RealMan Robot rm_driver User Manual V1.0

RealMan Intelligent Technology (Beijing) Co., Ltd.

Revision History:

No.	Date	Comment
V1.0	11/17/2023	Draft

Content

1. rm_driver package description

rm_driver package is very important in the ROS2 robotic arm package. This package realizes the function of controlling the robotic arm through communication between ROS and the robotic arm. The package will be introduced in detail in the following text through the following aspects:

1. Package use.
2. Package architecture description.
3. Package topic description.

Through the introduction of the three parts, it can help you:

1. Understand the package use.
2. Familiar with the file structure and function of the package.
3. Familiar with the topic related to the package for easy development and use.

Source code address:https://github.com/RealManRobot/ros2_rm_robot.git.

2. rm_driver package use

2.1 Basic use of the package

First, after configuring the environment and completing the connection, we can directly start the node and control the robotic arm through the following command.

The current control is based on the fact that we have not changed the IP of the robotic arm, which is still 192.168.1.18.

```
rm@rm-desktop:~$ ros2 launch rm_driver rm_<arm_type>_driver.launch.py
```

In practice, the above <arm_type> needs to be replaced by the actual model of the robotic arm. The available models of the robotic arm are 65, 63, eco65, and 75.

The following screen will appear if the underlying driver is successfully started.

```
rm@rm-desktop:~$ ros2 launch rm_driver rm_65_driver.launch.py
[INFO] [launch]: All log files can be found below /home/rm/.ros/log/2023-11-18-14-03-58-781410-rm-desktop-4330
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [rm_driver-1]: process started with pid [4331]
[rm_driver-1] [INFO] [1700287442.074033349] [rm_65_driver]: Rm_65_driver is running
[rm_driver-1]
```

2.2 Advanced use of the package

When our robotic arm's IP is changed, our start command is invalid. If we use the above command directly, we cannot successfully connect to the robotic arm. We can re-establish the connection by modifying the following configuration file.

The configuration file is located in the config folder under our rm_driver package.

```
rm@rm-desktop:~/ros2_ws/src/rm_driver/config$ ls
rm_63_config.yaml  rm_65_config.yaml  rm_75_config.yaml  rm_eco65_config.yaml
```

The contents of the configuration file are as follows:

```
rm_driver:
  ros__parameters:
    #robot param
    arm_ip: "192.168.1.18"            # Set the IP address for the TCP connection
    tcp_port: 8080#            # Set the port for the TCP connection

    arm_type: "RM_65"                  # set the robotic arm model
    arm_dof: 6                         # Set the degree of freedom of the robotic arm
```

```
udp_ip: "192.168.1.10"              # set the udp active reporting IP address
udp_cycle: 5                        # the active reporting cycle of UDP, which needs to be a multiple of 5.
udp_port: 8089                      # Set the udp active reporting port
udp_force_coordinate: 0             # Set the base coordinate of the six-axis force when the system is forced, where 0 is the sensor coordinate system, 1 is the current work coordinate system, and 2 is the current tool coordinate system
```

There are mainly the following parameters.

arm_ip: This parameter represents the current IP of the robotic arm

tcp_port: set the port when TCP is connected.

arm_type: This parameter represents the current model of the robotic arm. The parameters that can be selected are RM_65 (65 series), RM_eco65 (ECO65 series), RM_63 (63 series), and RM_75 (75 series).

arm_dof: set the degree of freedom of the robotic arm. 6 is 6 degrees of freedom, and 7 is 7 degrees of freedom.

udp_ip: set the udp active reporting IP address.

udp_cycle: the active reporting cycle of UDP, which needs to be a multiple of 5.

udp_port: set the udp active reporting port.

udp_force_coordinate: set the base coordinate of the six-axis force when the system is forced, where 0 is the sensor coordinate system (original data), 1 is the current work coordinate system, and 2 is the current tool coordinate system.

In practice, we choose the corresponding launch file to start, which will automatically select the correct model. If there are special requirements, you can modify the corresponding parameters here. After modification, recompile the configuration in the workspace directory, and then the modified configuration will take effect.

Run the colcon build command in the workspace directory.

```
rm@rm-desktop: ~/ros2_ws$ colcon build
```

After successful compilation, follow the above commands to start the package.

3. rm_driver package architecture description

3.1 Overview of package files

The current rm_driver package is composed of the following files.

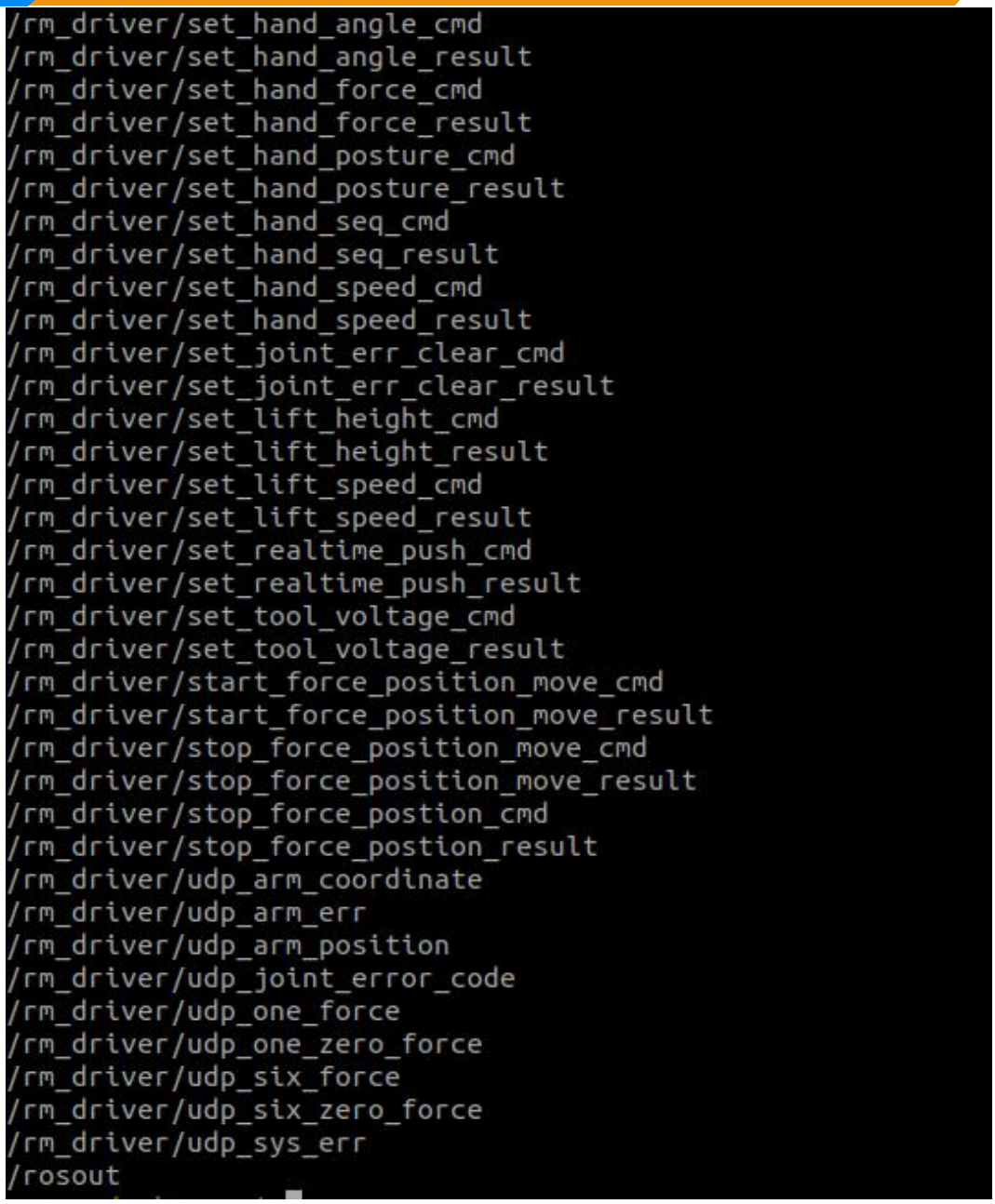
```
├── CMakeLists.txt                  # compilation rule file
├── config                                  # config folder
│   ├── rm_63_config.yaml       #63 configuration file
│   ├── rm_65_config.yaml       #65 configuration file
│   ├── rm_75_config.yaml       #75 configuration file
│   └── rm_eco65_config.yaml    # eco65 configuration file
├── include                             # dependency header file folder
│   └── rm_driver
│       ├── cJSON.h                 # API header file
│       ├── constant_define.h       # API header file
```

```
│       ├── rman_int.h                  # API header file
│       ├── rm_base_global.h            # API header file
│       ├── rm_base.h                   # API header file
│       ├── rm_define.h                 # API header file
│       ├── rm_driver.h                 #rm_driver.cpp header file
│       ├── rm_praser_data.h            # API header file
│       ├── rm_queue.h                  # API header file
│       ├── rm_service_global.h         # API header file
│       ├── rm_service.h                # API header file
│       └── robot_define.h              # API header file
├── launch
│   ├── rm_63_driver.launch.py          # 63 launch file
│   ├── rm_65_driver.launch.py          # 65 launch file
│   ├── rm_75_driver.launch.py          # 75 launch file
│   └── rm_eco65_driver.launch.py       # eco65 launch file
├── lib
│   ├── libRM_Service.so -> libRM_Service.so.1.0.0          # API library file
│   ├── libRM_Service.so.1 -> libRM_Service.so.1.0.0        # API library file
│   ├── libRM_Service.so.1.0 -> libRM_Service.so.1.0.0      # API library file
│   └── libRM_Service.so.1.0.0                               #API library file
├── package.xml                                              # dependency declaration file
└── src
    └── rm_driver.cpp                                        # driver code source file
```

4. rm_driver topic description

rm_driver has many topics, and you can learn about the topic information through the following commands.

```
rm@rm-desktop:~$ ros2 topic list
/joint_states
/parameter_events
/rm_driver/change_work_frame_cmd
/rm_driver/change_work_frame_result
/rm_driver/clear_force_data_cmd
/rm_driver/clear_force_data_result
/rm_driver/force_position_move_joint_cmd
/rm_driver/force_position_move_pose_cmd
/rm_driver/get_all_tool_frame_cmd
/rm_driver/get_all_tool_frame_result
/rm_driver/get_all_work_frame_cmd
/rm_driver/get_all_work_frame_result
/rm_driver/get_arm_software_version_cmd
/rm_driver/get_arm_software_version_result
/rm_driver/get_curr_workFrame_cmd
/rm_driver/get_curr_workFrame_result
/rm_driver/get_current_arm_original_state_result
/rm_driver/get_current_arm_state_cmd
/rm_driver/get_current_arm_state_result
/rm_driver/get_current_tool_frame_cmd
/rm_driver/get_current_tool_frame_result
/rm_driver/get_lift_state_cmd
/rm_driver/get_lift_state_result
/rm_driver/get_realtime_push_cmd
/rm_driver/get_realtime_push_result
/rm_driver/move_stop_cmd
/rm_driver/move_stop_result
/rm_driver/movec_cmd
/rm_driver/movec_result
/rm_driver/movej_canfd_cmd
/rm_driver/movej_cmd
/rm_driver/movej_p_cmd
/rm_driver/movej_p_result
/rm_driver/movej_result
/rm_driver/movel_cmd
/rm_driver/movel_result
/rm_driver/movep_canfd_cmd
/rm_driver/set_force_postion_cmd
/rm_driver/set_force_postion_result
/rm_driver/set_gripper_pick_cmd
/rm_driver/set_gripper_pick_on_cmd
/rm_driver/set_gripper_pick_on_result
/rm_driver/set_gripper_pick_result
/rm_driver/set_gripper_position_cmd
/rm_driver/set_gripper_position_result
```



It is mainly for the application of API to achieve some of the robotic arm functions; for a more complete introduction and use, please see the special document "RealMan Robotic Arm ROS2 Topic Detailed Description".