UNIVERSAL ROBOTS Support

实时数据交换 （RTDE） 指南

这是如何使用UR控制器的数据同步协议的指南

最后修改时间 Jan 26， 2022

创建日期：2019年10月10日。

这是关于如何使用优傲控制器的数据同步协议的指南，例如在为优傲机器人开发URCaps/UR+时。

注意：所有文件都可以在此页面底部下载。

示例适用于：
CB3 软件版本：从 3.4 转发
系列软件版本：所有版本
注意较新的软件版本可能行为不同。

RTDE 在软件 3.4 的 CB3/CB3.1 上可用，但特定功能可能仅在较新的软件版本中可用。

RTDE 将外部可执行文件与 UR 控制器（例如 URCap）同步，而不会破坏任何实时属性。本文档首先介绍协议，然后提供 Python 客户端参考实现。

要了解本地主机上使用的端口，请阅读UR论坛中的这篇文章：https://forum.universal-robots.com/t/overview-of-used-ports-on-local-host/8889

实时数据交换

    - RTDE_TEXT_MESSAGE（议定书第2节）
    - RTDE_DATA_PACKAGE
    - RTDE_CONTROL_PACKAGE_SETUP_OUTPUTS（议定书第1节）
    - RTDE_CONTROL_PACKAGE_SETUP_OUTPUTS（议定书第2节）
    - RTDE_CONTROL_PACKAGE_SETUP_INPUTS
    - RTDE_CONTROL_PACKAGE_START
    - RTDE_CONTROL_PACKAGE_PAUSE

- RTDE 客户端 Python 模块

介绍

实时数据交换 （RTDE） 接口提供了一种通过标准 TCP/IP 连接将外部应用程序与 UR 控制器同步的方法，而不会破坏 UR 控制器的任何实时属性。此功能对于与现场总线驱动器（例如以太网/ IP）交互，操作机器人I / O和绘制机器人状态（例如机器人轨迹）非常有用。默认情况下，RTDE 接口在 UR 控制器运行时可用。

同步是可配置的，例如可以涉及以下数据：

- 输出：机器人、关节、工具和安全状态、模拟和数字 I/O 以及通用输出寄存器
- 输入：数字和模拟输出以及通用输入寄存器

RTDE 功能分为两个阶段：设置过程和同步循环。

连接到 RTDE 接口时，客户端负责设置要同步的变量。可以指定客户端分别需要写入和读取的输入和输出寄存器的任意组合。为此，客户端会发送应包含在实际数据同步包中的命名输入和输出字段的设置列表。同步数据包格式的定义称为配方。最大限制为 2048 字节，用于表示客户端要订阅的输入/输出字段名称列表。返回时，RTDE 接口使用变量类型列表进行回复，或指定未找到特定变量。已成功配置的每个输入配方都将获得唯一的配方 ID。可在下面找到受支持的字段名称及其关联数据类型的列表。设置完成后，可以启动和暂停数据同步。

当同步循环启动时，RTDE 接口将按照客户端请求的相同顺序向客户端发送请求的数据。此外，客户端应在值更改时向 RTDE 接口发送更新的输入。数据同步使用序列化的二进制数据。

所有包都共享相同的常规结构，并带有标头和有效负载（如果适用）。用于安装过程的包将生成一条返回消息。同步循环包则不然。客户端和服务器都可以随时发送文本消息，警告级别指定问题的严重性。RTDE 在端口号 30004 上可用。

首先，我们建议使用或修改用 Python 编写的客户端示例。

主要特点

- 实时同步：RTDE通常以125 Hz的频率生成输出消息。但是，控制器中的实时环路具有比RTDE接口更高的优先级。因此，如果控制器缺少计算资源，它将跳过许多输出包。跳过的包以后不会发送，控制器将始终发送最新的数据。输入包将始终在接收到的控制周期内进行处理，因此控制器的负载将根据接收的包的数量而变化。

- 输入消息：控制器中变量的更新可以分为多个消息。可以有一条消息来更新所有内容，也可以有一条消息来更新每个变量或两者之间的任何划分。不需要恒定的更新速率;输入保留其上次接收的值。注： 任何时候都只允许一个 RTDE 客户端控制特定变量。
- 运行时环境：RTDE 客户端可以在 UR 控制盒 PC 或任何外部 PC 上运行。在控制盒上运行 RTDE 客户端的优点是没有网络延迟。但是，RTDE 客户端和 UR 控制器将争用计算资源。请确保 RTDE 客户端以标准操作系统优先级运行。计算密集型过程，例如图像处理，应在外部PC上运行。
- 协议更改：UR 可能随时更新 RTDE 协议。为了保证 RTDE 客户端的最大兼容性，RTDE 客户端可以请求 RTDE 接口来提供特定的协议版本。协议添加/更改被明确表示，否则假定版本1。

字段名称和关联类型

机器人控制器输入

名字	类型	评论	版本介绍
speed_slider_mask	UINT32	0 = 不要使用此输入更改速度滑块 1 = 使用speed_slider_fraction设置速度滑块值	
speed_slider_fraction	双	新的速度滑块值	
standard_digital_output_mask	UINT8	标准数字输出位模板	
configurable_digital_output_mask	UINT8	可配置的数字输出位模板	
standard_digital_output	UINT8	标准数字输出	
configurable_digital_output	UINT8	可配置的数字输出	
standard_analog_output_mask	UINT8	标准模拟输出模板 位 0-1：standard_analog_output_0 |standard_analog_output_1	
standard_analog_output_type	UINT8	输出域 {0=电流[mA]， 1=电压[V]} 位 0-1：standard_analog_output_0 |standard_analog_output_1	
standard_analog_output_0	双	标准模拟量输出 0 （比率） [0..1]	
standard_analog_output_1	双	标准模拟输出 1 （比率） [0..1]	
input_bit_registers0_to_31	UINT32	通用位 布尔输入寄存器的此范围保留用于现场总线/PLC 接口使用。	

input_bit_registers32_to_63	UINT32	通用位 布尔输入寄存器的此范围保留用于现场总线/PLC 接口使用。 64 个通用位	
input_bit_register_*X*	布尔	X： [64..127] - 布尔输入寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	3.9.0 / 5.3.0
input_int_register_*X*	国际直辖区32	48 个通用整数寄存器 X： [0..23] - 整数输入寄存器的下限范围保留给 FieldBus/PLC 接口使用。 X： [24..47] - 整数输入寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
input_double_register_*X*	双	48 个通用双寄存器 X： [0..23] - 双输入寄存器的下限范围保留给现场总线/PLC接口使用。 X： [24..47] - 双输入寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
external_force_torque	矢量6D	使用内置ft_rtde_input_enable时输入外部扳手。	3.3

机器人控制器输出

名字	**类型**	**评论**	**介绍 在版本中**
时间戳	双	自控制器启动以来经过的时间 [s]	
target_q	矢量6D	瞄准关节位置	
target_qd	矢量6D	目标接头速度	
target_qdd	矢量6D	目标关节加速度	
target_current	矢量6D	目标关节电流	
target_moment	矢量6D	目标接头力矩（扭矩）	

actual_q	矢量6D	实际关节位置	
actual_qd	矢量6D	实际接头速度	
actual_current	矢量6D	实际接头电流	
joint_control_output	矢量6D	联合控制电流	
actual_TCP_pose	矢量6D	工具的实际笛卡尔坐标：（x，y，z，rx，ry，rz），其中 rx、ry 和 rz 是工具方向的旋转矢量表示	
actual_TCP_speed	矢量6D	以笛卡尔坐标给出的工具的实际速度。速度以 [m/s] 为单位，TCP 速度的旋转部分（rx、ry、rz）是以 [rad/s] 为单位的角速度	
actual_TCP_force	矢量6D	TCP 中的广义力。它补偿有效载荷产生的力和扭矩的测量值	
target_TCP_pose	矢量6D	工具的目标笛卡尔坐标：（x，y，z，rx，ry，rz），其中 rx、ry 和 rz 是工具方向的旋转矢量表示	
target_TCP_speed	矢量6D	笛卡尔坐标中给出的工具的目标速度。速度以 [m/s] 为单位，TCP 速度的旋转部分（rx、ry、rz）是以 [rad/s] 为单位的角速度	
actual_digital_input_bits	UINT64	数字输入的当前状态。0-7：标准，8-15：可配置，16-17：工具	
joint_temperatures	矢量6D	每个关节的温度（摄氏度）	
actual_execution_time	双	控制器实时线程执行时间	
robot_mode	国际直辖区32	机器人模式	
joint_mode	矢量6INT32	联合控制模式	
safety_mode	国际直辖区32	安全模式	
safety_status	国际直辖区32	安全状态	3.10.0 / 5.4.0
actual_tool_accelerometer	矢量3D	工具 x、y 和 z 加速度计值	
speed_scaling	双	轨迹限制器的速度缩放	
target_speed_fraction	双	目标速度分数	

actual_momentum	双	笛卡尔线性动量范数	
actual_main_voltage	双	安全控制板：主电压	
actual_robot_voltage	双	安全控制板：机器人电压（48V）	
actual_robot_current	双	安全控制板：机器人电流	
actual_joint_voltage	矢量6D	实际接头电压	
actual_digital_output_bits	UINT64	数字输出的当前状态。0-7：标准，8-15：可配置，16-17：工具	
runtime_state	UINT32	程序状态	
elbow_position	矢量3D	机器人弯头在笛卡尔基坐标中的位置	3.5.0 / 5.0.0
elbow_velocity	矢量3D	笛卡尔基坐标中机器人弯头的速度	3.5.0 / 5.0.0
robot_status_bits	UINT32	位 0-3：电源是否打开|程序运行|示教按钮是否按下|电源按钮是否已按下	
safety_status_bits	UINT32	位 0-10：正常模式是否|是缩减模式|保护是否停止|恢复模式是否|保护是否已停止|系统是否紧急停止|机器人紧急停止|紧急停止|是否|故障|由于安全原因已停止	
analog_io_types	UINT32	位 0-3：模拟输入 0 |模拟输入 1 |模拟输出 0 |模拟输出 1， {0=电流[mA]， 1=电压[V]}	
standard_analog_input0	双	标准模拟量输入 0 [mA 或 V]	
standard_analog_input1	双	标准模拟输入 1 [mA 或 V]	
standard_analog_output0	双	标准模拟输出 0 [mA 或 V]	
standard_analog_output1	双	标准模拟输出 1 [mA 或 V]	
io_current	双	输入/输出电流 [mA]	
euromap67_input_bits	UINT32	欧洲地图67 输入位	
euromap67_output_bits	UINT32	欧洲地图67 输出位	
euromap67_24V_voltage	双	欧洲地图 24V 电压 [V]	
euromap67_24V_current	双	欧洲地图 24V 电流 [mA]	

tool_mode	UINT32	工具模式	
tool_analog_input_types	UINT32	输出域 {0=电流[mA]， 1=电压[V]} 位 0-1：tool_analog_input_0 |tool_analog_input_1	
tool_analog_input0	双	工具模拟量输入 0 [mA 或 V]	
tool_analog_input1	双	工具模拟量输入 1 [mA 或 V]	
tool_output_voltage	国际直辖区 32	刀具输出电压 [V]	
tool_output_current	双	刀具电流 [毫安]	
tool_temperature	双	刀具温度（摄氏度）	
tcp_force_scalar	双	TCP 强制标量 [N]	
output_bit_registers0_to_31	UINT32	通用位	
output_bit_registers32_to_63	UINT32	通用位	
output_bit_register_X （X=[64 .. 127]）	布尔	64 个通用位 X： [64..127] - 外部RTDE客户端（即URCAPS）可以使用布尔输出寄存器的上限范围。	3.9.0 / 5.3.0
output_int_register_X （X=[0 .. 47]）	国际直辖区 32	48 个通用整数寄存器 X： [0..23] - 整数输出寄存器的下限保留给现场总线/PLC接口使用。 X： [24..47] - 整数输出寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
output_double_register_X （X=[0 .. 47]）	双	48 个通用双寄存器 X： [0..23] - 双输出寄存器的下限范围保留给现场总线/PLC接口使用。 X： [24..47] - 双输出寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
input_bit_registers0_to_31	UINT32	通用位（输入回读） 布尔输出寄存器的此范围保留给现场总线/PLC接口使用。	3.4.0

input_bit_registers32_to_63	UINT32	通用位（输入回读） 布尔输出寄存器的此范围保留给现场总线/PLC接口使用。	3.4.0
input_bit_register_X （X=[64 .. 127]）	布尔	64 个通用位 X： [64..127] - 外部RTDE客户端（即URCAPS）可以使用布尔输出寄存器的上限范围。	3.9.0 / 5.3.0
input_int_register_*X* （X=[0 .. 48]）	国际直辖区32	48 个通用整数寄存器 X： [0..23] - 整数输入寄存器的下限范围保留给现场总线/PLC接口使用。 X： [24..47] - 整数输入寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
input_double_register_*X* （X=[0 .. 48]）	双	48 个通用双寄存器 X： [0..23] - 双输入寄存器的下限范围保留给现场总线/PLC接口使用。 X： [24..47] - 双输入寄存器的上限范围可由外部RTDE客户端（即URCAPS）使用。	[0..23] 3.4.0 [24..47] 3.9.0 / 5.3.0
tool_output_mode	UINT8	电流输出模式	
tool_digital_output0_mode	UINT8	数字输出的电流模式 0	
tool_digital_output1_mode	UINT8	数字输出1的电流模式	
有效载荷	双	有效载荷质量 Kg	3.11.0 / 5.5.1
payload_cog	矢量3D	有效载荷重心 （CoGx， CoGy， CoGz） m	3.11.0 / 5.5.1
payload_inertia	矢量6D	有效载荷惯性矩阵元素 （Ixx，Iyy，Izz，Ixy，Ixz，Iyz] 以 kg*m^2 表示	3.15 / 5.11
script_control_line	UINT32	给定机器人时实际控制机器人的脚本行号被脚本中的一个线程锁定。 如果没有线程锁定机器人，则此字段设置为"0"。不应将脚本行号与多面镜上显示的程序树行号混淆。	3.14.0 / 5.9.0
ft_raw_wrench	矢量6D	原始力和扭矩测量，不补偿有效载荷引起的力和扭矩	5.9.0

数据类型

名字	类型	大小（以位为单位）
布尔	0 = 假，其他所有内容 = 真	8
UINT8	无符号整数	8
UINT32	无符号整数	32
UINT64	无符号整数	64
国际直辖区32	有符号整数，二的补码	32
双	IEEE 754 浮点	64
矢量3D	3倍双倍	3x64
矢量6D	6倍双倍	6×64
矢量6INT32	6x32英寸	6×32
VECTOR6UINT32	6xUINT32	6×32
字符串	ASCII char array	长度 x8

网络字节顺序：数据始终按网络字节顺序发送，因此大端顺序。如果直接从套接字读取原始数据，您将在大端中看到它，并且可能需要转换为小端。UR 的 rtde-client 库将转换为 little。

协议

EE = 外部可执行文件

CON = 机器人控制器

输出：CON -> EE

输入： CON <- EE

页眉

字段名称	数据类型
包装尺寸	uint16_t

封装类型	uint8_t

摘要：所有包都使用标头。

支持的包类型包括：

封装类型	**值（12 月）**	**等效于 ASCII**
RTDE_REQUEST_PROTOCOL_VERSION	86	V
RTDE_GET_URCONTROL_VERSION	118	v
RTDE_TEXT_MESSAGE	77	M
RTDE_DATA_PACKAGE	85	U
RTDE_CONTROL_PACKAGE_SETUP_OUTPUTS	79	O
RTDE_CONTROL_PACKAGE_SETUP_INPUTS	73	我
RTDE_CONTROL_PACKAGE_START	83	S
RTDE_CONTROL_PACKAGE_PAUSE	80	P

方向： 双边

返回：不可用

RTDE_REQUEST_PROTOCOL_VERSION

字段名称	**数据类型**
页眉	见上文
协议版本	uint16_t

摘要：请求控制器使用"协议版本"

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
接受	uint8_t

摘要：控制器接受或不接受，即 1（成功）或 0（失败）。成功后，EE 应说出指定的协议版本，CON 将在该版本中应答。

RTDE_GET_URCONTROL_VERSION

字段名称	数据类型
页眉	见上文

摘要：检索 CON 主要、次要、错误修复和内部版本号。

方向： EE -> CON

返回

字段名称	数据类型
页眉	见上文
主要	uint32_t
次要	uint32_t
错误修复	uint32_t
建	uint32_t

摘要：主要、次要、错误修复和内部版本号。

RTDE_TEXT_MESSAGE（议定书第1节）

方向： CON -> EE

字段名称	数据类型
页眉	见上文
消息类型	uint8_t
消息	字符串

方向： EE -> CON

字段名称	数据类型
页眉	见上文
消息长度	uint8_t
消息	字符串
源长度	uint8_t

源	字符串
警告级别	uint8_t

摘要：发送异常、错误、警告或信息消息。

警告级别：EXCEPTION_MESSAGE、ERROR_MESSAGE、WARNING_MESSAGE、INFO_MESSAGE

EE->CON：异常表示 EE 程序失败，无法恢复。错误、警告和信息将最终出现在 PolyScope 日志中。

CON -> EE：主要表示不同类型的协议故障。

方向：见上文

返回：不可用。

RTDE_TEXT_MESSAGE（议定书第2节）

字段名称	**数据类型**
页眉	见上文
消息长度	uint8_t
消息	字符串
源长度	uint8_t
源	字符串
警告级别	uint8_t

摘要：发送异常、错误、警告或信息消息。

警告级别：EXCEPTION_MESSAGE、ERROR_MESSAGE、WARNING_MESSAGE、INFO_MESSAGE

EE->CON：异常表示 EE 程序失败，无法恢复。错误、警告和信息将最终出现在 PolyScope 日志中。

CON -> EE：主要表示不同类型的协议故障。

方向： 双边

返回：不可用。

RTDE_DATA_PACKAGE

字段名称	**数据类型**
页眉	见上文

食谱编号	uint8_t
<可变>	<数据类型>

摘要：分别对 CON/EE 输入的更新。

<可变>是打包/序列化的二进制文件，并且与SETUP_OUTPUTS或SETUP_INPUTS请求返回的类型匹配。

方向： 双边

返回：不可用

RTDE_CONTROL_PACKAGE_SETUP_OUTPUTS（议定书第1节）

字段名称	**数据类型**
页眉	见上文
变量名称	字符串

摘要：设置输出配方。目前，CON 仅支持一个输出配方。包应包含所有所需的输出变量。变量名称是逗号分隔的变量名称字符串的列表。

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
变量类型	字符串

摘要：按请求中提供的变量类型的相同顺序返回变量类型。

变量类型： VECTOR6D， VECTOR3D， VECTOR6INT32， VECTOR6UINT32， DOUBLE， UINT64， UINT32， INT32， BOOL， UINT8

如果变量不可用，则变量类型将为"NOT_FOUND"。

如果出现一个或多个"NOT_FOUND"返回值，则配方被视为无效，RTDE 不会输出此数据。

RTDE_CONTROL_PACKAGE_SETUP_OUTPUTS（议定书第2节）

字段名称	**数据类型**
页眉	见上文

输出频率	双
变量名称	字符串

摘要：设置输出配方。目前，CON仅支持一个输出配方，输出频率是可配置的。对于CB系列机器人，频率必须在1到125 Hz之间，输出速率将根据地板（125 /频率）而定，而对于e系列机器人，频率可以达到500 Hz。包应包含所有所需的输出变量。变量名称是逗号分隔的变量名称字符串的列表。

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
输出配方 id	uint8_t
变量类型	字符串

摘要：按请求中提供的变量类型的相同顺序返回变量类型。

变量类型： VECTOR6D， VECTOR3D， VECTOR6INT32， VECTOR6UINT32， DOUBLE， UINT64， UINT32， INT32， BOOL， UINT8

如果变量不可用，则变量类型将为"NOT_FOUND"。

如果出现一个或多个"NOT_FOUND"返回值，则该配方将被视为无效，并且 RTDE 不会输出此数据（输出配方 id = 0）。

RTDE_CONTROL_PACKAGE_SETUP_INPUTS

字段名称	**数据类型**
页眉	见上文
变量名称	字符串

摘要：设置 CON 输入配方。CON 支持 255 种不同的输入配方（0 为保留）。变量名称是逗号分隔的变量名称字符串的列表。

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
输入配方 id	uint8_t

变量类型	字符串

摘要：按请求中提供的变量类型的相同顺序返回变量类型。

变量类型： VECTOR6D， VECTOR3D， VECTOR6INT32， VECTOR6UINT32， DOUBLE， UINT64， UINT32， INT32， BOOL， UINT8

如果一个变量已被另一个 EE 声明，则变量类型将为"IN_USE"。

如果变量不可用，则变量类型将为"NOT_FOUND"。

如果出现一个或多个"IN_USE"或"NOT_FOUND"返回值，则配方被视为无效（输入配方 id = 0）。

RTDE_CONTROL_PACKAGE_START

字段名称	**数据类型**
页眉	见上文

摘要：请求控制器开始发送输出更新。例如，如果尚未配置输出包，这将失败。

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
接受	uint8_t

摘要：CON 接受与否。1（成功）或 0（失败）。

RTDE_CONTROL_PACKAGE_PAUSE

字段名称	**数据类型**
页眉	见上文

摘要：请求 CON 暂停发送输出更新。

方向： EE -> CON

返回

字段名称	**数据类型**
页眉	见上文
接受	uint8_t

摘要：CON 将始终接受暂停命令并返回 1（成功）。

RTDE 客户端 PYTHON 模块

RTDE 客户端可以用支持套接字通信的不同语言实现。这个用Python编写的RTDE客户端库的目的是提供一个简单的起点并显示一些示例应用程序。该功能是为Python 2.7.11开发的。

例子

record.py

将此脚本用作可执行文件来记录来自机器人的输出数据并将其保存到 csv 文件中。

可选参数

- --host：要连接到的主机或 IP 地址的名称（默认：本地主机）
- --端口：端口号（默认：30004）
- --样本：要记录的特定样本数（否则程序将记录数据，直到收到SIGINT / Ctrl + C）
- --频率：赫兹的采样频率
- --config：要使用的 XML 配置文件 - 它将使用带有键"out"的配方（默认：record_configuration.xml）
- --output：要写入的数据输出文件 - 现有文件将被覆盖（默认值：robot_data.csv）
- --详细：启用信息日志记录输出到控制台
- -h：显示帮助

example_plotting.py

提供一种从使用 *record.py* 记录的 csv 文件中读取和绘制数据的简单方法。

example_control_loop.py

简单控制回路的示例。具有两个输入配方和一个输出配方的配置将从 XML 文件中读取并发送到 RTDE 服务器。控制回路包括一个阻塞读取，然后是一些非常简单的数据处理，然后向机器人发送新信息。

RTDE 模块

本节介绍 rtde 模块的不同类及其公共接口。

类csv_reader。CSVReader（csvfile， delimiter）

读取 CSV 文件并将每列映射到对象的命名空间字典中的条目中。列标题是字典键，值是列中数据点的数组。

输入参数：

- csvfile（文件）：任何具有 *read（）* 方法的类似文件的对象。
- 分隔符（字符串）：用于分隔字段的单字符字符串。它默认为' '。

类csv_writer。CSVWriter（csvfile， names， types， delimiter）

返回一个编写器对象，该对象可以采用 RTDE 数据对象，并将它们转换为分隔字符串，并将它们写入类似文件的对象。

输入参数：

- csvfile（文件）：任何具有 *write（）* 方法的类似文件的对象。
- 名称（数组<字符串>）：变量名称列表
- 类型（数组<字符串>）：变量类型的列表
- 分隔符（字符串）：用于分隔字段的单字符字符串。它默认为' '。

写头（）

根据变量名称将列标题写入文件中的当前行。多维变量将获得附加到每列名称的索引。

写行（data_object）

根据提供的数据对象向文件写入新行。

输入参数：

- data_object（数据对象）：其成员变量与配置的 RTDE 变量的名称匹配的数据对象

类rtde_config。配置文件（文件名）

RTDE 配置可以从包含配方列表的 XML 文件加载。每个配方都应该有一个键和一个具有变量名称和类型的字段列表。示例 XML 配方：

```
<？xml version="1.0"？>
<rtde_config>
<recipe key="out">
<字段名称="时间戳"类型="DOUBLE"/>
<字段名称="target_q"类型="VECTOR6D"/>
<字段名称="target_qd"类型="VECTOR6D"/>
<字段名称="speed_scaling"类型="DOUBLE"/>
<字段名称="output_int_register_0"类型="INT32"/>
</食谱>
<recipe key="in1">
<字段名称="input_int_register_0"类型="INT32"/>
```

```
<字段名称="input_int_register_1"类型="INT32"/>
</食谱>
<recipe key="in2">
<字段名称="input_double_register_0"类型="DOUBLE"/>
</食谱>
</rtde_config>
```

get_recipe（键）

获取与指定键关联的配方，该键以名称列表和类型列表的形式提供。

输入参数：

- 键（字符串）：与配方关联的键

返回值：

- 变量（数组<字符串>）：变量名称列表
- 类型（数组<字符串>）：变量类型的列表

类序列化。数据对象（）：

一个数据传输对象，其中为同步配置的 RTDE 变量名称已添加到类的命名空间字典中，以便于访问。这意味着例如，可以在输出 DataObject 上访问时间戳，如下所示：*objName.timestamp*。数据对象用于输入和输出。

recipe_id

recipe_id是 DataObject 实例上的整数成员变量，用于标识在 RTDE 服务器中配置的输入包。它不用于输出包。

类 Rtde.RTDE（主机名，端口）

构造函数将主机名和端口号作为参数。

输入参数：

- 主机名（字符串）：RTDE 服务器的主机名或 IP
- 端口 （int）： [可选] 端口号（默认值：30004）

连接（）

初始化与主机的 RTDE 连接。

返回值：

- 成功（布尔值）

断开连接（）

关闭 RTDE 连接。

is_connected（）

如果连接处于打开状态，则返回 True。

返回值：

- 打开（布尔值）

get_controller_version（）

返回运行 RTDE 服务器的机器人控制器的软件版本。

返回值：

- 主要（英特）
- 次要 （整数）
- 错误修复 （int）
- 构建 （整数）

negotiate_protocol_version（协议）

与服务器协商协议版本。如果控制器支持指定的协议版本，则返回 True。我们建议您使用它来确保您的应用程序与机器人控制器的未来版本之间的完全兼容性。

输入参数：

- 协议 （int）：协议版本号

返回值：

- 成功（布尔值）

send_input_setup（变量、类型）

配置外部应用程序将发送到机器人控制器的输入包。输入包是外部应用程序将在单个更新中提供给机器人控制器的输入变量的集合。变量是变量名称的列表，应是 RTDE 接口支持输入的名称的子集。类型列表是可选的，但如果提供了任何类型，则其长度应与变量列表相同。提供的类型将与 RTDE 接口所需的类型匹配，如果它们不相等，则该函数将返回 None。可以配置多个输入包。返回的 InputObject 具有对配方 ID 的引用，该 ID 用于在发送更新时标识特定的输入格式。

输入参数：

- 变量（数组<字符串>）：可能 RTDE 输入列表中的变量名称
- 类型（数组<字符串>）： [可选] 与变量匹配的类型

返回值：

- input_data（数据对象）：其成员变量与配置的 RTDE 变量的名称匹配的空对象

send_output_setup（变量、类型）

配置机器人控制器将以控制频率发送到外部应用程序的输出包。变量是变量名称的列表，应是 RTDE 接口支持输出的名称的子集。类型列表是可选的，但如果提供了任何类型，则其长度应与变量列表相同。提供的类型将与 RTDE 接口预期的类型匹配，如果它们不相等，则该函数返回 False。只能指定一种输出包格式，因此输出不使用配方 ID。

输入参数：

- 变量（数组<字符串>）：可能的 RTDE 输出列表中的变量名称
- 类型（数组<字符串>）：[可选] 与变量匹配的类型

返回值：

- 成功（布尔值）

send_start（）

向 RTDE 服务器发送启动命令以启动实际同步。所有输入和输出的设置应在开始同步之前完成。

返回值：

- 成功（布尔值）

send_pause（）

向 RTDE 服务器发送暂停命令以暂停同步。暂停后，可以更改输入和输出配置并再次启动同步。

返回值：

- 成功（布尔值）

发送（input_data）

将数据对象的内容作为输入发送到 RTDE 服务器。如果成功，则返回 True。

输入参数：

- input_data（数据对象）：其成员变量与配置的 RTDE 变量的名称匹配的对象

返回值：

- 成功（布尔值）

接收（）

阻止调用以从 RTDE 服务器接收下一个输出数据对象。

返回值：

- output_data（数据对象）：其成员变量与配置的 RTDE 变量的名称匹配的对象