const Toolset & toolset error_code &ec) Calculate the inverse kinematics result in the given tool/work object frame based on the posture
Parameter [in] posture Robot end-effector posture, in the external reference frame [in] toolset tool & work object information [out] ec Error code **Return** Axis angle, in rad

calcFk()
template<unsigned short DoF> CartesianPosition rokae::Model_T< DoF >::calcFk (const JointArray & joints, error_code & ec) Calculate the forward kinematics result based on the axis angle
Parameter [in] joints Axis angle, in rad [out] ec Error code **Return** Robot end-effector posture, in the external reference frame

calcFk()
template<unsigned short DoF> CartesianPosition rokae::Model_T< DoF >::calcFk (const JointArray & joints, const Toolset & toolset error_code & ec) Calculate the forward kinematics result in the given tool/work object frame based on the axis angle
Parameter [in] joints Axis angle, in rad [in] toolset tool & work object information [out] ec Error code **Return** Robot end-effector posture, in the external reference frame

setToolset()
void rokae:: BaseRobot::setToolset (const std::string &toolName, const std::string &wobjName, error_code &ec) Set the tool & work object information using the tools and work objects that have been created.
Note Prerequisites: 1) Using tool and work object defined in an RL project: The corresponding RL project must be loaded first. 2) Global tool and work object: They can be used directly without loading a project. e.g. setToolset("g_tool_0", "g_wobj_0") The tool and work object cannot be simultaneously handheld or external. If there is a conflict, the position of the tool prevails, for example, the tool and work object are simultaneously handheld, no error is returned, but the coordinate of the work object becomes external. **Parameter** [in] toolName: tool name [in] wobjName: work object name [out] ec Error code

calibrateFrame ()
template<WorkType Wt, unsigned short DoF> FrameCalibrationResult rokae::Robot_T< Wt, DoF >::calibrateFrame (FrameType type,

const std::vector< std::array< double, DoF > > &points, bool is_held, error_code &ec, const std::array< double, 3 > & base_aux = {}) Frame calibration (N-point calibration)
Note Calibration methods and precautions supported by various frame types: 1) Tool frame: three-point/four-point/six-point calibration 2) Work object frame: three-point calibration The calibration result will not be transformed relative to the user frame, that is, if it is an external workpiece, the returned result is relative to the base frame. 3) Base frame: Six-point calibration. Please ensure that the dynamic constraints and feedforward are turned off before calibration. If the calibration is successful (without error codes), the controller will automatically save the calibration results and take effect after restarting the controller. 4) Rail frame: Three-point calibration. If the calibration is successful (without error codes), the controller will automatically save the calibration results and take effect after restarting the controller. **Parameter** [in] points Axis angle list, with a length of N. For example, when using the three-point method to calibrate the tool frame, 3 sets of axis angles should be passed in. The unit of axis angle is radians. [in] is_held true - obot handheld | false - external. Only affects the calibration of tools/workpieces. [out] ec Error code [in] base_aux Auxiliary points used for base frame calibration, in m. **Return** Calibration result, valid when the error code is not set

clearServoAlarm()
void rokae::BaseRobot::clearServoAlarm (error_code & ec) Clear servo alarms
Parameter [out] ec Error code, which is set to −1 in case of a servo alarm that cannot be cleared.

enableCollisionDetection()
template<unsigned short DoF> void Cobot<DoF>::enableCollisionDetection(const std::array<double, DoF> sensitivity, StopLevel behaviour, double fallback_compliance, error_code &ec) Set collision detection parameters and enable collision detection.
Parameter [in] sensitivity Collision detection sensitivity between 0.01 and 2.0 [in] behaviour Robot behavior after collision, including stop1 (safe stop, stop0 and stop1 are the same) and stop2 (triggered pause) [in] fallback_compliance 1) When the post-collision behavior is a safe stop or triggered pause, this parameter represents the fallback distance after collision, in m.2) When the post-collision behavior is a soft stop, this parameter indicates the compliance level, within the range [0.0, 1.0]. [out] ec Error code

disableCollisionDetection()
void BaseCobot::disableCollisionDetection(error_code &ec) Disable collision detection
Parameter [out] ec Error code

getSoftLimit()
template<WorkType Wt, unsigned short DoF> bool rokae::Robot_T< Wt, DoF >::getSoftLimit (std::array< double[2], DoF > &limits, error_code &ec) Get the current soft limit value
Parameter [out] limits Soft limits for each axis [lower limit, upper limit], in rad [out] ec Error code **Return** true - enabled | false - disabled

setSoftLimit()
template<WorkType Wt, unsigned short DoF> void rokae::Robot_T< Wt, DoF >::setSoftLimit (bool enable, error_code &ec, const std::array< double[2], DoF > & limits = {{DBL_MAX, DBL_MAX}}) Set the soft limit. Soft limit setting requirements: 1) When enabling the soft limit, the robotic arm should be powered off and in manual mode; 2) The soft limit cannot exceed the mechanical hard limit; 3) The current angle of each axis of the robotic arm should be within the set limit range.

Parameter	[in] enable true- enabled | false - disabled [out] ec Error code [in] limits Each axis [lower limit, upper limit], in rad. 1) When "limits" is the default value, it is considered that only the soft limit is enabled without modifying the value; When it is not the default value, modify the soft limit first and then open it. 2) When disabling the soft limit, the limit value will not be modified

queryControllerLog()

std::vector< LogInfo > rokae::BaseRobot::queryControllerLog (unsigned count,
const std::set< LogInfo::Level > & level,
error_code & ec)

Query the latest controller log

Parameter	[in] count number of queries, max. 10 [in] level Specify the log level. An empty set means the log level is not specified. [out] ec Error code
Return	Log information

recoverState()

void rokae::BaseRobot::recoverState(int item, error_code &ec)

Restore the robot state according to the options

Parameter	[in] item Recovery option, 1: Emergency stop recovery [out] ec Error code

rebootSystem()

void rokae::BaseRobot::rebootSystem (error_code& ec);

Restart IPC Note: Restarting operation is not allowed in the automatic mode, power-off state, motion, and non-idle state

Parameter	[out] ec Error code

setRailParameter()

template<typename R>
void rokae::setRailParameter(const std::string &name, R value, error_code &ec)

Set rail parameters

Template parameters	Parameter type
Parameter	[in] name Parameter name. See the value description [in] value

Parameter	Parameter Name	Data Type
Switch	enable	bool
Base frame	baseFrame	Frame
Rail name	name	std::string
Encoder resolution	encoderResolution	int
Reduction ratio	reductionRatio	double
Maximum speed of motor (rpm)	motorSpeed	int
Soft limit (m), [lower limit, upper limit]	softLimit	std::vector<double>
Range of motion (m), [lower limit, upper limit]	range	std::vector<double>
Maximum speed (m/s)	maxSpeed	double
Maximum acceleration (m/s^2)	maxAcc	double
Maximum jerk (m/s^3)	maxJerk	double

[out] ec Error code

getRailParameter()

template<typename R>
void rokae::getRailParameter (const std::string &name, R &value, error_code &ec)

Read rail parameter

Template parameters	Parameter type
Parameter	[in] name Parameter name. See setRailParameter() [out] value Parameter values. See setRailParameter() [out] ec Error code, indicating that parameter name does not exist or that data type does not match with the returned error code

configNtp()

void rokae::BaseRobot::configNtp(const std::string &server_ip, error_code &ec)

Configure NTP. A non-standard feature requires additional installation

Parameter	[in] server_ip NTP server IP

	[out] ec Error code, indicating that NTP is not installed correctly or that the IP address format is incorrect

syncTimeWithServer()	
void rokae::BaseRobot::syncTimeWithServer(error_code &ec) It is required to manually perform a one-time time synchronization, and the remote IP is configured via configNtp. The operation takes several seconds to block and wait until synchronization is complete. The API has a default timeout of 12s.	
Parameter	[out] ec Error code. NTP service is not installed correctly or cannot synchronize with the server

sdkVersion()	
static std::string rokae::BaseRobot::sdkVersion ( ) Query xCore SDK version	
Return	Version No.

setTeachPendantMode()	
void rokae::BaseRobot::setTeachPendantMode (bool enable, error_code& ec) Teach pendant hot plug. Note: The teach pendant hot plug is only available for some models. An error code will be returned for unsupported models. After not using the teach pendant, the enable button and emergency stop button will become invalid	
Parameter	[in] enable true - using the teach pendant | false - not using the teach pendant [out] ec The current robot state cannot be switched (in motion/power-on in manual mode); switching fails due to reasons such as models

8.3.2 Motion control (non-real-time mode)

setMotionControlMode()	
void rokae::BaseRobot::setMotionControlMode (MotionControlMode mode, error_code & ec) Set motion control mode	
Note	The control mode must be set before calling motion control APIs.
Parameter	[in] mode Mode [out] ec Error code

moveReset()	
void rokae::BaseRobot::moveReset (error_code &ec) Reset motion, and clear sent motion commands and execution information	
Note	The robot class calls a motion reset once during initialization. A motion reset is also required before switching control between RL programs and SDK motion commands.
Parameter	[out] ec Error code

stop()	
void rokae::BaseRobot::stop (error_code & ec) After pausing the robot motion, you can call moveStart() to continue the moving. If it is required to stop completely and no longer execute the added commands, you can call moveReset().	
Note	Currently, only stop2 is supported. The robot is scheduled to stop and remain powered-on. For details, refer to StopLevel. After calling this API, you can call moveStart() to continue the moving after pause. If it is required to stop completely and no longer execute the added commands, you can call moveReset().
Parameter	[out] ec Error code

moveStart()	
void rokae::BaseRobot::moveStart(error_code &ec) Start/continue moving	
Parameter	[out] ec Error code

moveAppend() [1/3]	
template<class Command > void rokae::BaseRobot::moveAppend (const std::vector<Command> &cmds, std::string &cmdID, error_code &ec) Add single or multiple moving command(s), and then call moveStart() to start the moving.	
Template parameters	Command Motion command: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand
Parameter	[in] cmds Command list, covering 1-100 commands of the same type

	[out] cmdID Command ID, which can be used to query the command execution information [out] ec Code of errors before command issuing, including: 1) network connection error; 2) wrong number of commands

moveAppend() [2/3]	
template<class Command > void rokae::BaseRobot::moveAppend (std::initializer_list< Command > cmds, std::string &cmdID, error_code &ec) Add single or multiple moving command(s), and then call moveStart() to start the moving.	
Template parameters	Command Motion command: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand
Parameter	[in] cmds Command list, covering 1-100 commands of the same type [out] cmdID Command ID, which can be used to query the command execution information [out] ec Code of errors before command issuing, including: 1) network connection error; 2) wrong number of commands

moveAppend() [3/3]	
template<class Command > void rokae::BaseRobot::moveAppend(const Command &cmds, std::string &cmdID, error_code &ec) Add a single moving command, and then call moveStart() to start the moving.	
Template parameters	Command Motion command: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand | MoveWaitCommand
Parameter	[in] cmds Command [out] cmdID Command ID, which can be used to query the command execution information [out] ec Code of errors before command issuing, including: 1) network connection error; 2) wrong number of commands

executeCommand() [1/2]	
template<class Command > void rokae::BaseRobot::executeCommand(std::initializer_list< Command > cmds, error_code & ec) Execute single or multiple motion command(s), to start the robot motion immediately after calling	
Template parameters	Command Motion command: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand;
Parameter	[in] cmds List of commands, with 1–1,000 commands allowable [out] ec Error code, only feedback on errors before execution, including: 1) network connection issues; 2) inconsistent number of commands; 3) unable motion of the robot in its current state, such as not being powered on

executeCommand() [2/2]	
template<class Command > void rokae::BaseRobot::executeCommand(std::vector< Command > cmds, error_code & ec) Execute single or multiple motion command(s), to start the robot motion immediately after calling	
Template parameters	Command Motion command: MoveJCommand | MoveAbsJCommand | MoveLCommand | MoveCCommand | MoveCFCommand | MoveSPCommand;
Parameter	[in] cmds List of commands, with 1–1,000 commands allowable [out] ec Error code, only feedback on errors before execution, including: 1) network connection issues; 2) inconsistent number of commands; 3) unable motion of the robot in its current state, such as not being powered on

setDefaultSpeed()	
void rokae::BaseRobot::setDefaultSpeed (double speed, error_code &ec) Set a default motion speed, with an initial value of 100	
Note	This value represents the maximum end-effector linear speed (in mm/s), and the corresponding joint speed is automatically calculated
Parameter	[in] speed The API places no restrictions on the range of parameters. The actual effective range of end-effector linear speed is (0, 4000] for collaborative robots and (0, 7000] for industrial robots. The percentage of joint speed is divided into 5 ranges: < 100: 10%; 100–200: 30%; 200–500: 50%; 500–800: 80%; > 800: 100% [out] ec Error code

setDefaultZone()
void rokae::BaseRobot::setDefaultZone (double zone, error_code & ec)

Set default turning zone	
Note	This value indicates the maximum radius of turning zone (in mm), and the turning percentage is automatically calculated. If not set, the value is 0 (fine, no turning zone).
Parameter	[in] zone The API places no restrictions on the range of parameters. The actual effective range of turning zone radius is 0–200. The turning percentage is divided into 4 ranges: < 1: 0 (fine); 1–20: 10%; 20–60: 30%; > 60: 100% [out] ec Error code

setDefaultConfOpt()	
void rokae::BaseRobot::setDefaultConfOpt(bool forced, error_code &ec)	
Set whether to use axis configuration data (confData) to calculate the inverse kinematics result, with the initial value, false	
Parameter	[in] forcedtrue - strictly followed, at this point, the inverse Cartesian point solution will be calculated using the confData of the motion instruction. If the calculation fails, an error will be returned; False - not strictly followed, the closest solution of the current axis angle of the robotic arm will be selected for inverse solution [out] ec Error code

setMaxCacheSize()	
void rokae::BaseRobot::setMaxCacheSize(int number, error_code &ec)	
Set the maximum number of buffered commands, which refers to the number of path points pending planning in the controller. The allowable range is [1, 1,000], with a default value of 300.	
Note	If the trajectory is mostly short, you can increase this value to avoid the robot stopping due to delayed command sending (if there are unexecuted commands after stopping, you can continue with moveStart());
Parameter	[in] number Number [out] ec Error code

setAutoIgnoreZone()	
void rokae::BaseRobot::setAutoIgnoreZone(bool enable, error_code& ec)	
Set whether motion commands automatically cancel the turning zone, with the initial value, true	
Parameter	[in] enable true - automatically canceling the turning zone | false - not automatically canceling the turning zone [out] ec Error code

adjustSpeedOnline()	
void rokae::BaseRobot::adjustSpeedOnline(double scale, error_code &ec)	
Dynamically adjust the robot motion speed, effective in non-real-time mode.	
Parameter	[in] scale The scale for the speed of motion commands, with the range of 0.01–1. When the scale is set to 1, the robot will move at the original speed defined in the path. [out] ec Error code

getAcceleration()	
void rokae::BaseRobot::getAcceleration(double &acc, double &jerk, error_code &ec)	
Read the current acceleration/deceleration and Jerk	
Parameter	[out] acc Percentage of preset acceleration in the system [out] jerk Percentage of preset jerk in the system [out] ec Error code

adjustAcceleration()	
void rokae::BaseRobot::adjustAcceleration(double acc, double jerk, error_code &ec)	
Adjust the acceleration/deceleration and jerk of the motion. If it is called during robot motion, the currently executing command will not take effect, and the next one will do	
Parameter	[in] acc Percentage of the system default acceleration, with the range [0.2, 1.5]. Values outside this range are not considered errors and will be clamped to the nearest valid limit

	[in] jerk Percentage of the system default jerk, with the range [0.1, 2]. Values outside this range are not considered errors and will be clamped to the nearest valid limit [out] ec Error code

setEventWatcher()	
void rokae::BaseRobot::setEventWatcher(Event eventType, const EventCallback &callback, error_code &ec) Set the callback function for receiving events	
Parameter	[in] eventType Event type [in] callback Callback function for handling events Note: 1) For Event::moveExecution, the callback function is executed in the same thread. It is required to avoid performing long-running operations inside the function; 2) For Event::safety, each callback runs in a separate thread, and there is no restriction on execution time [out] ec Error code

queryEventInfo()	
EventInfo rokae::BaseRobot::queryEventInfo(Event eventType, error_code &ec) Query work object information Same information as provided by the setEventWatcher() during the callback. But this API uses an active query approach instead.	
Parameter	[in] eventType Event type [out] ec Error code
Return	Event information

startJog()	
void rokae::BaseRobot::startJog (JogOpt::Space space, double rate, double step, unsigned index, bool direction, error_code &ec) Start jogging the robot after switching to the manual mode.	
Note	After the robot starts motion by calling this API, stop() must be called to stop jogging regardless of whether the robot has stopped motion automatically. Otherwise, the robot will remain in the jogging state.
Parameter	[in] space jog reference frame. 1) The usage principle of tool/work object frame is the same as that of setToolset(); 2) Both industrial six-axis models and xMateCR/SR six-axis models support two singularity avoidance methods of jog: Space::singularityAvoidMode, Space::baseParallelMode 3) CR5-axis models support only the base-parallel method of jog: Space::baseParallelMode [in] rate Rate between 0.01 and 1 [in] step Step length. Cartesian space - mm | joint space - degree. The step length is a positive value without an upper limit. If the robot cannot continue jogging, it will stop moving automatically. [in] index The meaning of this parameter varies depending on different space: 1) World frame, base frame, flange frame, tool/work object frame: a) For 6-axis models: Indexes 0–5 correspond to X, Y, Z, Rx, Ry, and Rz, respectively. Values greater than 5 represent external axes (if any). b) For 7-axis models: Index 6 corresponds to the elbow joint. Values greater than 6 represent external axes (if any). 2) Joint space: Joint number, starting from 0. 3) Singularity avoidance mode, base-parallel mode: a) For 6-axis models: Indexes 0–5 correspond to X, Y, Z, J4 (4-axis), Ry, and J6 (6-axis), respectively; b) For 5-axis models: Indexes 0–4 correspond to X, Y, Z, Ry, and J5 (5-axis), respectively. [in] direction The meaning of this parameter depends on different space and indexes, as follows: 1) Singularity avoidance mode (J4): true - ±180° | false - 0°; 2) Base-parallel mode (J4 & Ry): true - ±180° | false - 0°; 3) Others: true - positive direction | false - negative direction [out] ec Error code

setAvoidSingularity()
void rokae::xMateRobot::setAvoidSingularity (AvoidSingularityMethod method, bool enable, double limit, error_code &ec)
void rokae::StandardRobot::setAvoidSingularity (AvoidSingularityMethod method, bool enable, double limit, error_code &ec)

Enable or disable the singularity avoidance feature. This is only supported on some models:
1) Four-axis locking: Supported on industrial six-axis, xMateCR, and xMateSR six-axis models;
2) Sacrifice pose: Supported on all six-axis models;
3) Joint space interpolation: Supported on industrial six-axis models only.

Parameter

[in] method Singularity avoidance method

[in] enable true - enabled | false - disabled For the four-axis locking mode, ensure that axis 4 is at its zero position before enabling the feature.

[in] limit The meaning of this parameter varies depending on different avoidance methods:
1) Sacrifice pose: the maximum allowable pose error, with the range (0, PI*2], in rad
2) Joint space interpolation: singularity avoidance radius, with a range of [0.005, 10], in m
3) Four-axis locking: no parameters

[out] ec Error code

getAvoidSingularity()

bool rokae::xMateRobot::getAvoidSingularity (AvoidSingularityMethod method, error_code &ec)

bool rokae::StandardRobot::getAvoidSingularity (AvoidSingularityMethod method, error_code &ec)
Query whether it is in a state of avoiding singularity

Parameter

[in] method Singularity avoidance method

[out] ec Error code

Return

true - enabled | false - disabled

getAvoidSingularity()

bool rokae::xMateRobot::getAvoidSingularity (AvoidSingularityMethod method, error_code &ec)

bool rokae::StandardRobot::getAvoidSingularity (AvoidSingularityMethod method, error_code &ec)
Query whether it is in a state of avoiding singularity

Parameter

[in] method Singularity avoidance method

[out] ec Error code

Return

true - enabled | false - disabled

checkPath() [1/3]

```
std::vector<double> rokae::BaseRobot::checkPath(const CartesianPosition &start,
                                                const std::vector<double> &start_joint,
                                                const CartesianPosition &target,
                                                error_code &ec)
```

Check whether a Cartesian trajectory is reachable — linear path only. With the rail supported, the returned target joints include both axes and external axes

Parameter

[in] start Start point

[in] start_joint Start joint [rad]

[in] target Target point

[out] ec Error code

Return

The calculated target joint angles are valid only when there is no error code

checkPath() [2/3]

```
int rokae::BaseRobot::checkPath(const std::vector<double> &start_joint,
                                const std::vector<CartesianPosition> &points,
                                std::vector<double> &target_joint_calculated,
                                error_code &ec)
```

Verify multiple linear trajectories

Parameter

[in] start_joint Start joint, in rad

[in] points Cartesian point position, requiring at least 2 points, with the first one being the start point

[out] target_joint_calculated If verification passes, the calculated target joints are returned

[out] ec Reason for verification failure

Return

If verification fails, return the index of the failed target point in points; otherwise, return 0

checkPath() [3/3]	
std::vector<double> rokae::BaseRobot::checkPath(const CartesianPosition &start, const std::vector<double> &start_joint, const CartesianPosition &aux, const CartesianPosition &target, error_code &ec, double angle =0.0, MoveCFCommand::RotType rot_type = MoveCFCommand::constPose) Check whether a Cartesian trajectory is reachable, including arcs and full circles. With the rail supported, the returned target joints include both axes and external axes	
Parameter	[in] start Start point [in] start_joint Start joint [rad] [in] aux Auxiliary point [in] target Target point [out] ec Error code [in] angle Full-circle execution angle — a non-zero value indicates that the full-circle trajectory should be validated [in] rot_type Full-circle rotation type
Return	The calculated target joint angles are valid only when there is no error code

8.3.3 Real-time motion control

getRtMotionController()	
template <unsigned short DoF> std::weak_ptr<RtMotionControlCobot<DoF>> Cobot<DoF>::getRtMotionController ( ) std::weak_ptr<RtMotionControlIndustrial<6>> StandardRobot::getRtMotionController() Create an instance of the real-time motion control class, and use the pointer to this instance to perform operations related to real-time mode.	
Note	Unless this API is called repeatedly, the internal logic of the client will not actively destruct the returned objects, including but not limited to disconnecting from the robot disconnectFromRobot(), switching to non-real-time motion control mode, etc. However, exceptions may occur in real-time mode control after the above operations are performed.
Return	Controller object
Exceptions	RealtimeControlException Failed to create RtMotionControl instance due to network exceptions ExecutionException Not switched to real-time motion control mode

reconnectNetwork()	
void rokae::MotionControl< MotionControlMode::RtCommand >::reconnectNetwork (error_code & ec)	
Reconnect to real-time control server	
Parameter	[out] ec Error code

disconnectNetwork()
void rokae::MotionControl< MotionControlMode::RtCommand >::disconnectNetwork ( )
Disconnect from the real-time control server and disable ports for receiving data and sending commands. However, it will not disconnect from the robot. If the robot is in motion, immediately stop the robot motion after disconnection.

setControlLoop()	
template<class Command> void rokae::MotionControl< MotionControlMode::RtCommand >::setControlLoop (const std::function<Command(void)>& callback, int priority = 0, bool useStateDataInLoop = false) Set callback functions using periodic scheduling.	
Note	1) The callback function plans the moving command according to the period of 1 ms, and the planning result is the return value of the function. The SDK filters the return value and sends it to the controller. 2) The length of the JointPosition array and that of the JointTorque array should be the same as the number of robot axes. If they are different, no error will be reported, but an unreasonable command may be issued. 3) At the end of a moving cycle, it can be identified by the returned Command.setFinish(), and the SDK will stop the moving and the callback function.
Template	JointPosition | CartesianPosition | Torque

parameters	
Parameter	[in] callback: callback function Depending on the control mode (RtControllerMode), there are 3 types of function return values: joint angle/Cartesian posture/torque. Among which, Cartesian posture uses a rotation matrix to represent the rotation amount, and pos is the posture of the end-effector in base. [in] priority: task priority, and 0 for not specified. This parameter is only valid in a real-time operating system. If it cannot be set, a controller error message will be printed. [in] useStateDataInLoop: confirm whether to use state data in loop. When it is set to true: 1) xCore-SDK will update the real-time state data (updateStateData()) before the callback function, and directly getStateData() in the callback function. 2) The sending cycle of the state data should be consistent with the control cycle, which is 1ms: startReceiveRobotState(interval = milliseconds(1)).

startLoop()	
void rokae::MotionControl< MotionControlMode::RtCommand >::startLoop (bool blocking = true)	
Start executing periodic scheduling tasks.	
Parameter	[in] blocking Whether to block the thread calling this function. If the thread is non-blocking, you must call stopLoop() to stop the scheduled task; otherwise, the next cycle cannot start.
Exceptions	RealtimeControlException Network error when sending commands; the command type does not match the control mode; or errors occur when the sent commands are executed by the controller.

stopLoop()	
void rokae::MotionControl< MotionControlMode::RtCommand >::stopLoop ( )	
Stop executing periodic scheduling tasks.	
Exceptions	RealtimeControlException Errors occur during execution

startMove()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::startMove (RtControllerMode rtMode) Specify the control mode for the robot to get ready for motion. This API needs to be called before execution of each callback. Calling this API does not cause the robot to start moving immediately; motion will begin only after a motion command is sent.	
Note	1) Set parameters such as filter impedance before calling startMove(). After startMove is called, execution of other commands such as power off may fail. The correct way to stop is to call stopMove; 2) If the state data to be received is not set using startReceiveRobotState(), the data will be set automatically when this function is called.
Parameter	[in] rtMode Control mode
Exceptions	RealtimeStateException Duplicate call after motion has already started RealtimeParameterException Unsupported control modes specified RealtimeControlException The controller failed to switch to the specified control mode. This typically occurs when switching to the force control mode, causing the robot to stop moving and reject motion commands from the client.

stopMove()	
void rokae::MotionControl< MotionControlMode::RtCommand >::stopMove ( )	
Note	Besides, the JointPosition/CartesianPosition/Torque commands can mark the end of a motion cycle using setFinished(), and the robot will stop motion after marking, which displays the same result as calling stopMove(). This function is only used for real-time control and cannot be used to stop non-real-time motion commands.

startReceiveRobotState()	
template<WorkType Wt, unsigned short DoF> void rokae::Robot_T::startReceiveRobotState(std::chrono::steady_clock::duration interval, const std::vector<std::string>& fields) Get the robot to start sending real-time state data. Block and wait for the first-frame message. The timeout period is 3 seconds.	
Parameter	[in] interval Interval at which the controller sends state data; allowed values: 1 ms/2 ms/4 ms/8 ms/1s [in] fields Received robot state data. The maximum total length is 1024 bytes. The supported data and names can be found in data_types.h, RtSupportedFields
Exceptions	RealtimeControlException Unsupported state data is set; the robot cannot start sending data; or the total length exceeds 1024.

	RealtimeStateException The robot has started sending data, or the first-frame message is not received after timeout.

stopReceiveRobotState()
void rokae::BaseRobot::stopReceiveRobotState ( )
Stop receiving real-time state data and the controller stops sending data. It can be used to reset the state data to be received.

updateRobotState()	
unsigned rokae::BaseRobot::updateRobotState(std::chrono::steady_clock::duration timeout)	
Receive the robot state data once. Call this function before reading data every cycle. It is recommended to call according to the set sending cycle to obtain the latest data.	
Parameter	[in] timeout Timeout period
Return	Received data length. If no data is received before the timeout, then 0 is returned.
Exceptions	RealtimeControlException Unable to receive data, or unable to parse data due to errors in received data.

getStateData()	
template<typename R > int rokae::BaseRobot::::getStateData (const std::string &fieldName, R & data) Read robot state data	
Note	Note that the input data type should be consistent with the data type.
Template parameters	R data type
Parameter	[in] fieldName Data name [out] data Value
Return	If the data name does not exist, or it is not set as the data to be received using startReceiveRobotState(), or the data type is not the same as R, return −1. If the data is successfully read, return 0.
Exceptions	RealtimeStateException Network error

MoveJ()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::MoveJ (double speed, const std::array< double, DoF > & start, const std::array< double, DoF > & target) MoveJ command. The host computer plans the path. It is blocked before reaching the target. If errors occur during motion, it will stop the blocked state and return.	
Parameter	[in] speed Speed scaling factor [in] start The start joint angle must be the current joint angle of the robot. Otherwise, the robot may be powered off. [in] target Target joint angle of the robot
Exceptions	RealtimeMotionException Errors occur when the robot is in motion

MoveL()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::MoveL (double speed, CartesianPosition & start, CartesianPosition & target) MoveL command. The host computer plans the path. It is blocked before reaching the target. If errors occur during motion, it will stop the blocked state and return.	
Parameter	[in] speed Speed scaling factor between 0 and 1 [in] start The start posture must be the current robot posture. Otherwise, the robot may be powered off. If TCP is set, it should be the posture of the tool in the base frame. [in] target Robot target posture Similarly, if TCP is set, it should be the posture of TCP in the base frame
Exceptions	RealtimeParameterException Start pose or target posture parameter error RealtimeMotionException Errors occur when the robot is in motion

MoveC()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::MoveC (double speed, CartesianPosition & start, CartesianPosition & aux, CartesianPosition & target) MoveC command. It is blocked before reaching the target. If errors occur during motion, it will stop the blocked state and return.	
Parameter	[in] speed Speed scaling factor [in] start The start posture must be the current robot posture. If TCP is set, it should be the posture of the tool in the base frame. [in] aux Auxiliary point posture of the robot Similarly, if TCP is set, it should be the posture of TCP in the base frame [in] target Robot target posture Similarly, if TCP is set, it should be the posture of TCP in the base frame
Exceptions	RealtimeParameterException Point position error. Unable to calculate the arc path. RealtimeMotionException Errors occur when the robot is in motion

setFilterLimit()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::setFilterLimit(bool limit_rate, double cutoff_frequency) Set clipping and filtering parameters.	
Parameter	[in] limit_ratetrue - clipping enabled [in] cutoff_frequency Cut-off frequency Range: 0–1000 Hz, recommended range: 10–100 Hz.
Return	true - set successfully

setCartesianLimit()	
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::setCartesianLimit (const std::array< double, 3 > & lengths, const std::array< double, 16 > & frame, error_code & ec) Set Cartesian space motion area. The motion stops if it goes beyond the set area. Non-force-controlled virtual wall. If the robot end-effector or TCP end goes beyond the safe zone, the motor will also be powered off.	
Parameter	[in] lengths Length (X), width (Y), and height (Z) of the safe zone, in m [in] frame Posture of the cuboid center of the safe region in the base frame [out] ec Error code

setJointImpedance()	
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setJointImpedance (const std::array< double, DoF > & factor, error_code & ec) Set the joint space impedance control factor that takes effect during joint space impedance motion.	
Parameter	[in] factor Joint space impedance coefficient, in N.m/rad. The maximum stiffness of the xMateErPro model is { 3000, 3000, 3000, 3000, 300, 300, 300 } The maximum stiffness of other models is { 3000, 3000, 3000, 300, 300, 300 } The actual effective maximum value is related to the hardware state of sensors, etc. If there is shaking or other abnormal phenomena, please try to reduce the impedance coefficient. [out] ec Error code

setCartesianImpedance()	
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setCartesianImpedance (const std::array< double, 6 > & factor, error_code & ec) Set Cartesian space impedance control factor that takes effect during Cartesian impedance motion.	
Parameter	[in] factor Cartesian space impedance control factor, maximum { 3000, 3000, 3000, 300, 300, 300 }, in N/m, N.m/rad [out] ec Error code

setCollisionBehaviour()
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setCollisionBehaviour (const std::array< double, DoF > & torqueThresholds, error_code & ec) Set collision detection threshold Collision detection only takes effect during position control. It does not take effect during force control. If a collision is detected, the controller will issue a power-off command, and the motor will be powered off by the suction band-type brake.
Parameter [in] torqueThresholds Joint collision detection threshold, in N The maximum values for xMateErPro models are { 75, 75, 60, 45, 30, 30, 20 } The maximum values for other models are { 75, 75, 45, 30, 30, 20 } The maximum values for 5-axis models are { 75, 75, 45, 30, 20 } [out] ec Error code

setEndEffectorFrame()
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::setEndEffectorFrame (const std::array< double, 16 > & frame, error_code & ec) Set the end-effector posture in the robot flange. After TCP is set, the controller will save the settings, and default settings will be restored after the robot is restarted.
Parameter [in] frame Homogeneous matrix of the end-effector frame in the flange frame, in rad, m. [out] ec Error code

setLoad()
template <WorkType Wt, unsigned short DoF> void RtMotionControl<Wt, DoF>::setLoad (const Load & load, error_code & ec) Set the mass, center of mass, and inertia matrix of the tool and load. After the load is set, the controller will save the settings, and default settings will be restored after the robot is restarted.
Parameter [in] load Load information [out] ec Error code

setFilterFrequency()
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setFilterFrequency (double jointFrequency, double cartesianFrequency, double torqueFrequency, error_code & ec) Set the filter cut-off frequency of the robot controller to smooth the commands. Allowable range: 1–1,000 Hz, recommended range: 10–100 Hz.
Parameter [in] jointFrequency Filter cut-off frequency of the joint position, in Hz [in] cartesianFrequency Cartesian space position filter cut-off frequency, in Hz [in] torqueFrequency Filter cut-off frequency of the joint torque, in Hz [out] ec Error code

setCartesianImpedanceDesiredTorque()
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setCartesianImpedanceDesiredTorque(const std::array< double, 6 > & torque, error_code & ec) Set end-effector desired force that takes effect in Cartesian space impedance motion.
Parameter [in] torque End-effector desired force in Cartesian space, allowable range: { ±60, ±60, ±60, ±30, ±30, ±30 }, in N, N.m [out] ec Error code

setTorqueFilterCutOffFrequency()

template <unsigned short DoF> void RtMotionControlCobot<DoF>::setTorqueFilterCutOffFrequency (double frequency, error_code & ec) Set filtering parameters
Parameter [in] frequency Allowable range: 1–1,000 Hz [out] ec Error code

setFcCoor()
template <unsigned short DoF> void RtMotionControlCobot<DoF>::setFcCoor (const std::array< double, 16 > & frame, FrameType type, error_code & ec) Set the robot force control frame
Parameter [in] frame Transformation matrix of the force control frame in the flange frame [in] type Type, specifying the frame to be used as the force control task frame. Supported types: 1) World frame FrameType::world; 2) Tool frame FrameType::tool; 3) Path frame FrameType::path (the necessary process of the frame for force control tasks tracking the trajectory variation) [out] ec Error code

automaticErrorRecovery()
void rokae::MotionControl< MotionControlMode::RtCommand >::automaticErrorRecovery (error_code & ec) Automatically recover the robot after an error occurs.
Parameter [out] ec Error code

setRtNetworkTolerance()
void rokae::BaseCobot::setRtNetworkTolerance (unsigned percent, error_code & ec) Set the network delay threshold for sending real-time motion commands, i.e., the "packet loss threshold" in the RobotAssist - RCI settings interface. Set it before switching to RtCommand mode. Otherwise, it will not take effect.
Parameter [in] percent Allowable range: 0–100 [out] ec Error code

useRciClient()
void rokae::BaseCobot::useRciClient (bool use, error_code & ec) APIs compatible with the RCI client. After the motion control is set to real-time mode through SDK, the RCI client can no longer be used to control the robot. To use the original version, call this API after switching to non-real-time mode. Then enable RCI on RobotAssist to use the RCI client.
Parameter [in] use true - switch to the original version [out] ec Error code

hasMotionError()
bool rokae::MotionControl< MotionControlMode::RtCommand >::hasMotionError() noexcept; Whether there is any motion error in real-time mode
Return true - error report

8.3.4 Communication

getDI()
bool rokae::BaseRobot::getDI (unsigned int board, unsigned int port, error_code & ec) Query digital input signal value

Parameter	[in]	board IO board serial number
	[in]	port Signal port number
	[out]	ec Error code
Return	true-enable | false-disable	

getDO()
bool rokae::BaseRobot::getDO (unsigned int board, unsigned int port, error_code &ec)
Query digital output signal value

Parameter	[in]	board IO board serial number
	[in]	port Signal port number
	[out]	ec Error code
Return	true-enable | false-disable	

setDI()
void rokae::BaseRobot::setDI (unsigned int board, unsigned int port, bool state, error_code &ec)
Set the digital input signal. This can only be configured when the input simulation mode is enabled (see setSimulationMode()).

Parameter	[in]	board IO board serial number
	[in]	port Signal port number
	[in]	state true-enable | false-disable
	[out]	ec Error code

setDO()
void rokae::BaseRobot::setDO (unsigned int board, unsigned int port, bool state, error_code & ec)
Set digital output signal value

Parameter	[in]	board IO board serial number
	[in]	port Signal port number
	[in]	state true-enable | false-disable
	[out]	ec Error code

getAI()
double rokae::BaseRobot::getAI (unsigned board, unsigned port, error_code & ec)
Read analog input signal value

Parameter	[in]	board IO board serial number
	[in]	port Signal port number
	[out]	ec Error code
Return	Signal value	

setAO()
void rokae::BaseRobot::setAO (unsigned board, unsigned port, double value, error_code & ec)
Set analog output signal value

Parameter	[in] board IO board serial number [in] port Signal port number [in] value Output value [out] ec Error code

readRegister()
template<typename T > void rokae::BaseRobot::readRegister (const std::string & name, unsigned index, T & value, error_code & ec) Read register value. Read a single register, a register array, or a register array by index. To read the whole register array, the value is passed into the vector of the corresponding type, and the index value is ignored.
Template parameters T Read value type
Parameter [in] name Register name [in] index Read register array elements by index, starting from 0. An error is reported in the following two cases: 1) The index length exceeds the array length; 2) The register is not an array, but the index is greater than 0 [out] value Register value, types allowed: bool/int/float [out] ec Error code

writeRegister()
template<typename T > void rokae::BaseRobot::writeRegister (const std::string & name, unsigned index, T value, error_code & ec) Write register value. Write to a single register, a register array, or a register array element by index. To write the whole register array, the value is passed into the vector of the corresponding type, and the index value is ignored.
Template parameters T Written value type
Parameter [in] name Register name [in] index Array index starting from 0 An error is reported in the following two cases: 1) The index length exceeds the array length; 2) The register is not an array, but the index is greater than 0 [in] value Written value; allowable types: bool/int/float and std::vector<bool/int/float> [out] ec Error code

setxPanelVout()
void rokae::BaseCobot::setxPanelVout (xPanelOpt::Vout opt, error_code & ec) Set xPanel external power supply mode Note: The xPanel feature is only available for some models. An error code will be returned for unsupported models.
Parameter [in] opt Mode [out] ec Error code

setSimulationMode()
void rokae::BaseRobot::setSimulationMode(bool state, error_code &ec) Set simulation mode
Parameter [in] state true- enabled | false - disabled [out] ec Error code

getKeypadState()
KeyPadState rokae::BaseRobot::getKeypadState(error_code& ec) Get the state of end-effector keypads, and for unsupported models, return an error code
Parameter [out] ec Error code

Return	State of end-effector keypads. See the figure of the end-effector handle in the *xCore Control System User Manual* for keypad numbering.

setxPanelRS485 ()
void rokae::BaseCobot::setxPanelRS485(xPanelOpt::Vout opt, bool if_rs485, error_code& ec) To use the 485 communication function on CR and SR end-effectors, the end-effector's parameter configuration must be modified. This can be done through this API
Parameter [in] opt External power supply mode. 0: no output, 1: reserved, 2: 12 V, 3: 24 V [in] if_rs485 API working mode; indicates whether to enable end-effector 485 communication [out] ec Error code

XPRWModbusRTUReg ()
void rokae::BaseCobot::XPRWModbusRTUReg (int slave_addr, int fun_cmd, int reg_addr, std::string data_type, int num, std::vector<int>& data_array, bool if_crc_reverse, error_code& ec) Read and write modbus registers via the xPanel end-effector
Parameter [in] slave_addr Device address 0-65535 [in] fun_cmd Function code 0x03 0x04 0x06 0x10 [in] reg_addr register address 0-65535 [in] data_type supported data types: int32, int16, uint32, uint16 [in] num Number of registers in a continuous operation: 0–3. When the type is int16/uint16, the maximum is 3; for int32/uint32 or float, the maximum is 1. This parameter is invalid when the function code is 0x06 [in/out] data_array Array for sending or receiving data (non-const). For the function code of 0x06, only data [0] of this array is used, and the value of num is invalid. Except for the function code of 0x06, the size must match num [in] if_crc_reverse Whether to swap CRC verification high and low bytes. The default is false. Some manufacturers'end-effectors need to be reversed [out] ec Error code

XPRWModbusRTUCoil ()
void rokae::BaseCobot::XPRWModbusRTUCoil (int slave_addr, int fun_cmd, int coil_addr, int num, std::vector<int>& data_array, bool if_crc_reverse, error_code& ec) Read and write modbus coil or discrete input via the xPanel end-effector
Parameter [in] slave_addr Device address 0-65535 [in] fun_cmd Function code 0x01 0x02 0x05 0x0F [in] coil_addr coil or discrete input register address 0-65535 [in] num Number of consecutive coils or discrete inputs to read/write in one operation (0-48). This value is invalid for the function code of 0x05 [in/out] data_array Array for sending or receiving data (non-const). For the function code of 0x05, only data [0] of this array is used, and the value of num is invalid. Except for the function code of 0x05, the size must match num [in] if_crc_reverse Whether to swap CRC verification high and low bytes. The default is false. Some manufacturers'end-effectors need to be reversed [out] ec Error code

XPRS485SendData ()
void rokae::BaseCobot::XPRS485SendData(int send_byte, int rev_byte, const std::vector<uint8_t>& send_data, std::vector<uint8_t>& rev_data, error_code& ec) Transmit raw RTU protocol data directly through the xPanel end-effector
Parameter [in] send_byte Length of sent data 0-16 [in] rev_byte Length of received data 0-16 [in] send_data Sent byte data The array length needs to be consistent with the parameter of send_byte [out] rev_data Received byte data The array length needs to be consistent with the parameter of rev_byte [out] ec Error code

8.3.5 RL Project

projectsInfo()	
std::vector< RLProjectInfo > rokae::BaseRobot::projectsInfo (error_code & ec) Query the RL project name and task in the IPC	
Parameter	[out] ec Error code
Return	Project information list. An empty list will be returned if no project is created

loadProject()	
void rokae::BaseRobot::loadProject(const std::string & name, const std::vector< std::string > & tasks, error_code & ec) Load project	
Parameter	[in] name Project name [in] tasks Tasks to be performed. This parameter must be specified and cannot be empty. Otherwise, the project cannot be executed. [out] ec Error code

ppToMain()	
void rokae::BaseRobot:: ppToMain(error_code &ec) The program pointer jumps to main. After calling, wait for the controller to parse the project and then return. The blocking duration depends on the size of the project. The timeout period is set to 10 seconds.	
Parameter	[out] ec Error code The error code only provides limited information, which does not include errors such as "RL syntax error" and "variable that does not exist". Use queryControllerLog() to query error logs.

runProject()	
void rokae::BaseRobot::runProject (error_code &ec) Start running the currently loaded project	
Parameter	[out] ec Error code

pauseProject()	
void rokae::BaseRobot::pauseProject (error_code &ec) Pause running project	
Parameter	[out] ec Error code

setProjectRunningOpt()	
void rokae::BaseRobot::setProjectRunningOpt (double rate, bool loop, error_code &ec) Set the project's running rate and loop mode	
Parameter	[in] rate Running rate between 0.01 and 1 [in] loop true - loop | false - single cycle [out] ec Error code

toolsInfo()	
std::vector< WorkToolInfo > rokae::BaseRobot::toolsInfo (std::error_code &ec) Query tool information of the currently loaded project	
Parameter	[out] ec Error code
Return	Tool information list. If no project is loaded or no tool is created, the information of the default tool tool0 is returned.

wobjsInfo()

std::vector< WorkToolInfo > rokae::BaseRobot::wobjsInfo(std::error_code &ec)
Query work object information of the currently loaded project
Parameter [out] ec Error code
Return Work object information list. If no project is loaded or no work object is created, an empty vector is returned.

importProject()
std::string rokae::BaseRobot::importProject (const std::string &file_path, bool overwrite, error_code &ec)
Import the local RL project package into the controller. The function blocks until the import completes or fails
Parameter [in] file_path Path to the local .zip package file, its size within 10 M [in] overwrite Whether to overwrite files with the same name. true: overwrite; false: auto-rename [out] ec Error code
Return Project name (e.g., the name after auto-renaming or returning for renaming)

removeProject()
void rokae::BaseRobot::removeProject (const std::string &project_name, error_code &ec, bool remove_all = false)
Remove the RL project from the controller
Parameter [in] project_name Project name [in] remove_all Whether to delete all projects. The default is false. [out] ec Error code

importFile()
std::string rokae::BaseRobot::importFile (std::string src_file_path, std::string dest, bool overwrite, error_code &ec)
Import local files to the controller. The function blocks until the import completes or fails
Parameter [in] src_file_path Path to the local file, its size within 10 M [in] dest Target path 1) Transfer a single RL project .mod file: project/[project name]/[task name]</[mod file name]> 2) Transfer configuration files in .json/.xml/.sys format of the RL project: project/[project name]</[file name]>. Note: Configuration file names cannot be changed. For example, the task file must be named as "task.xml" [in] overwrite Overwrite existing files: true - overwrite | false - auto-rename. Auto-rename is only supported for .mod files. [out] ec Local file not found; invalid file format; transfer failed; invalid destination path, etc.
Return File name after successfully imported

removeFiles()
void rokae::BaseRobot::removeFiles (std::vector<std::string> file_path_list, error_code &ec)
Remove files from the controller. Note: Configuration files such as .xml and .json of projects cannot be deleted; they can only be replaced.
Parameter [in] file_path_list List of file paths. Individual file path is as follows: 1) Delete. mod files under a task of a certain project: project/[project name]/[task name]/[mod file name] 2) Delete a task of a certain project: project/[project name]/[task name] [out] ec Parameter format error or network error. No error is returned, as the project, task, or file is not found.

setToolInfo()
void rokae::BaseRobot:: setToolInfo (const WorkToolInfo &tool_info, error_code &ec)
Set global tool information, or create/set tool posture and load information in an RL project * @note Note: * 1) Global tools: The controller supports 16 global tools, with fixed names "g_tool_0" to "g_tool_15". * 2) RL project tools: There are several limitations when using. It not is recommended to set project tools via this API. Use global tools instead. Limitations include: a) A project must be loaded before configuring project tools. Any tool name not matching a global tool is considered a

project tool. b) If the tool does not be found, it will be created; if it exists, it will be modified. c) The project's tool configuration file must also be modified accordingly. Otherwise, RL commands may fail to parse the tool information properly, and data after setting will not be saved. * 3) Setting tool envelope information is currently not supported
Parameter [in] tool_info Tool information [out] ec Global tools generally do not fail to set up. Tools in the project may fail to be configured. For example, if the project's tool configuration file is pushed to the controller, but the project is not reloaded. Thus, an error code will be returned due to inconsistent tool configurations.

setWobjInfo()
void rokae::BaseRobot::setWobjInfo (const WorkToolInfo &wobj_info, error_code &ec) Set global work object information, or create/set work object posture and load information in an RL project
Note 1) Global work objects: The controller supports 16 global work objects, with fixed names "g_wobj_0" to "g_wobj_15". 2) RL project work objects: There are several limitations when using. It not is recommended to set project work objects via this API. Use global work objects instead. Limitations include: a) A project must be loaded before configuring project work objects. Any work object name not matching a global work object is considered a project work object. b) If the work object does not be found, it will be created; if it exists, it will be modified. c) The project's work object configuration file must also be modified accordingly. Otherwise, RL commands may fail to parse the tool information properly, and data after setting will not be saved. 3) Setting related user frames is not supported for now. The global work object frame defaults to "g_user_0", and the project work object defaults to "userframe0". **Parameter** [in] wobj_info Work object information [out] ec Similarly to tool interface configuration, global work objects generally do not fail to set up. However, work objects in the project may encounter setup failures.

8.3.6 Collaboration-related

enableDrag()
void rokae::BaseCobot::enableDrag(DragParameter::Space space, DragParameter::Type type, error_code & ec , bool enable_drag_button = false) Enable Drag mode
Parameter [in] space Drag space. Only free drag is supported for joint space drag. [in] type Drag type [out] ec Error code [in] enable_drag_button true - After enabling the drag function, you can directly drag the robot without holding the end-effector keypad

disableDrag()
void rokae:: BaseCobot::disableDrag (error_code & ec) Disable Drag mode
Parameter [out] ec Error code

startRecordPath()
void rokae::BaseCobot::startRecordPath(int duration, error_code &ec) Start path recording
Parameter [in] duration Path duration, in seconds, range: 1–1800. The duration is only used for range check. After the duration has elapsed, the controller will not stop recording. To stop recording, call stopRecordPath(). [out] ec Error code

stopRecordPath()
void rokae::BaseCobot::stopRecordPath (error_code & ec) Stop path recording. If recording is successful (with no error code), the path data will be saved in the cache.
Parameter [out] ec Error code

cancelRecordPath()
void rokae::BaseCobot::cancelRecordPath(error_code & ec)
Cancel recording. The cached path data will be deleted.
Parameter [out] ec Error code

saveRecordPath()
void rokae::BaseCobot::saveRecordPath (const std::string & name, error_code & ec, const std::string & saveAs = "")
Save recorded path
Parameter [in] name Path name [out] ec Error code [in] saveAs Rename (optional parameters) If a path has been recorded but not saved, save the path with this name. If no path is unsaved, rename the saved path "name" to "saveAs".

replayPath()
void rokae::BaseCobot::replayPath (const std::string &name, double rate, error_code & ec) Motion command - path replay Similar to other motion commands, after calling replayPath, you need to call moveStart to start the movement.
Parameter [in] name Name of the path to be replayed [in] rate Replay speed (< 3.0). 1 is the original path speed. Please note that when the playback speed is greater than 1, a drive following error might occur. [out] ec Error code

removePath()
void rokae: BaseCobot::removePath (const std::string &name, error_code & ec, bool removeAll = false) Delete saved path
Parameter [in] name Name of the path to be deleted [out] ec Error code If the path does not exist, the error code will not be set. [in] removeAll Whether to delete all paths. Optional parameter; default: no

queryPathLists()
std::vector< std::string > rokae::BaseCobot::queryPathLists (error_code &ec)
Query all saved path names
Parameter [out] ec Error code **Return** Path name list. An empty list will be returned if there is no path.

calibrateForceSensor()
template<unsigned short DoF> void rokae::Cobot<DoF>::calibrateForceSensor(bool all_axes, int axis_index, error_code &ec) Enable Drag mode
Parameter [in] all_axes true - calibration of all axes | false - single-axis calibration [in] axis_index Axis index, range [0, DoF). It only takes effect when a single axis is calibrated [out] ec Error code

8.3.7 Force control commands

forceControl()
template<unsigned short DoF>

ForceControl_T<DoF> rokae::Cobot<DoF>::forceControl ( )
Force control commands
Return ForceControl_T

getEndTorque()
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::getEndTorque (FrameType ref_type, std::array< double, DoF > & joint_torque_measured, std::array< double, DoF > & external_torque_measured, std::array< double, 3 > & cart_torque, std::array< double, 3 > & cart_force, error_code &ec) Get the current torque information
Parameter [in] ref_type Reference frame for relative torque: 1) FrameType::world - torque information of the end-effector in the world frame; 2) FrameType::flange - torque information of the end-effector in the flange plate; 3) FrameType::tool - torque information of the end-effector in the TCP point [out] joint_torque_measured Measuring forces on each axis [out] external_torque_measured External forces on each axis [out] cart_torque Cartesian spatial torque [X, Y, Z] in N.m [out] cart_force Cartesian space force [X, Y, Z] in N [out] ec Error code

fcInit()
void rokae::BaseForceControl::fcInit(FrameType frame_type, error_code &ec) Initialize the force control
Parameter [in] frame_type force control frame, supporting world/wobj/tool/base/flange. The tool/work object frame is set using setToolset() [out] ec Error code

fcStart()
void rokae::BaseForceControl::fcStart(error_code &ec) Start force control, and call after fcInit(). If motion commands need to be executed in the force control mode, they can be done after fcStart(). Note: If motion commands were issued via moveAppend() before fcStart(), but the motion had not started, those commands will be executed after fcStart.
Parameter [out] ec Error code

fcStop()
void rokae::BaseForceControl::fcStop(error_code &ec) Stop force control
Parameter [out] ec Error code

setControlType()
void rokae::BaseForceControl::setControlType(int type, error_code &ec) Set impedance control type
Parameter [in] type 0 - joint impedance | 1 - Cartesian impedance [out] ec Error code

setLoad()
void rokae::BaseForceControl::setLoad(const Load &load, error_code &ec) Set the load information used by the force control module, which can be called after fcStart().
Parameter [in] load Load [out] ec Error code

setJointStiffness()
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::setJointStiffness (const std::array< double, DoF > & stiffness,

error_code &ec) Set the joint impedance stiffness. The call takes effect after fcInit() As the maximum stiffness of each model is different, please refer to the SetJntCtrlStiffVec command instruction in the *xCore Control System Manual*
Parameter [in] stiffness Stiffness of each axis [out] ec Error code

setCartesianStiffness()
void rokae::BaseForceControl::setCartesianStiffness(const std::array<double, 6> &stiffness, error_code &ec) Set the Cartesian impedance stiffness. The call takes effect after fcInit() As the maximum stiffness of each model is different, please refer to the SetCartCtrlStiffVec command instruction in the *xCore Control System Manual*
Parameter [in] stiffness In order: impedance force stiffness in X, Y, and Z directions [N/m], and impedance torque stiffness in X, Y, and Z directions [N.m/rad] [out] ec Error code

setCartesianNullspaceStiffness()
void rokae::BaseForceControl::setCartesianNullspaceStiffness(double stiffness, error_code &ec) Set the Cartesian null-space impedance stiffness. The call takes effect after fcInit()
Parameter [in] stiffness Range [0,4]. If stiffness is greater than 4, it will be set to 4 by default, in N.m/rad [out] ec Error code

setJointDesiredTorque()
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::setJointDesiredTorque(const std::array<double, DoF> &torque, error_code &ec) Set the desired torque of the joint. Call after fcStart()
Parameter [in] torque Torque value, range [-30,30], in N.m [out] ec Error code

setCartesianDesiredForce()
void rokae::BaseForceControl::setCartesianDesiredForce(const std::array<double, 6> &value, error_code &ec) Set the desired Cartesian force/torque. Call after fcStart()
Parameter [in] value In order: desired Cartesian force in X, Y, and Z directions, range [-60, 60], in N; desired Cartesian torque in X, Y, and Z directions, range [-10, 10], in N.m [out] ec Error code

setSineOverlay()
void rokae::BaseForceControl::setSineOverlay(int line_dir, double amplify, double frequency, double phase, double bias, error_code &ec) Set the sine overlay rotating around a single axis. This call takes effect after setting the impedance control type to Cartesian impedance (i.e., setControlType(1)) and before startOverlay(). The maximum overlay amplitude and overlay frequency vary by robot model. Please refer to the SetSineOverlay command instruction in the *xCore Control System Manual*
Parameter [in] line_dir Overlay reference axis: 0 - X | 1 - Y | 2 - Z [in] amplify Overlay amplitude, in N.m [in] frequency Overlay frequency, in Hz [in] phase Overlay phase, range [0, PI], in rad [in] bias Overlay offset, range [0, 10], n N.m [out] ec Error code

setLissajousOverlay()
void rokae::BaseForceControl::setLissajousOverlay(int plane, double amplify_one, double frequency_one, double amplify_two, double frequency_two, double phase_diff, error_code &ec) Set the Lissajous overlay within a plane This call takes effect after setting the impedance control type to Cartesian impedance (i.e., setControlType(1)) and before startOverlay().
Parameter [in] plane Overlay reference plane: 0 - XY | 1 - XZ | 2 - YZ

	[in] amplify_one Overlay direction I amplitude, range [0, 20], in N.m [in] frequency_one Overlay direction I frequency, range [0, 5], in Hz [in] amplify_two Overlay direction II amplitude, range [0, 20], in N.m [in] frequency_two Overlay direction II frequency, range [0, 5], in Hz [in] phase_diff Phase deviation between overlays in two directions, range [0, PI], in rad [out] ec Error code

startOverlay()	
void rokae::BaseForceControl::startOverlay(error_code &ec) Start the overlays. The call takes effect after fcStart() The overlay is a superposition of the previously configured setSineOverlay() or setLissajousOverlay()	
Parameter	[out] ec Error code

stopOverlay()	
void rokae::BaseForceControl::stopOverlay(error_code &ec) Stop the overlays	
Parameter	[out] ec Error code

pauseOverlay()	
void rokae::BaseForceControl::pauseOverlay(error_code &ec) Pause the overlay The call takes effect after startOverlay()	
Parameter	[out] ec Error code

restartOverlay()	
void rokae::BaseForceControl::restartOverlay(error_code &ec) Restart the paused overlays. The call takes effect after pauseOverlay().	
Parameter	[out] ec Error code

setForceCondition()	
void rokae::BaseForceControl::setForceCondition(const std::array<double, 6> &range, bool isInside, double timeout, error_code &ec) Define termination conditions related to contact force	
Parameter	[in] range Force limitations in all directions { X_min, X_max, Y_min, Y_max, Z_min, Z_max }, in N. When setting the lower limit, a negative value represents the maximum value in the negative direction; when setting the upper limit, a negative value represents the minimum value in the negative direction. [in] isInside true - Stop waiting when exceeding the restrictions; false - Stop waiting when the restrictions are met [in] timeout Timeout, range [1, 600], in s [out] ec Error code

setTorqueCondition()	
void rokae::BaseForceControl::setTorqueCondition(const std::array<double, 6> &range, bool isInside, double timeout, error_code &ec) Define termination conditions related to contact torque	
Parameter	[in] range Torque limitations in all directions { X_min, X_max, Y_min, Y_max, Z_min, Z_max }, in N. When setting the lower limit, a negative value represents the maximum value in the negative direction; when setting the upper limit, a negative value represents the minimum value in the negative direction. [in] isInside true - Stop waiting when exceeding the restrictions; false - Stop waiting when the restrictions are met [in] timeout Timeout, range [1, 600], in s [out] ec Error code

setPoseBoxCondition()	
void rokae::BaseForceControl::setPoseBoxCondition(const Frame &supervising_frame, const std::array<double, 6> &box, bool isInside, double timeout, error_code &ec) Define termination conditions related to contact location	
Parameter	[in] supervising_frame Reference frame where the cuboid is located, relative to the external work object frame The external work object frame is set through setToolset() (Toolset::ref)

	[in] box Defining a cuboid { X_start, X_end, Y_start, Y_end, Z_start, Z_end }, in m [in] isInside true - Stop waiting when exceeding the restrictions; false - Stop waiting when the restrictions are met [in] timeout Timeout, range [1, 600], in s [out] ec Error code

waitCondition()	
void rokae::BaseForceControl::waitCondition(error_code &ec) Activate the previously set termination conditions and wait until they are met or a timeout occurs	
Parameter	[out] ec Error code

fcMonitor()	
void rokae::BaseForceControl::fcMonitor(bool enable, error_code &ec) Activate/deactivate force control module protection monitoring. The configured monitoring parameters do not take effect immediately. They become active after calling fcMonitor(true) and remain in effect until fcMotion(false) is called to deactivate them. After completion, the protection thresholds revert to their default values, meaning the protection effect remains active. The user-defined parameters are no longer in effect after monitoring is disabled.	
Parameter	[in]enabletrue - On | false - Off [out] ec Error code
Reference	setCartesianMaxVel() setJointMaxVel() setJointMaxMomentum() setJointMaxEnergy()

setJointMaxVel()	
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::setJointMaxVel(const std::array<double, DoF> &velocity, error_code &ec) Set the maximum axial velocity in force control mode	
Parameter	[in] velocity Axis velocity [rad/s], range >=0 [out] ec Error code

setJointMaxMomentum()	
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::setJointMaxMomentum(const std::array<double, DoF> &momentum, error_code &ec) Set the maximum axial momentum in force control mode. Calculation method: F*t. It can be understood as momentum. F is the reading from the torque sensor, and t is the control cycle. If the momentum exceeds the threshold for more than 30 cycles, the protection is triggered	
Parameter	[in] momentum Momentum [N·s], range >=0 [out] ec Error code

setJointMaxEnergy()	
template<unsigned short DoF> void rokae::ForceControl_T<DoF>::setJointMaxEnergy(const std::array<double, DoF> &energy, error_code &ec) Set the maximum axial energy in force control mode. Calculation method: F*v. It can be understood as power. F is the reading from the torque sensor, and t is the joint speed. If the energy exceeds the threshold for more than 30 cycles, the protection is triggered	
Parameter	[in] energy Energy [N·rad/s], range >=0 [out] ec Error code

setCartesianMaxVel()	
void rokae::BaseForceControl::setCartesianMaxVel(const std::array<double, 6> &velocity, error_code &ec) Set the maximum speed of the robotic arm end-effector in the base frame in force control mode	
Parameter	[in] velocity In order: X, Y, and Z [m/s], A, B, and C [rad/s], range >=0 [out] ec Error code

8.3.8 Path planning

CartMotionGenerator()

rokae::CartMotionGenerator::CartMotionGenerator (double speed_factor, double s_goal) Generate a joint space trajectory according to the target joint position and speed factor, which can be used to return to zero pose or reach the specified position.
Parameter [in] speed_factor Speed factor [0, 1]. Final speed/acceleration = maximum speed/acceleration * speed factor [in] s_goal Target joint angle

calculateSynchronizedValues()
void rokae::CartMotionGenerator::calculateSynchronizedValues (double s_init) S-curve planning. Arc length S-curve planning for Cartesian space.
Parameter [in] s_init Initial arc length

calculateDesiredValues()
bool rokae::CartMotionGenerator::calculateDesiredValues (double t, double * delta_s_d) const Calculate arc length s at time t
Parameter [in] t Time from the start of planning, in s [out] delta_s_d Calculated result **Return** false: motion planning not completed | true: motion planning completed

getTime()
double rokae::CartMotionGenerator::getTime ( ) Get total motion time
Return Motion time, in seconds

setMax()
void rokae::CartMotionGenerator::setMax (double ds_max, double dds_max_start, double dds_max_end) Set Cartesian space motion parameters
Parameter [in] ds_max Maximum speed [m/s], default value 1.0m/s. [in] dds_max_start Maximum initial acceleration [m/s^2], default value 2.5 m/s2 [in] dds_max_end Maximum final acceleration [m/s^2], default value 2.5 m/s2

JointMotionGenerator()
rokae::JointMotionGenerator::JointMotionGenerator (double speed_factor, std::array< double, 7 > q_goal) Generate a joint space trajectory according to the target joint position and speed factor, which can be used to return to zero pose or reach the specified position.
Parameter [in] speed_factor Speed factor [0, 1]. Final speed of each axis/acceleration = maximum speed of joint space/acceleration * speed factor [in] q_goal Target joint angle [rad]

calculateSynchronizedValues()
void rokae::JointMotionGenerator::calculateSynchronizedValues (const std::array< double, 7 > & q_init) S-curve planning
Parameter [in] q_init Initial joint angle

calculateDesiredValues()
bool rokae::JointMotionGenerator::calculateDesiredValues(double t, std::array< double, 7 > & delta_q_d) const Calculate joint angle increment at time t
Parameter [in] t Time point, in s [out] delta_q_d Calculated result **Return** false: motion planning not completed | true: motion planning completed

getTime()
double rokae::JointMotionGenerator::getTime ( ) Get total motion time
Return Motion time, in seconds

setMax()
void rokae::JointMotionGenerator::setMax (const std::array< double, 7 > & dq_max, const std::array< double, 7 > & ddq_max_start, const std::array< double, 7 > & ddq_max_end) Set motion parameters for joint space S-speed planning
Parameter [in] dq_max Maximum speed [rad/s], default values J1–J4 1.0 rad/s, J5–J7 1.25 rad/s [in] ddq_max_start Maximum initial acceleration [rad/s^2], default value 2.5 rad/s^2 [in] ddq_max_end Maximum final acceleration [rad/s^2], default value 2.5 rad/s^2

FollowPosition()
template<unsigned short DoF> FollowPosition<DoF>::FollowPosition(Cobot<DoF>& robot, xMateModel<DoF>& model, const Eigen::Transform<double, 3, Eigen::Isometry>& endInFlange) Follow position, which can be Cartesian posture or joint angle.
Parameter robot rokae::Robot model rokae::xMateModel, getting it via robot.model() [in] endInFlange: posture of the end-effector in flange

FollowPosition()
template<unsigned short DoF> FollowPosition<DoF>::FollowPosition() Follow position, which can be Cartesian posture or joint angle. It is the default constructor, which must be initialized by calling init().

init()
template<unsigned short DoF> void FollowPosition<DoF>::init(Cobot<DoF>& robot, XMateModel<DoF>& model) Initialize the FollowPosition.
Parameter robot rokae::Robot model rokae::xMateModel, getting it via robot.model()

start() [1/2]
template<unsigned short DoF> void FollowPosition<DoF>::start(const std::array<double, DoF> &jnt_desire) Start FollowPosition − Cartesian posture. This interface is non-blocking.
Parameter [in] bMe_desire: desired target posture, which is the frame of the end-effector in base, i.e. TCP posture.

start() [2/2]
template<unsigned short DoF> void FollowPosition<DoF>::start(const Eigen::Transform<double, 3, Eigen::Isometry>& bMe_desire) Start FollowPosition − joint angle. This interface is non-blocking.
Parameter [in] jnt_desire: desired joint angle

stop()
template<unsigned short DoF> void FollowPosition<DoF>::stop() Stop FollowPosition.

update() [1/2]	
template<unsigned short DoF> void FollowPosition<DoF>::update(const Eigen::Transform<double, 3, Eigen::Isometry>& bMe_desire); Update the desired posture.	
Parameter	[in] bMe_desire: frame of the end-effector in base, i.e. TCP posture

update() [2/2]	
template<unsigned short DoF> void FollowPosition<DoF>::update(const std::array<double, DoF>& jnt_desired); Update the desired target joint angle.	
Parameter	[in] jnt_desired: desired joint angle in rad

setScale()	
template<unsigned short DoF> void FollowPosition<DoF>::setScale(double scale); Set the speed scale, which can be adjusted at the FollowPosition. The final speed is limited by the maximum value, and the current value cannot be changed. The maximum speeds of each axis are: 120.0, 120.0, 180.0, 180.0, 200.0, 200.0, 200.0 [°/s]; The maximum acceleration of each axis is 500.0 [°/s^2]	
Parameter	[in] scale Speed scale, by default of 0.5

8.3.9 xMate Model Library for Kinematics and Dynamics Calculation

model()	
template<unsigned short DoF> XMateModel< DoF > rokae::Cobot< DoF >::model ( ) Get xMate model	
Return	XMateModel
Exceptions	ExecutionException Failed to read model parameters from the controller

getCartAcc()	
template<unsigned short DoF> std::array< double, 6 > rokae::XMateModel< DoF >::getCartAcc (const std::array< double, DoF > & jntPos, const std::array< double, DoF > & jntVel, const std::array< double, DoF > & jntAcc, SegmentFrame nr = SegmentFrame::flange) Get Cartesian space acceleration	
Parameter	[in] jntPos Cartesian space joint angle to be calculated [in] jntVel Cartesian space joint angular velocity to be calculated [in] jntAcc Cartesian space joint angular acceleration to be calculated [in] nr Specified frame
Return	Calculated result

getCartPose()	
template<unsigned short DoF> std::array< double, 16 > rokae::XMateModel< DoF >::getCartPose (const std::array< double, DoF > & jntPos, SegmentFrame nr = SegmentFrame::flange) Get Cartesian space position	
Parameter	[in] jntPos Cartesian posture joint angle to be calculated [in] nr Specified frame, with a default value of flange
Return	Vectorized 4 × 4 posture matrix, row-major

getCartVel()	
template<unsigned short DoF> std::array< double, 6 > rokae::XMateModel< DoF >::getCartVel (const std::array< double, DoF > & jntPos, const std::array< double, DoF > & jntVel, SegmentFrame nr = SegmentFrame::flange)	

Get Cartesian space velocity

Parameter	[in] jntPos Cartesian space joint angle to be calculated
	[in] jntVel Cartesian space joint angular velocity to be calculated
	[in] nr Specified frame, with a default value of flange
Return	Calculated result

getJointAcc()

template<unsigned short DoF>
std::array< double, DoF > rokae::XMateModel< DoF >::getJointAcc(const std::array< double, 6 > & cartAcc,
const std::array< double, DoF > & jntPos,
const std::array< double, DoF > & jntVel)
Inverse kinematics for joint space acceleration

Parameter	[in] cartAcc Flange Cartesian space acceleration
	[in] jntPos Current joint angle
	[in] jntVel Current joint angular velocity
Return	Calculated result

getJointPos()

template<unsigned short DoF>
int rokae::XMateModel< DoF >::getJointPos (const std::array< double, 16 > & cartPos,
double elbow,
const std::array< double, DoF > & jntInit,
std::array< double, DoF > & jntPos)
Inverse kinematics for joint space position. One posture may correspond to multiple joint angles. The solution closest to q_init should be selected as q_out.

Parameter	[in] cartPos Flange Cartesian space posture
	[in] elbow Elbow
	[in] jntInit Initial joint angle
	[out] jntPos Joint space position
Return	Inverse kinematics calculation results – 1) -1, -2, -3: No solution. The cause is that cartPos is beyond the robot's workspace; 2) -4, -5: Large difference between jntPos and jntInit. jntInit is generally considered representing the current robot position. The difference between jntPos and jntInit can be regarded as equivalent motor speed. If it exceeds the rated speed of the robot axis, -4 or -5 is returned; 3) -6, -7: jntPos exceeds the soft limit; 4) -8: It indicates robot singularity.

getJointVel()

template<unsigned short DoF>
std::array< double, DoF > rokae::XMateModel< DoF >::getJointVel (const std::array< double, 6 > & cartVel,
const std::array< double, DoF > & jntPos)
Inverse kinematics for joint space velocity

Parameter	[in] cartVel Flange Cartesian space velocity
	[in] jntPos Current joint angle
Return	Calculated result

getTorqueNoFriction()

template<unsigned short DoF>
void rokae::XMateModel< DoF >::getTorqueNoFriction (const std::array< double, DoF > & jntPos,
const std::array< double, DoF > & jntVel,
const std::array< double, DoF > & jntAcc,
std::array< double, DoF > & trq_full,
std::array< double, DoF > & trq_inertia,
std::array< double, DoF > & trq_coriolis,
std::array< double, DoF > & trq_gravity)
Calculate joint torque without friction by model. Calculated result, in N.m. If there is a load, set load parameters first using setLoad().
The original getTorque() and getTorqueWithFriction are no longer supported, and this API is used to calculate joint torque

Parameter	[in] jntPos Joint angle
	[in] jntVel Joint angular velocity

	[in] jntAcc Joint angular acceleration
	[out] trq_full Total joint torque
	[out] trq_inertia Centrifugal force
	[out] trq_coriolis Coriolis force
	[out] trq_gravity Gravitational torque

jacobian() [1/2]	
template<unsigned short DoF> std::array< double, DoF *6 > rokae::XMateModel< DoF >::jacobian(const std::array< double, DoF > & jntPos, const std::array< double, 16 > & f_t_ee, const std::array< double, 16 > & ee_t_k, SegmentFrame nr = SegmentFrame::flange) Get Jacobian matrices of the specified coordinate system in the base coordinate system, row-major	
Parameter	[in] jntPos Joint angle
	[in] f_t_ee End-effector posture in the flange
	[in] ee_t_k Posture of the stiffness frame in the end-effector.
	[in] nr Specified frame
Return	Calculated result

jacobian() [2/2]	
template<unsigned short DoF> std::array< double, DoF *6 > rokae::XMateModel< DoF >::jacobian(const std::array< double, DoF > & jntPos, SegmentFrame nr = SegmentFrame::flange) Get Jacobian matrices of the specified coordinate system in the base coordinate system, row-major	
Parameter	[in] jntPos Joint angle
	[in] nr Specified frame
Return	Calculated result

setLoad()	
template<unsigned short DoF> void rokae::XMateModel< DoF >::setLoad (double mass, const std::array< double, 3 > & cog, const std::array< double, 3 > & inertia) Set load parameters. The parameters are used for calculation only and will not be sent to the robot controller. After setting, the results of dynamics calculation will change accordingly.	
Parameter	[in] mass Mass
	[in] cog Center of mass, in m
	[in] inertia Inertia
Return	Refer to Load

setTcpCoor()	
template<unsigned short DoF> void rokae::XMateModel< DoF >::setTcpCoor (const std::array< double, 16 > & f_t_ee, const std::array< double, 16 > & ee_t_k) Set TCP tools. The parameters are used for calculation only and will not be sent to the robot controller. After TCP is set, the forward/inverse kinematics results and input parameters will change accordingly.	
Parameter	[in] f_t_ee End-effector posture in the flange
	[in] ee_t_k Posture of the stiffness frame in the end-effector

8.3.10 Others

degToRad() [1/2]
template<size_t S> static std::array< double, S > rokae::Utils::degToRad (const std::array< double, S > & degrees) Convert from degrees to radians for arrays

Parameter	[in] degrees Degrees
	rad

degToRad() [2/2]

static double rokae::Utils::degToRad (double degrees)
Convert degrees to radians

Parameter	[in] degrees Degrees
Return	rad

radToDeg() [1/3]

template<size_t S>
static std::array< double, S > rokae::Utils::radToDeg (const std::array< double, S > & rad)
Convert from radians to degrees for arrays

Parameter	[in] rad Radian
Return	degree

radToDeg() [2/3]

static std::vector< double > rokae::Utils::radToDeg (const std::vector<double> & rad)
Convert from radians to degrees for arrays

Parameter	[in] rad Radian
Return	degree

radToDeg() [3/3]

static double rokae::Utils::radToDeg (double rad)
Convert radians to degrees

Parameter	[in] rad Radian
Return	degree

arrayToTransMatrix()

static void rokae::Utils::arrayToTransMatrix (const std::array< double, 16 > & array,
Eigen::Matrix3d & rot,
Eigen::Vector3d & trans)
Convert arrays to transformation matrices

Parameter	[in] array Array, row-major
	[out] rot Rotation matrix
	[out] trans Translation vector

transMatrixToArray()

static void rokae::Utils::transMatrixToArray (const Eigen::Matrix3d & rot,
const Eigen::Vector3d & trans,
std::array< double, 16 > &array)
Convert transformation matrices to arrays

Parameter	[in] rot Rotation matrix
	[in] trans Translation vector
	[out] array Conversion result, row-major

eulerToMatrix()

static void rokae::Utils::eulerToMatrix (const Eigen::Vector3d & euler,
Eigen::Matrix3d & matrix)
Convert Euler angles to rotation matrices

Parameter	[in] euler Euler angle, order [z, y, x], in rad

	[out] matrix Rotation matrix

postureToTransArray()	
static void rokae::Utils::postureToTransArray(const std::array<double, 6> &xyz_abc, std::array<double, 16> &transMatrix) Transform the posture array {X, Y, Z, RX, RY, RZ} into the row-major homogeneous transformation matrix.	
Parameter	[in] xyz_abc: input posture {X, Y, Z, RX, RY, RZ} [out] transMatrix: transformation result

transArrayToPosture ()	
static void rokae::Utils::transArrayToPosture(const std::array<double, 16> &transMatrix, std::array<double, 6> &xyz_abc) Transform the row-major homogeneous transformation matrix into the array {X, Y, Z, RX, RY, RZ}.	
Parameter	[in] transMatrix: row-major homogeneous transformation matrix [out] xyz_abc: transformation result

transMatrixToArray_all ()	
static void rokae::Utils::transMatrixToArray_all(const Eigen::Matrix4d& R ,std::array<double, 16>& array) Convert transformation matrices to arrays	
Parameter	[in] R Transformation matrix [out] array Conversion result, row-major

arrayToTransMatrix_all()	
static void rokae::Utils::arrayToTransMatrix_all(const std::array<double, 16>& array, Eigen::Matrix4d& R) Convert transformation matrices to arrays	
Parameter	[in] array Array, row-major [out] R 4*4 transformation matrix

quaternionToEuler()	
static std::array<double, 3> rokae::Utils::quaternionToEuler (double w, double x, double y, double z) Convert quaternions to Euler angles	
Parameter	[in] w Q1 [in] x Q2 [in] y Q3 [in] z Q4
Return	{ Rx, Ry, Rz }

eulerToQuaternion()	
static std::array<double, 4> rokae::Utils::eulerToQuaternion (const std::array<double, 3> &rpy) Convert Euler angles to quaternions	
Parameter	[in] rpy Euler angle, in rad
Return	Quaternions { Q1, Q2, Q3, Q4 }

toolsetCalcPos()	
static void rokae::Utils::toolsetCalcPos(const Toolset &tool_set, std::array<double, 16> &ref_trans, std::array<double, 16> &end_trans) Convert Toolset to tools and work objects	
Parameter	[in] tool_set Tool & work object [out] ref_trans External frame transformation matrix [out] end_trans End-effector frame transformation matrix

EndInRefToFlanInBase()	
static std::array<double, 6> rokae::Utils::EndInRefToFlanInBase(const std::array<double, 6>& base_in_world, const Toolset &tool_set, const std::array<double, 6> &end_in_ref) Frame transformation: Convert the coordinate of the end-effector in the external reference frame into that of the flange in the base frame	
Parameter	[in] base_in_world Base frame, in world frame [in] tool_set Tool & work object set [in] end_in_ref Coordinate of end-effector in the (external) reference frame
Return	Coordinate of the flange in base frame

FlanInBaseToEndInRef ()	
static std::array<double, 6> rokae::Utils::FlanInBaseToEndInRef(const std::array<double, 6>& base_in_world, const Toolset &tool_set, const std::array<double, 6> flan_in_base) Frame transformation: Convert the coordinate of the end-effector in the external reference frame into that of the flange in the base frame	
Parameter	[in] base_in_world Base frame, in world frame [in] tool_set Tool & work object set [in] flan_in_base Coordinate of the flange in base frame
Return	Coordinate of the end-effector in the (external) reference frame

ROKAE

Light-Weight Robot Expert

400-010-8700

Beijing Headquarters: Floor 7, Block A, Haiqing Building, No. 6 Agriculture Science Academy West Road, Haidian District, Beijing
Shangdong Company: No. 888 Hengfeng Rd., Electromechanical Industrial Park, Zhongxindian Town, Zoucheng, Jining, Shandong
Suzhou Company: 1-A1F, Creative Industrial Park, No. 328 Xinghu Street, Suzhou Industrial Park, Suzhou, Jiangsu
Shenzhen Company: 1st Floor, Building 10, COFCO Fuan Robot Intelligent Manufacturing Industrial Park, Baoan District, Shenzhen