Feel U

新一代柔性协作机器人

ROKAE CONTROL INTERFACE

丰富的二次开发接口；

支持位置控制，阻抗控制，直接力矩控制；

1KHz控制频率；

运动学与动力学计算接口。

本手册中包含的信息如有变更，恕不另行通知，且不应视为珞石的承诺。珞石对本手册中可能出现的错误概不负责。

除本手册中有明确陈述之外，本手册中的任何内容不应解释为珞石对个人损失、财产损坏或具体适用性等做出的任何担保或保证。

珞石对因使用本手册及其中所述产品而引起的意外或间接伤害概不负责。

可通过联系珞石技术支持工程师以获取此手册的纸质复印件。

目录

RCI 用户使用手册

一 RCI-SDK 介绍

RCI 是 ROKAE Control Interface 的首字母缩写，使用 C++语言编写而成，包含了一系列底层控制接口，科研或二次开发用户可以使用该软件包实现最高达 1KHz 的实时控制，用于算法验证以及新应用的开发。文档第二章介绍了软件的环境配置。第三章列出了软件的主要功能和接口。第四章给出了机器人本体参数、运动约束条件等。RCI_SDK 版本：V0.3.15。

二 软件使用环境配置

2.1 PC 操作系统

RCI 目前运行在 Linux 和 Windows 平台，为保障实时性，推荐使用 Linux 搭配实时内核使用。

Linux with PREEMPT_RT patched kernel ：推荐使用 Ubuntu16.04 with 4.14.12-rt10 kernel。

2.2 Linux 实时环境配置

RCI 需运行在实时环境中，实时环境配置有以下步骤：

1) 安装依赖

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

2) 下载实时内核补丁

通过 uname -r 命令可以知道本机正在使用的内核；

通过网站 https://www.kernel.org/pub/linux/kernel/projects/rt/查找离现在内核版本最接近的 kernal；

下载文件：

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

如果国内网速很慢可以直接手动从网站上下载或者找其他镜像源；

解压：

xz -d linux-4.14.12.tar.xz

xz -d patch-4.14.12-rt10.patch.xz

检查 sign 文件完整性

gpg2 --verify linux-4.14.12.tar.sign 会得到类似于如下的信息：

$ 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

记下 ID 6092693E 执行：

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

同理对于 patch 文件，执行相同的操作。

下载完成 server key 后再次验证，若得到如下信息就说明是正确的。

$ 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

同理验证一下 patch 文件。

3) 编译内核

解压：

tar xf linux-4.14.12.tar

cd linux-4.14.12

patch -p1 < ../patch-4.14.12-rt10.patch

配置内核：

make oldconfig

出现以下信息：

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)

选 5 然后一直 enter。

开始编译：

fakeroot make -j4 deb-pkg

dpkg 安装：

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

4) 验证是否安装成功

重启一下，进 ubuntu 高级选项，可以看到你安装的内核。 选择新安装的内核进入后，通过 uname -r 查看对应内核版本，如果版本正确，/sys/kernel/realtime 里内容是 1。

2.3 Linux 环境下 RCI 编译过程

1) 解压 SDK 包；
2) cd rci_client/build/
3) rm -rf *
4) cmake ..
5) make

生成的可执行文件在 build/examples 目录下。执行程序时需要添加 root 权限。

2.4 Windows 环境下 RCI 编译过程

1) 解压 SDK 包；
2) 打开 example 目录下的 vs 工程，编译运行

2.5 网络配置

使用 RCI 之前需要对网络进行配置，将用户 PC 与机器人连接起来。

机器人配置有 2 个网口，一个是外网口，一个是直连网口。直连网口默认静态 IP 地址是 192.168.0.160。连接机器人有两种方式。

连接方式 1：机器人与用户 PC 采用网线直连的方式连接。如果用户工控机与机器人不处于同一个网段，需要配置用户 PC 的 IP 使其与机器人静态 IP 地址处于同一个网段，例如 192.168.0.22。

连接方式 2：机器人外网口连接路由器或者交换机，用户 PC 也连接路由器或者交换机，两者处于同一局域网。

用户打开 HMI，在 RCI 界面将 IP 设置成用户 PC 的 IP，端口号设置成 1337，按下使能开关即可连接机器人。

推荐使用方式 1 进行连接，连接方式 2 可能会造成机器人运动不稳定现象。

2.6 RCI 自启动

开启 RCI 自启动功能后，可以脱离 HMI 对机器人进行控制。开启 RCI 自启动只需要将 HMI 上的 RCI 使能开关打开即可，机器人重启后会保持 RCI 打开状态，并自动切换成自动模式。

三 软件接口功能

通常将使用 RCI 的用户工控机视为客户端，机器人控制器作为服务端。

RCI 提供了使用简单的网络通信与控制功能接口：

1) 处理非实时命令。
2) 处理实时命令。
3) 获取机器人状态和回调机制。
4) 提供计算机器人运动学与动力学的 Model 库。
5) S 轨迹规划接口。

3.1 非实时命令

非实时命令也叫 TCP 命令，主要功能是：设置运动开始前的运动参数。TCP 命令有：

3.1.1 startMove

说明	运动开始前指定运动和控制模式
定义	startMove(ControllerMode，MotionGeneratorMode)；

ControllerMode

5 种控制器模式：

轴空间位置控制

笛卡尔空间位置控制

轴空间阻抗控制

笛卡尔空间阻抗控制

直接力矩控制

MotionGeneratorMode

2 种运动模式：

轴空间运动

笛卡尔空间运动

3.1.2 stopMove

说明

停止运动，停止收发 UDP 数据。

定义

stopMove()；

3.1.3 setCollisionBehavior

说明

设置碰撞检测阈值

定义

setCollisionBehavior(upper_torque_thresholds_nominal)；

upper_torque_thresholds_nominal

碰撞检测阈值，各轴最大值：（50，50，40，30，15，10，10）

3.1.4 setCartesianLimit

说明

设置安全区域

定义

setCartesianLimit(object_world_size，object_frame，object_activation)；

object_world_size

安全区域长宽高，对应 XYZ，单位：m

object_frame

安全区域中心位姿

object_activation

bool 变量，是否开启安全区域

3.1.5 setJointImpedance

说明

设置轴空间阻抗系数

定义

setJointImpedance(K_theta)；

K_theta

轴空间阻抗系数：max={{ 1500，1500，1500，1500，100，100，100 }}

3.1.6 setCartesianImpedance

说明

设置笛卡尔空间阻抗系数

定义

setCartesianImpedance(K_x)；

K_x

笛卡尔空间阻抗系数：max={{ 1500，1500，1500，100，100，100 }}

3.1.7 setCoor

说明

设置工具 TCP

定义

setCoor(F_T_EE);

F_T_EE

工具 TCP 相对于机器人末端法兰的位姿

3.1.8 setLoad

说明

设置负载

定义

setLoad(load_mass， load_center，load_inertia);

load_mass

负载质量，单位：kg

load_center

负载质心，单位：m

load_inertia

负载惯量，kg·m²

3.1.9 setFilters

说明

设置滤波截至频率

定义

setFilters(joint_position_filter_frequency，cartesian_position_filter_frequency,
torque_filter_frequency)；

joint_position_filter_frequency

轴空间位置滤波截至频率，单位：Hz

cartesian_position_filter_frequency

笛卡尔空间位姿滤波截至频率，单位：Hz

torque_filter_frequency

关节力矩滤波截至频率，单位：Hz

3.1.10 setCartImpDesiredTau

说明

设置笛卡尔空间阻抗期望力

定义

setCartImpDesiredTau(tau);

tau

笛卡尔空间阻抗期望力，例如 tau={{ 0，0，10，0，0，0}}，单位：N，N·m。

3.1.11 setMotorPower

说明

机器人上电或者下电

定义

setMotorPower(int motor_state);

motor_state

机器人上电状态，1 代表使机器人上电，0 代表使机器人下电。

3.1.12 getMotorState

说明

获取机器人上电状态，返回值：0 代表机器人处于下电状态，1 代表电机处于上电状态；

定义

getMotorState ( );

3.1.13 setDO

说明

设置 IO OUTPUT 信号

定义

setDO(DOSIGNAL DO_signal, bool state);

DO_signal

DO 端口号，分别对应{DO0_0, DO0_1, DO0_2, DO0_3, DO1_0, DO1_1}。

state

1 代表高电平输出，0 代表低电平输出。

3.1.14 getDI

说明

获取 IO INPUT 信号，返回值，true 代表高电平，false 代表低电平。

定义

getDI(DISIGNAL DI_signal)；

DI_signalS

DI 端口号，分别对应{DI0_0, DI0_1, DI0_2, DI0_3, DI1_0, DI1_1}。

3.2 实时数据

实时数据通过 UDP 的方式来发送，主要有 RCI 发送给机器人的数据 RobotCommand 和机器人给 RCI 发送的数据 RobotState。

3.2.1 RobotCommand

说明	
	RCI 发送给机器人数据结构

定义

```
struct MotionCommand {
        std::array<double, 7> q_c；//关节角度
        std::array<double, 16> toolTobase_pos_c；//末端位姿，行优先排列
        bool psi_valid；//是否有臂角
        double psi_c；//臂角
        bool motion_generation_finished；//运动是否结束
};
struct TorqueCommand {
        std::array<double, 7> tau_c；//关节力矩
};
struct RobotCommand{
        uint64_t message_id；
        MotionCommand motion；
        TorqueCommand torque；
};
```

3.2.2 RobotState

说明	
	机器人发送给 RCI 的数据结构

定义

```
struct RobotState {
    uint64_t message_id;
    std::array<double, 7> q{};//关节角度
    std::array<double, 7> q_c{};//指令关节角度
```

```
    std::array<double, 7> dq_m{};//关节速度
    std::array<double, 7> dq_c{};//指令关节速度
    std::array<double, 7> ddq_c{};//指令关节加速度
    std::array<double, 16> toolTobase_pos_m{};//机器人位姿
    std::array<double, 16> toolTobase_pos_c{};//指令机器人位姿
    std::array<double, 6> toolTobase_vel_c{};//指令机器人末端速度
    std::array<double, 6> toolTobase_acc_c{};//指令机器人末端加速度
    double psi_m;//臂角
    double psi_c;//指令臂角
    double psi_vel_c;//指令臂角速度
    double psi_acc_c;//指令臂角加速度
    std::array<double, 7> tau_m;//关节力矩
    std::array<double,7> tau_m_filtered;
    std::array<double, 7> tau_c;//指令关节力矩
    std::array<double, 7> tau_vel_c;//指令力矩微分
    std::array<double, 6> tau_ext_in_base;//基坐标系中外部力矩
    std::array<double, 6> tau_ext_in_stiff;//力控坐标系中外部力矩
    std::array<double, 7> theta_m;//电机位置
    std::array<double, 7> theta_vel_m;//电机位置微分
    std::array<double, 7> motor_tau;//电机转矩
    std::array<double,7> motor_tau_filtered;
    std::array<bool, 20> errors;//错误位
    double control_command_success_rate;//指令接收成功率
    MotionGeneratorMode motion_generator_mode;//运动发生模式
    ControllerMode controller_mode;//控制模式
    RobotMode robot_mode;//机器人状态
};
```

3.3 获取机器人状态和回调机制

RCI 通过回调函数的形式实现一个控制周期内机器人状态 RobotState 的获取与运动指令 RobotCommand 的下发,控制周期是 1ms。

调用 Control()函数来开启回调，客户端每隔 1ms 接收一次 RobotStates 数据，同时下发此时的 RobotCommand 命令。

示例程序：

```
int main(int argc, char *argv[]) {
    std::string ipaddr = "192.168.0.160";
    uint16_t port = 1337;

    // RCI 连接机器人
    xmate::Robot robot(ipaddr, port);
    //防止网络连接失败
    sleep(2);
    //机器人上电
    int power_state=1;
    robot.setMotorPower(power_state);
    std::array<double,7> q_init;
    std::array<double,7> q_drag = {{0,PI/6,0,PI/3,0,PI/2,0}};
    //接收一次 udp 数据，即机器人状态
    q_init = robot.receiveRobotState().q;
    //调用 MOVEJ 移动到目标位姿
    MOVEJ(0.2,q_init,q_drag,robot);

     //用户规划回调程序
    std::array<double, 7> q_end = {{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}};
    //  创建一个 S 规划器，0.2 是速度缩放系数，数值越大代表速度越快
    JointMotionGenerator joint_s(0.2, q_end);
    //开始运动前先设置控制模式和运动模式
    robot.startMove(RCI::robot::StartMoveRequest::ControllerMode::kJointPosition,
                    RCI::robot::StartMoveRequest::MotionGeneratorMode::kJointPosition);
    std::array<double, 7> init_position;
    std::array<double, 7> joint_delta;
    JointPositions output{};
    static bool init = true;
```

```
    double time = 0;
    // 用户回调函数，规划或者控制算法写在这里
    JointControl joint_position_callback;
    joint_position_callback = [&](RCI::robot::RobotState robot_state) -> JointPositions {
        time += 0.001;
        if(init==true){
            init_position = robot_state.q;
            init=false;
        }
        joint_s.calculateSynchronizedValues_joint(init_position);
        // calculateDesiredValues_joint 返回 true 代表规划结束
        if (joint_s.calculateDesiredValues_joint(time, joint_delta) == false) {
            output.q = {{init_position[0] + joint_delta[0], init_position[1] + joint_delta[1],
                         init_position[2] + joint_delta[2], init_position[3] + joint_delta[3],
                         init_position[4] + joint_delta[4], init_position[5] + joint_delta[5],
                         init_position[6] + joint_delta[6]}};
        }else{
            std::cout<<"运动结束："<<std::endl;
            return MotionFinished(output);
        }
        return output;
    };
    //调用 control 函数开启回调，运动结束前处于阻塞状态
    robot.Control(joint_position_callback);
    return 0;
}
```

3.4 运动学与动力学计算

RCI 为用户提供了 xMate 运动学与动力学计算的库，方便计算机器人正逆解和雅克比矩阵等。主要接口功能包括：

1. 运动学正解：pose = model.GetCartPose(q)，由关节角度 q 正解得到笛卡尔空间位姿 pose。
2. 动力学正逆解：model.GetJointPos(pose,q_init,q)，由目标笛卡尔空间位姿 pose 和机器人初始关节角度 q 逆解得到目标关节角度 q。
3. 雅可比矩阵计算：jac = model. Jacobian(q)，由关节角度 q 计算得到雅可比矩阵 jac。
4. 动力学正解：model. GetTauNoFriction(q,dq,ddq,trq_full,trq_inertial,trq_coriolis,trq_gravity)，由关节角度 q，关节角速度 dq，关节角加速度 ddq 正解得到总关节力矩 trq_full，离心力 trq_inertial，trq_coriolis 科氏力，trq_gravity 重力矩。

具体接口和使用方法可以参考 RCI_SDK 中 model.h 文件和 torques_control.cpp 文件。

3.5 S 轨迹规划接口

S 规划指规划轨迹速度是 S 曲线，能保证速度加速度的连续可导，使得运动高效平稳。用户可根据需求进行轴空间或者笛卡尔空间的 S 规划，是一种离线的规划方法，应该明确的是此方法不涉及到路径规划，路径应由用户定义与生成。

以轴空间的 s 规划为例：

JointMotionGenerator joint_s(0.2, q_end)，创建一个目标关节角度为 q_end 的规划器，0.2 为速度系数。

joint_s.calculateSynchronizedValues_joint(q_start)，指定初始关节角度 q_start，并做同步处理。

joint_s.calculateDesiredValues_joint(t, q_delta)，计算在时间 t 时的关节角度相对于 q_start 的增量 q_delta。

具体使用方法参见 RCI_SDK 中 joint_motion.h， cart_motion.h， joint_s.cpp 文件。

四 注意事项

用户下发的 RobotCommand 需要满足建议条件和必要条件。建议条件是为了让机器人运动更加平稳，性能最优。必要条件则是必须满足的，若不满足，机器人将会停止。

4.1 轴空间运动

1) 必要条件

轴空间轨迹平滑，至少保证速度是连续可导的：

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

2) 建议条件

力矩指令不超限:

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

运动开始时：

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

运动结束时：

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

4.2 笛卡尔空间运动

1) 必要条件

不处于奇异位且在工作空间内:

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

2) 建议条件

力矩指令不超限：

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

运动开始时：

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

运动结束时：

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

4.3 力矩直接控制

1) 必要条件

力矩指令不超限且连续：

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

$$-\tau_{jmax} < \tau_{jc} < \tau_{jmax}$$

4.4 运动限制条件

笛卡尔空间运动限制条件：

Name	Translation	Rotation
$\dot{p}_c$	$1.0\ m/s$	$2.5\ rad/s$
$\ddot{p}_c$	$10.0\ m/s^2$	$10.0\ rad/s^2$
$\dddot{p}_c$	$5000.0\ m/s^3$	$5000.0\ rad/s^3$

轴空间运动限制条件：

Name	Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	Joint7	Unit
$q_{\max}$	170	120	170	120	170	120	360	°
$q_{\min}$	-170	-120	-170	-120	-170	-120	-360	°
$\dot{q}_{\max}$	2.175	2.175	2.175	2.175	2.610	2.610	2.610	rad/s
$\ddot{q}_{\max}$	15	7.5	10	10	15	15	20	rad/s^2
$\dddot{q}_{\max}$	5000	3500	5000	5000	7500	7500	7500	rad/s^3

直接力矩控制限制条件

Name	Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	Joint7	Unit
$\tau_{\max}$	85	85	85	85	36	36	36	Nm
$\dot{\tau}_{\max}$	1500	1500	1500	1500	1000	1000	1000	Nm/s

4.5 DH 参数

xMate 3kg DH 参数表：

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

xMate 7kg DH 参数表：

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

4.6 错误处理

RCI 运行过程中可能会发生以下几种错误：

1) 实时性错误；
2) 网络通信错误；
3) 运动规划错误或控制算法不合理；
4) 碰撞等常规错误；

Errors	错误原因	解决方法
TCP 连接失败	RCI 客户端或者机器	检查 IP，配置正

	人 HMI 上 IP 地址配置不正确	确的 IP 和端口号
实时性错误	在 Linux 系统中，RCI 客户端没有正确运行在实时系统上	配置正确的实时内核补丁，运动时需要开启 root 权限
运动规划错误 kActualJointPositionLimitsViolation：0 kActualCartesianPositionLimitsViolation：1 kActualCartesianMotionGeneratorElbowLimitViolation：2 kActualJointVelocityLimitsViolation：3 kActualCartesianVelocityLimitsViolation：4 kActualJointAccelerationLimitsViolation：5 kActualCartesianAccelerationLimitsViolation：6 kCommandJointPositionLimitsViolation：7 kCommandCartesianPositionLimitsViolation：8 kCommandCartesianMotionGeneratorElbowLimitViolation：9 kCommandJointVelocityLimitsViolation：10 kCommandCartesianVelocityLimitsViolation：11 kCommandJointAccelerationLimitsViolation：12 kCommandCartesianAccelerationLimitsViolation：13 kCommandJointAccelerationDiscontinuity：14 kCommandTorqueDiscontinuity：17 kCommandTorqueRangeViolation：18	 实际轴角度超限 实际末端位姿超限 实际臂角超限 实际轴速度超限 实际末端速度超限 实际轴加速度超限 实际末端加速度超限 指令轴角度超限 指令末端位姿超限 指令臂角超限 指令轴速度超限 指令末端速度超限 指令轴加速度超限 指令末端加速度超限 指令轴加速度不连续 指令力矩不连续 指令力矩超限	检查规划轨迹是否连续可导，一般规划出来的轨迹需要做到速度和角速度连续可导，机器人运行不平稳或出现异响优先考虑轨迹是否平滑，必要时做额外的滤波处理。
其他错误 kCollision：15	检测到碰撞	若频繁触发碰撞检测，应适当调高碰撞检测阈值，急停触发

		也有可能触发此报错
kCartesianPositionMotionGeneratorInvalidFrame：16	机器人奇异	笛卡尔空间运动时机器人不应该经过奇异位姿
kInstabilityDetection：19	检测到不稳定	检查丢包阈值是否过于低，一般设置为10~20，或此时按下了急停开关也会触发此错误