User Manual

Segway RMP
(Robot Mobile Platform)

RMP Lite 220

©2021 Segway-Ninebot All Rights Reserved
General Information

The Robot Mobile Platform (RMP) provides general or integrated robot chassis solutions for enterprises or third-party developers, with versatility, durability and performance in mind. RMP Lite 220 (hereinafter referred to as RMP) is a mobile robot chassis product designed for indoor and outdoor distribution, inspection, service, cleaning, and warehousing AGV. It is designed to provide large-scale and customized services for companies in the robotics field such as special application robots.

Main features of RMP:

- Hardware: compact and equipped with a large-capacity battery;
- Software: compatible with ROS and Isaac operating systems;
- Hardware modular design, interface with software SDK, support secondary development or customized service;
- Support extension kits include: light strips, sensor mounting rods, and so on.
Safety

Incorrect use of RMP may cause loss of control, collision or fall of the RMP, resulting in property damage, personal injury, and even death. Therefore, in order to reduce risks and avoid injury, please read and follow all instructions and warnings in this manual.

The following secure messaging conventions are used in this document:

<table>
<thead>
<tr>
<th>Warning!</th>
<th>Warns you of operations that may cause serious injury or even death</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention!</td>
<td>Warns you of operations that may cause minor or moderate injuries</td>
</tr>
<tr>
<td>Please note</td>
<td>Indicates important information, but does not involve personal injury</td>
</tr>
</tbody>
</table>

Warning!

- Please keep RMP out of the reach of children and pets. Accidental movement of RMP may cause injury or even death.
- Please do not sit, stand or ride on the RMP. Doing so may cause injury or even death.
- Please do not control RMP to hit people or animals. Collision may cause injury or even death.
• When RMP is running, remind people nearby at all times. Accidental collision with RMP may cause injury or even death.

• Avoid power failure on slopes. RMP cannot maintain its position on the slope when the power is off. The power off will cause RMP to slide, which may cause injury or even death.

• RMP can accelerate quickly. It is recommended that using low speed to practice until the you are familiar with controlling RMP. Accidental movement of RMP may cause injury or even death.

• Please do not try to disassemble the battery, it may cause electric shock, burns or even fire. Attempting to open the battery case will damage the battery case, release toxic and harmful substances, and also render the battery unusable.

• The same with all rechargeable batteries, please do not charge near flammable materials, which may cause a fire.

• If the battery case is damaged or the battery emits peculiar smell, smoke, overheating or leakage, please do not continue to use the battery and do not contact with any substances leaking from the battery to avoid poisoning.

• Strictly observe and follow all safety information on the warning label on the battery. Failure to do so may result in injury or even death.
• Please do not use cables that have been seriously worn or damaged, which may shock yourself or damage the RMP.

Attention!
• The performance parameters should be set correctly and carefully. RMP follows the commands issued to it, users are responsible for implementing correct and safe performance parameters.
• Do not charge the battery may cause permanent damage to the battery.
• Only can use the charger that is provided with RMP Lite 220 to charge the battery.
• Before operating RMP, please be sure to read the user’s manual and be familiar with the operation of RMP and various precautions.

Please Note
• If the user modifies the chassis without communication with Segway-Ninebot and causes an accident, Segway-Ninebot does not assume any responsibility.
# Table Of Contents

1 Product introduction ........................................................................................................7  
  1.1 Product diagram ...................................................................................................... 7  
  1.2 Component description ............................................................................................ 8  
  1.3 Remote control ........................................................................................................ 9  
    1.3.1 Remote control diagram .................................................................................. 9  
    1.3.2 Receiver pairing ............................................................................................... 10  
    1.3.3 Remote control control car instructions .......................................................... 11  
    1.3.4 Upper computer control car instructions ......................................................... 12  
2 Software introduction ................................................................................................... 16  
  2.1 Documents provided to users ................................................................................. 166  
  2.2 Interface function introduction .............................................................................. 166  
    2.2.1 C/C++ Interface introduction ........................................................................... 166  
    2.2.2 ROS Interface introduction—SmartCar ............................................................ 20  
    2.2.3 Error code information table ........................................................................... 23  
3 Firmware upgrade and version upgrade ...................................................................... 25  
  3.1 Firmware upgrade .................................................................................................... 25  
  3.2 Version upgrade ....................................................................................................... 28  
Appendix I System parameters and mode switching logic ............................................. 31  
Appendix II Connector welding instructions .................................................................. 34  
Appendix III Connector pin angle definition instructions ............................................. 37  
Appendix IV C/C++ API Reference documents ................................................................ 37
1. Product introduction

1.1. Product diagram

Figure 1

Figure 2
## 1.2 Component description

### Table 1

<table>
<thead>
<tr>
<th>No.</th>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Power button</td>
<td>On: Hold down, when power button and indicator light are always on and accompanied by a beep, chassis boots successfully. At this time, the chassis is in lock mode, and the indicator light is steady yellow. Off: Hold down until the prompt sound starts, release the button, chassis shut down successfully. At this time, power button and indicator light are always off.</td>
</tr>
<tr>
<td>2</td>
<td>Indicator light</td>
<td>Colors and status of the indicator light represent different modes of the product.</td>
</tr>
<tr>
<td>3</td>
<td>Driving wheel</td>
<td>11 inch high adhesion pneumatic tire with good shock absorption and design with drainage tank.</td>
</tr>
<tr>
<td>4</td>
<td>Universal wheel</td>
<td>Super Artificial rubber, lightweight and shock absorption.</td>
</tr>
<tr>
<td>5</td>
<td>Mounting hole</td>
<td>Use to install upper equipment.</td>
</tr>
<tr>
<td>6</td>
<td>Battery</td>
<td>Power supply to chassis and upper system.</td>
</tr>
<tr>
<td>7</td>
<td>Emergency stop button</td>
<td>Use to switch the chassis to emergency stop mode in an emergency.</td>
</tr>
<tr>
<td>8</td>
<td>Charging port</td>
<td>Connect the charger to charge the device.</td>
</tr>
<tr>
<td>9</td>
<td>Battery lock</td>
<td>Used to fix the battery, need to use the matching key to open.</td>
</tr>
<tr>
<td>10</td>
<td>Upper computer power supply port</td>
<td>It supplies power to the upper computer, with a maximum current of 10A.</td>
</tr>
<tr>
<td>11</td>
<td>Electric control box</td>
<td>Use to install circuit module to control the chassis.</td>
</tr>
<tr>
<td>12</td>
<td>Upper computer power supply port</td>
<td>It supplies power to the upper computer.</td>
</tr>
</tbody>
</table>
1.3 Remote control

1.3.1 Remote control diagram

*The forward or backward of the remote control input (throttle or rudder) can be realized by flipping the enable switch under the T8FB.
*The alarm voltage of the remote control is adaptive to 2S, 3S, 4S lithium batteries and 4 NI–MH batteries. That is, if T8FB is powered by 2S, 3S, 4S lithium batteries or 4 NI–HM batteries, after connecting the battery, T8FB will automatically set the low voltage alarm value according to the battery type.

1.3.2 Receiver pairing

Each transmitter has an independent ID code. Before starting to use the device, receiver must pair the code with the transmitter. After pairing the code, the ID code is stored in the receiver, and there is no need to pair it again, unless the receiver is used with another transmitter. When you have a new receiver, you must pair the code again, otherwise the receiver will not work normally.

(1) Place the remote control and receiver horizontally with a spacing of about 50cm;

(2) Turn on the power switch of the remote control to supply power to the receiver, and the receiver’s LED light starts to flash slowly;

(3) Press the pairing code button (ID SET) on the side of the receiver for more than 1 second, the LED light starts to flash quickly, that means the code is being paired, and the receiver will look for the
nearest remote control to pair the code;

(4) When the receiver’s LED light stops flashing, it means that pairing the code is completed. If the receiver’s LED light flashes slowly, it means that pairing the code has failed, the code needs to be paired again.

1.3.3 Remote control control car instructions

(1) Turn on the RMP chassis: press the RMP power button;

Note: Please check the RMP status. Press and hold until the buzzer sounds and there is no continuous beeping, the indicator light is steady yellow.

(2) Turn on the remote control: push up the power switch of the remote control;

Note: ensure that the remote control is not in the emergency stop state and enters the enable state. That is, the emergency stop switch is not at the bottom, and the enable switch is dialed from the top to the bottom.

(3) At this time RMP is in Normal mode, see the table below for specific operations:
Table 2: Car control and remote control operation

<table>
<thead>
<tr>
<th>Car control</th>
<th>Remote control operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn left/ right</td>
<td>Rudder lever</td>
</tr>
<tr>
<td>Move forward/backward</td>
<td>Throttle lever</td>
</tr>
<tr>
<td>Emergency stop/ exit emergency stop</td>
<td>Emergency stop switch: At the top: exit the emergency stop; At the bottom: start the emergency stop</td>
</tr>
<tr>
<td>Adjust the maximum value of angular velocity</td>
<td>Maximum angular velocity adjustment knob: Turn left: the maximum angular velocity decreases, Turn right: increases.</td>
</tr>
<tr>
<td>Adjust the maximum value of linear velocity</td>
<td>Maximum linear velocity adjustment knob: Turn left: the maximum linear velocity decreases, Turn right: increases.</td>
</tr>
<tr>
<td>Enable/Disable</td>
<td>Turn the enable switch from the top to the bottom: Enable; Turn the enable switch from the bottom to the top: Disable.</td>
</tr>
</tbody>
</table>

1.3.4 Upper computer control car instructions

The host computer is the control computer, which can directly issue control commands and display various information changes on the screen. The upper computer controls the lower computer and provides some necessary operating environment for the lower computer, and extends the man–machine control or demonstration function that the lower computer can provide. The upper computer has the characteristics of leading management, coordinating resources, monitoring agency, and controlling RMP.

(1) Turn on the RMP chassis: press the RMP power button;
Note 1: Please check the RMP status. Press and hold until the buzzer sounds and there is no continuous beeping, the indicator light is steady yellow.

Note 2: When the upper computer controls the car, the remote control cannot be turned on. Or if the remote control is turned on, turn its enable switch upward.

(2) Ensure that the RMP serial line or CAN line is connected to the upper computer;

(3) In the upper computer, give permissions to “/sdcard/segway/hardware_log/” folder, otherwise it will fail to create a new log file; give permissions to all files in the “/catkin_ws/src/RosCode/segwayrmp/lib/” directory (no need to reset after first setup):

```
cd /sdcard/segway/hardware_log
sudo chmod 777 /sdcard/segway/hardware_log/
cd $PRO_HOME$/catkin_ws/src/RosCode/segwayrmp/lib/
sudo chmod 777 *
```

(4) In “catkin_ws/src/RosCode/segwayrmp/Cmakelists.txt” file, according to the upper computer in x86_64 or arm platform, select the compilation option, as shown below when compiling in x86_64 platform, comment out “libctrl_arm64-v8a.so” with the symbol “#”, (no need to reset after first setup):

```
cmakelists..."libctrl_arm64-v8a.so"...
```
`target_link_libraries(SmartCar`

`$\{catkin_LIBRARIES\}`

`#\$\{PROJECT_SOURCE_DIR\}/lib/libctrl_arm64-v8a.so    //in x86_64 platform, comment out this line, in arm platform, do not comment out this line`'

`$\{PROJECT_SOURCE_DIR\}/lib/libctrl_x86_64.so    //in arm platform, comment out this line, in x86_64 platform, do not comment out this line`'

(5) Enter ROS system, run the following command to compile the “segway_msgs” package message.

```
cd catkin_ws

  catkin_make

-DCATKIN_WHITELIST_PACKAGES='segway_msgs'
```

(6) Enter ROS system, run the following command to compile the “segwayrmp” package message.

```
cd catkin_ws

  catkin_make

-DCATKIN_WHITELIST_PACKAGES='segwayrmp'
```

(7) Control car in ROS system:

1) Create a new terminal, run the following command:

```
cd catkin_ws
```
roscore

2) Create a new terminal, run the following command, run SmarCar node:
   cd catkin_ws
   source devel/setup.bash
   rosrun segwayrmp SmartCar

3) Create a new terminal, run the following command, run routine test node:
   cd catkin_ws
   source devel/setup.bash
   rosrun segwayrmp ChassisResponseTest
2 Software introduction

This chapter introduces relevant documents, software interface functions and fault code information provided by RMP.

2.1 Documents provided to users

<table>
<thead>
<tr>
<th>Document</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libctrl_x86_64.so</td>
<td>Provide C/C++ chassis-related interfaces in x86 platform</td>
</tr>
<tr>
<td>Libctrl_arm64-v8a.so</td>
<td>Provide C/C++ chassis-related interfaces in arm platform</td>
</tr>
<tr>
<td>Comm_ctrl_navigation.h</td>
<td>C/C++ API interface header file</td>
</tr>
<tr>
<td>ROS package</td>
<td>Provide chassis control ROS nodes</td>
</tr>
</tbody>
</table>

2.2 Interface function introduction

2.2.1 C/C++ Interface introduction

<table>
<thead>
<tr>
<th>Callback type</th>
<th>Callback No.</th>
<th>Function description</th>
<th>Data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chassis_Data_Speed</td>
<td>1</td>
<td>Chassis speed information</td>
<td>typedef struct{ int16_t L_speed;</td>
</tr>
</tbody>
</table>
1. Odom data: The default heading angle is 0 degrees at start up.

2. IMU (gyroscope and accelerometer) data: the carrier coordinate system XYZ corresponds to the right front up.

### Table 5 event definition

<table>
<thead>
<tr>
<th>Event type</th>
<th>Event No.</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChassisBootReadyEvent</td>
<td>1</td>
<td>Chassis central control</td>
</tr>
</tbody>
</table>
board start up completed
Chassis shutdown
Enter emergency stop
Exit emergency stop
Locked rotor event occurs
Locked rotor event removes
Lost control event occurs
Lost control event removes
Calibrate the gyroscope success
Calibrate the gyroscope fail
Calibrate phase current success
Calibrate phase current fail

<table>
<thead>
<tr>
<th>Interface name</th>
<th>Interface description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_err_state</td>
<td>Get the error code of the upper computer/central control board/motor board/battery</td>
</tr>
<tr>
<td>get_bat_soc</td>
<td>Get the percentage of battery remaining</td>
</tr>
<tr>
<td>get_bat_charging</td>
<td>Get battery charge status (1: charging; 0: non–charging)</td>
</tr>
<tr>
<td>get_bat_mvoll</td>
<td>Get battery voltage (unit: millivolt (mV))</td>
</tr>
<tr>
<td>get_bat_mcurrent</td>
<td>Get battery current (unit: mA)</td>
</tr>
<tr>
<td>get_bat_temp</td>
<td>Get battery temperature (unit: degrees Celsius (˚C))</td>
</tr>
<tr>
<td>get_chassis_work_model</td>
<td>Get chassis work model (0: Unload; 1: Onload)</td>
</tr>
<tr>
<td>get_chassis_load_state</td>
<td>Get chassis load state (0: empty; 1: full)</td>
</tr>
<tr>
<td>get_chassis_mode</td>
<td>Get chassis mode (0: Locked; 1: Control; 2: Push; 3: Emergency stop; 4: Error)</td>
</tr>
<tr>
<td>get_ctrl_cmd_src</td>
<td>Get the current chassis control source (0: remote control; 1: upper computer)</td>
</tr>
<tr>
<td>get_vehicle_meter</td>
<td>Get chassis mileage (unit: meter(m))</td>
</tr>
<tr>
<td>get_host_version</td>
<td>Get the upper computer version number</td>
</tr>
<tr>
<td>get_chassis_central_version</td>
<td>Get the control board version number</td>
</tr>
<tr>
<td>get_chassis_motor_version</td>
<td>Get the motor board version number (Reserved)</td>
</tr>
<tr>
<td>get_line_forward_max_vel_fb</td>
<td>Get the forward speed limit value of the chassis (unit: meter per hour(m/h))</td>
</tr>
<tr>
<td>get_line_backward_max_vel_fb</td>
<td>Get the backward speed limit value of the chassis (unit: meter per hour(m/h))</td>
</tr>
<tr>
<td>get_angular_max_vel_fb</td>
<td>Get the limit value of chassis angular velocity (unit:</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getIapTotalProgress</td>
<td>Get IAP progress</td>
</tr>
<tr>
<td>iapCentralBoard</td>
<td>Upgrade the central control board IAP</td>
</tr>
<tr>
<td>iapMotorBoard</td>
<td>Upgrade the motor board IAP</td>
</tr>
<tr>
<td>isHostIapOver</td>
<td>Check if IAP is over</td>
</tr>
<tr>
<td>getHostIapResult</td>
<td>Get the IAP result (3: completed; 4: failed; 5: interrupted; 0: meaningless)</td>
</tr>
<tr>
<td>getHostIapErrorCode</td>
<td>Get IAP error code</td>
</tr>
<tr>
<td>get_chassis_hang_mode</td>
<td>Get whether the chassis is in the hang mode (0: not in the hang mode; 1: in the hang mode)</td>
</tr>
<tr>
<td>get_charge_mos_ctrl_status</td>
<td>Get charging MOS status (1: charging MOS is on, 0: MOS is off) (temporarily reserved)</td>
</tr>
<tr>
<td>set_cmd_vel</td>
<td>Set the chassis linear velocity and angular velocity (unit: meter per second(m/s) and radian per second(rad/s))</td>
</tr>
<tr>
<td>set_line_forward_max_vel</td>
<td>Set the forward speed limit value of the chassis (unit: meter per second(m/s))</td>
</tr>
<tr>
<td>set_line_backward_max_vel</td>
<td>Set the backward speed limit value of the chassis (unit: meter per second(m/s))</td>
</tr>
<tr>
<td>set_angular_max_vel</td>
<td>Set the limit value of chassis angular velocity (unit: radian per second(rad/s))</td>
</tr>
<tr>
<td>set_enable_ctrl</td>
<td>Set the enable state of the upper computer control car on the chassis (1: enable; 0 disable)</td>
</tr>
<tr>
<td>init_control_ctrl</td>
<td>Chassis initialization interface</td>
</tr>
<tr>
<td>exit_control_ctrl</td>
<td>Chassis exit initialization interface</td>
</tr>
<tr>
<td>set_smart_car_serial</td>
<td>Set the serial port name used by the upper computer dynamic library</td>
</tr>
<tr>
<td>set_comu_interface</td>
<td>Set the communication interface for communication with the chassis (0: serial port; 1: CAN)</td>
</tr>
<tr>
<td>set_chassis_load_state</td>
<td>Set chassis load state (0: empty; 1: full)</td>
</tr>
<tr>
<td>set_chassis_poweroff</td>
<td>Set chassis shutdown command</td>
</tr>
<tr>
<td>set_remove_push_cmd</td>
<td>Remove chassis push command</td>
</tr>
<tr>
<td>setHostIapCanceled</td>
<td>Cancel the upper computer IAP command</td>
</tr>
<tr>
<td>set_chassis_hang_mode</td>
<td>Set chassis hang mode (1: enter the hang mode; 0: exit the hang mode)</td>
</tr>
<tr>
<td>set_charge_mos_ctrl</td>
<td>Set charging MOS switch (1: turn on MOS, 0: turn off MOS) (temporarily reserved)</td>
</tr>
</tbody>
</table>

2.2.2 ROS Interface introduction—SmartCar

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>Function Description</th>
<th>Message Type</th>
<th>Message Type Info</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Battery related information

<table>
<thead>
<tr>
<th>TopicName</th>
<th>Function Description</th>
<th>Message Type</th>
<th>Message Type Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bms_fb</td>
<td>Battery related information</td>
<td>Segway_msgs/Bms_fb</td>
<td>int16 bat_soc, int16 bat_charging, int32 bat_vol, int32 bat_current, int16 bat_temp</td>
</tr>
<tr>
<td>Chassis_ctrl_src_fb</td>
<td>Chassis control command source</td>
<td>Segway_msgs/Chassis_ctrl_src_fb</td>
<td>uint16 chassis_ctrl_cmd_src</td>
</tr>
<tr>
<td>Chassis_mileage_meter_fb</td>
<td>Chassis mileage</td>
<td>Segway_msgs/Chassis_mileage_meter_fb</td>
<td>uint32 vehicle_meters</td>
</tr>
<tr>
<td>Chassis_mode_fb</td>
<td>Chassis mode</td>
<td>Segway_msgs/Chassis_mode_fb</td>
<td>uint16 chassis_mode</td>
</tr>
<tr>
<td>Error_code_fb</td>
<td>Chassis error code</td>
<td>Segway_msgs/Error_code_fb</td>
<td>uint32 host_error, uint32 central_error, uint16 left_motor_error, uint16 right_motor_error, uint32 bms_error</td>
</tr>
<tr>
<td>Motor_work_mode_fb</td>
<td>Chassis working mode</td>
<td>Segway_msgs/Motor_work_mode_fb</td>
<td>uint16 motor_work_mode #0: no output torque 1: output torque</td>
</tr>
<tr>
<td>Speed_fb</td>
<td>Chassis speed</td>
<td>Segway_msgs/Speed_fb</td>
<td>float32 car_speed, float32 turn_speed, float32 l_speed, float32 r_speed, uint64 speed_timestamp</td>
</tr>
<tr>
<td>Ticks_fb</td>
<td>Chassis encoder information</td>
<td>Segway_msgs/Ticks_fb</td>
<td>int32 l_ticks, int32 r_ticks, uint64 ticks_timestamp</td>
</tr>
<tr>
<td>Odom</td>
<td>Odom data</td>
<td>Nav_msgs/odom</td>
<td>40</td>
</tr>
<tr>
<td>imu</td>
<td>Imu data</td>
<td>Sensor_msgs/imu</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 8 News Subscription

<table>
<thead>
<tr>
<th>TopicName</th>
<th>Function Description</th>
<th>Message Type</th>
<th>Message Type Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cmd_vel</td>
<td>Control</td>
<td>Geometry_msgs/twist</td>
<td>Angular.z //rad/s</td>
</tr>
</tbody>
</table>
## Table 9 Service Client

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Function Description</th>
<th>Message Type</th>
<th>Message Type Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>chassis_send_event_srv</td>
<td>Send time number</td>
<td>Segway_msgs/chassis_send_event</td>
<td>chassis_send_event_id, ros_is_received</td>
</tr>
</tbody>
</table>

## Table 10 Service Server

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Function Description</th>
<th>Message Type</th>
<th>Message Type Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>ros_get_load_param_cmd_srv</td>
<td>Get load parameter</td>
<td>Segway_msgs/rosl-load-param_cmd</td>
<td>ros_get_load_param #0:no_load, 1:full_load</td>
</tr>
<tr>
<td>ros_get_charge_mos_ctrl_status_cmd_srv</td>
<td>Get chassis charging MOS status (reserved temporarily)</td>
<td>Segway_msgs/rosl-get_charge_mos_ctrl_status_cmd</td>
<td>ros_get_chassis_charge_ctrl_status #1: MOS opened; 0: MOS closed</td>
</tr>
<tr>
<td>ros_get_sw_version_cmd_srv</td>
<td>Get software version</td>
<td>Segway_msgs/rosl-get_sw_version_cmd</td>
<td>uint16 host_version, uint16 central_version, uint16 motor_version</td>
</tr>
<tr>
<td>ros_get_vel_max_feedback_cmd_srv</td>
<td>Get the maximum speed limit</td>
<td>Segway_msgs/rosl-get_vel_max_feedback_cmd</td>
<td>ros_get_vel_max_fb_cmd #0:forward_max_vel_fb, 1:backward_max_vel_fb, angular_max_vel_fb</td>
</tr>
<tr>
<td>ros_set_charge_mos_ctrl_cmd_srv</td>
<td>Set chassis charging MOS (reserved temporarily)</td>
<td>Segway_msgs/rosl-set_charge_mos_ctrl_cmd</td>
<td>ros_set_chassis_charge_ctrl_result #1: MOS opened; 0: MOS closed</td>
</tr>
<tr>
<td>ros_set_chassis_enable_cmd_srv</td>
<td>Set chassis enable command</td>
<td>Segway_msgs/rosl-set_chassis_enable_cmd</td>
<td>ros_set_chassis_enable_cmd</td>
</tr>
</tbody>
</table>

Linear.x \( /m/s \)
<table>
<thead>
<tr>
<th>Action name</th>
<th>Function Description</th>
<th>Message type</th>
<th>Message type info</th>
</tr>
</thead>
<tbody>
<tr>
<td>ros_set_iap_cmd_action</td>
<td>Upgrade the board firmware IAP</td>
<td>Segway_msgs/roset_iap_cmdAction</td>
<td>Bool central_board_iap_enable _ _ _ Int16 iap_result #3: iap_state_complete; 4: iap_state_fail; 5: iap_state_abort Int16 error_code #When iap_result value is 4, this value represents the error code _ _ _ Int16 iap_percent</td>
</tr>
</tbody>
</table>

**Table 11 Action server**

<table>
<thead>
<tr>
<th>Action name</th>
<th>Function Description</th>
<th>Message type</th>
<th>Message type info</th>
</tr>
</thead>
<tbody>
<tr>
<td>ros_set_chassis_poweroff_cmd_srv</td>
<td>Set chassis shutdown command</td>
<td>Segway_msgs/roset_chassis_poweroff_cmd</td>
<td>ros_set_chassis_poweroff_cmd _ _ _ chassis_set_poweroff_result</td>
</tr>
<tr>
<td>ros_set_load_param_cmd_srv</td>
<td>Set chassis load state</td>
<td>Segway_msgs/roset_load_param_cmd</td>
<td>ros_set_load_param #0:no_load, 1: full_load _ _ _ chassis_set_load_param_result</td>
</tr>
<tr>
<td>ros_set_remove_push_cmd_srv</td>
<td>Set remove chassis push command</td>
<td>Segway_msgs/roset_remove_push_cmd</td>
<td>ros_set_remove_push_cmd _ _ _ chassis_set_remove_push_result</td>
</tr>
<tr>
<td>ros_set_vel_max_cmd_srv</td>
<td>Set the maximum speed limit</td>
<td>Segway_msgs/roset_vel_max_cmd_srv</td>
<td>ros_set_forward_max_vel ros_set_backward_max_vel ros_set_angular_max_vel _ _ _ chassis_set_max_vel_result</td>
</tr>
</tbody>
</table>

**2.2.3 Error code information table**

The error code is obtained through:
“uint32_t get_err_state(board_name_e board_name)” interface, and the corresponding information is as follows:

Table 12 Error code

<table>
<thead>
<tr>
<th>Board name</th>
<th>Bit</th>
<th>Error info</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>0x00000000</td>
<td>No error</td>
</tr>
<tr>
<td>host</td>
<td>0x00000001</td>
<td>Loss of control board</td>
</tr>
<tr>
<td>host</td>
<td>0x00000002</td>
<td>Unplug the serial port module</td>
</tr>
<tr>
<td>host</td>
<td>0x00000000</td>
<td>No error</td>
</tr>
<tr>
<td>host</td>
<td>0x00000001</td>
<td>Car control command communication interrupted</td>
</tr>
<tr>
<td>host</td>
<td>0x00000002</td>
<td>Motor board communication interrupted</td>
</tr>
<tr>
<td>host</td>
<td>0x00000004</td>
<td>IMU initialization failed</td>
</tr>
<tr>
<td>host</td>
<td>0x00000008</td>
<td>IMU failed to read data</td>
</tr>
<tr>
<td>host</td>
<td>0x00000010</td>
<td>Lost control</td>
</tr>
<tr>
<td>host</td>
<td>0x00000020</td>
<td>Locked rotor</td>
</tr>
<tr>
<td>host</td>
<td>0x00000040</td>
<td>Failed to calibrate IMU</td>
</tr>
<tr>
<td>host</td>
<td>0x00000080</td>
<td>Read Flash failed</td>
</tr>
<tr>
<td>host</td>
<td>0x00000100</td>
<td>IMU data update failed</td>
</tr>
<tr>
<td>host</td>
<td>0x00000200</td>
<td>Bms initialization failed to enter test mode</td>
</tr>
<tr>
<td>host</td>
<td>0x00000400</td>
<td>Rollover</td>
</tr>
<tr>
<td>host</td>
<td>0x00000800</td>
<td>Any motor board restart is detected</td>
</tr>
<tr>
<td>host</td>
<td>0x00001000</td>
<td>Left magnetic encoder fault</td>
</tr>
<tr>
<td>host</td>
<td>0x00002000</td>
<td>Right magnetic encoder fault</td>
</tr>
<tr>
<td>host</td>
<td>0x00004000</td>
<td>Battery communication interrupted</td>
</tr>
<tr>
<td>host</td>
<td>0x00008000</td>
<td>Battery communication interrupted (30s)</td>
</tr>
<tr>
<td>Central</td>
<td>0x00000000</td>
<td>No error</td>
</tr>
<tr>
<td>Motor</td>
<td>0x00000000</td>
<td>No error</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x00000001</td>
<td>Phase current fault</td>
<td></td>
</tr>
<tr>
<td>0x00000002</td>
<td>Phase voltage fault</td>
<td></td>
</tr>
<tr>
<td>0x00000004</td>
<td>Lack of phase</td>
<td></td>
</tr>
<tr>
<td>0x00000008</td>
<td>Under voltage</td>
<td></td>
</tr>
<tr>
<td>0x00000010</td>
<td>Over voltage</td>
<td></td>
</tr>
<tr>
<td>0x00000020</td>
<td>Over current</td>
<td></td>
</tr>
<tr>
<td>0x00000040</td>
<td>Over temperature</td>
<td></td>
</tr>
<tr>
<td>0x00000080</td>
<td>Locked rotor</td>
<td></td>
</tr>
<tr>
<td>0x00000100</td>
<td>Electrical angle fault</td>
<td></td>
</tr>
<tr>
<td>0x00000200</td>
<td>Excessive power fault</td>
<td></td>
</tr>
<tr>
<td>0x00000400</td>
<td>Over speed fault</td>
<td></td>
</tr>
<tr>
<td>0x00000800</td>
<td>Rotational speed sensor fault</td>
<td></td>
</tr>
<tr>
<td>0x00010000</td>
<td>Angle sensor fault</td>
<td></td>
</tr>
<tr>
<td>0x00020000</td>
<td>Current loop fault</td>
<td></td>
</tr>
<tr>
<td>0x00040000</td>
<td>Speed loop fault</td>
<td></td>
</tr>
<tr>
<td>0x00080000</td>
<td>Angle loop fault</td>
<td></td>
</tr>
</tbody>
</table>

**Battery**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>No error</td>
</tr>
<tr>
<td>0x00000008</td>
<td>Overcharge</td>
</tr>
<tr>
<td>0x00000010</td>
<td>Charge over temperature protection</td>
</tr>
</tbody>
</table>
3 Firmware upgrade and version upgrade

IAP is a software function module of the system, that is, in application programming, that means upgrade the single chip computer program online. This function uses the upper computer to burn the new version bin file to the single chip computer (including the central control board, the motor drive board, etc.) when the program is running. The premise is that the single chip computer’s bin file, which will be burned, needs to be named according to the requirements of the upper computer, and places it under the “/sdcard/firmware/” path of the upper computer. And then the bin file can be upgraded online through the command at the terminal.

3.1 Firmware upgrade

Before the firmware upgrade, it is necessary to test the data communication between the upper computer and each lower computer to check whether the communication is normal. Use the command to test in the shell terminal.

(1) View the path of the upper computer program
Enter the path where the upper computer program is located, and check whether the upper computer executable file exists. As shown in the figure below, they are arm executable file, x86 executable file, arm dynamic library, and x86 dynamic library:

![Figure 5](image)

(2) Check the software version of each lower computer board

Check the software version of the lower computer. This step can test the communication between the upper computer and the lower computer at the same time. If the software version of each section of the lower computer can be checked through the upper computer, it indicates that the communication is normal.

Central control board test command:

```
./ctrl_x86_64 s  -test central
```

1) When connecting for the first time, if the serial port’s USB port does not have execution permission, the program requires root permission to modify the executable permission of the serial port’s USB port. At this time, you need to enter the system login.
password, and then hit the enter key, as shown in the figure below:

```
password: ...
```

Figure 6

2) When communication fails, the version number is 0, as shown in the figure below:

```
host version build date: [21-04-19]
host version build time: [20:15:06]
communication interface adding SERIAL INTERFACE
Please enter the administrator permission login password:
serial open success serial port/dev/ttyCOM0, baudrate 9600
Scheduler Num 0 Start: Task Num = 1, Period = 100000
Scheduler Num 1 Start: Task Num = 1, Period = 10000
Scheduler Num 2 Start: Task Num = 1, Period = 20000
central board test started...........
get chassis control version: 0
get_chassis getProperty: 0
get_chassis getProperty: 0
get_chassis getProperty: 0
```

Figure 7

3) When the communication is successful, the version number is printed as follows, and it is a non-zero number. At this time, the communication between the upper computer and the single chip computer is normal, and the online upgrade can be performed:

```
host version build date: [21-04-19]
host version build time: [20:15:06]
communication interface adding SERIAL INTERFACE
Please enter the administrator permission login password:
serial open success serial port/dev/ttyCOM0, baudrate 9600
Scheduler Num 0 Start: Task Num = 1, Period = 100000
Scheduler Num 1 Start: Task Num = 1, Period = 10000
Scheduler Num 2 Start: Task Num = 1, Period = 20000
central board test started...........
get chassis control version: 810000
get_chassis getProperty: 0
get_chassis getProperty: 0
get_chassis getProperty: 0
```

Figure 8
3.2 Version upgrade

(1) Single chip computer bin file placement

Put the board’s bin file which will be upgraded into the
“/sdcard/firmware” path of the upper computer, for example,
the central control board’s bin file “central.bin”, as shown in the
following figure:

```
$ ll
drwxrwxrwx 2 root root 4096 4月 20 17:45 / 
drwxrwxrwx 5 root root 4096 6月 30 2020 /
-rwxr-xr-x 1 root root 10066 4月 19 20:06 central.bin
-rwxr-xr-x 1 root root 65896 4月 19 20:06 motor.bin
```

Figure 9

(2) Online burning of bin file in lower computer

Enter the path where the upper computer program
executable file “ctrl_x86_64” or “ctrl_arm64-v8a” is located,
as follows:

```
$ ll
drwxrwxrwx 1 root root 4096 4月 20 17:35 / 
drwxrwxrwx 1 root root 4096 12月 23 14:46 /
-rwxr-xr-x 1 root root 13350517 4月 28 17:35 ctrl_arm64-v8a
-rwxr-xr-x 1 root root 370088 4月 20 17:35 ctrl_x86_64
-rwxr-xr-x 1 root root 435588 4月 20 17:35 libctrl_arm64-v8a.so
-rwxr-xr-x 1 root root 419352 4月 20 17:35 libctrl_x86_64.so
```

Figure 10

The commands for online upgrade of each board are as
follows. After entering the upper computer program path, execute
the following commands (use ‘s’ when using the serial port; use ‘c’ when using the CAN port):

Central control board upgrade command:

“./ctrl_x86_64 s -iap central”

Motor board upgrade command:

“./ctrl_x86_64 s -iap motor”

Take the central control board as an example, enter the command: “./ctrl_x86_64 s -iap central” to upgrade, as shown in the figure below:

![Figure 11](image)

During the upgrading, you can view the upgrade progress. Progress represents the percentage of the IAP upgrade progress. When the Progress value reaches 100, it means that the routing board bin file has been programmed into the central control board chip. As shown below:
(3) Test the online upgrade result of the IAP version:

Perform step 1, test and check the software version number, enter the command: 

```
./ctrl_x86_64 s -test central
```

in the path where the upper computer program is located, as shown below:

At this time, the software version number of the central control board is 0x1000, indicating that the online upgrade has been successful, and the communication between the upper computer and the central control board is good.
Appendix I: System parameters and mode switching logic

<table>
<thead>
<tr>
<th>Structural parameter</th>
<th>System parameters</th>
<th>Performance parameter</th>
<th>Communication</th>
<th>Battery</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Length<em>Width</em>Height (mm)</td>
<td>730<em>500</em>280</td>
<td>UART, CAN</td>
<td>48V 24Ah</td>
<td>Button</td>
</tr>
<tr>
<td>Structural parameters</td>
<td>Wheelbase<em>Tread</em>Ground clearance (mm):</td>
<td>513.5<em>414</em>68</td>
<td>Support driver, API</td>
<td>C/C++, ROS</td>
<td>Emergency stop button, Power button</td>
</tr>
<tr>
<td>Tire size</td>
<td>11Inch (280mm) hub motor</td>
<td></td>
<td>Feedback data</td>
<td>Magnetic, Hall, IMU</td>
<td>Status indication</td>
</tr>
<tr>
<td>Weight (with battery)</td>
<td>27.2kg</td>
<td>Maximum speed</td>
<td>3m/s</td>
<td></td>
<td>Power status indicator, Chassis status indicator, Control source indication, Battery display, Charging status display</td>
</tr>
<tr>
<td>Nominal load</td>
<td>50kg</td>
<td>Maximum steering speed</td>
<td>3rad/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obstacle surmounting</td>
<td>4cm/8° Slope/Deceleration zone</td>
<td>Minimum turning radius</td>
<td>0m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overhang</td>
<td>4mm Rear overhang</td>
<td>Braking distance</td>
<td>No load: 3m/s 0.95m, Braking acceleration: 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gear train structure</td>
<td>Front drive, differential steering</td>
<td>Control mode</td>
<td>Remote control, Upper computer control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protection level</td>
<td>IPX5</td>
<td>Braking mode</td>
<td>Electric brake</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 System parameters

<table>
<thead>
<tr>
<th>Mode switching logic</th>
</tr>
</thead>
</table>

Table 2 Mode switching logic
<table>
<thead>
<tr>
<th>Chassis mode</th>
<th>Enter</th>
<th>Implement</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locked car mode</td>
<td>1. The default mode of chassis power on</td>
<td>0 speed closed loop, shield speed command, and the status indicator light keeps yellow</td>
<td>1. An unrecoverable exception errorcode is detected and enters the error mode</td>
</tr>
<tr>
<td></td>
<td>2. Default mode after emergency stop recovery</td>
<td></td>
<td>2. Press the emergency stop button to enter the emergency stop mode</td>
</tr>
<tr>
<td></td>
<td>3. In control car mode, when recoverability abnormality (such as communication timeout, communication disconnection, etc.) occurs, enters the locked car mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control car mode</td>
<td>1. In locked car mode, the enable command is received</td>
<td>Closed loop, accepting control commands. Remote control car: indicator light flashes green; upper computer controls car: indicator light keeps green</td>
<td>1. An unrecoverable exception errorcode is detected and enters the error mode</td>
</tr>
<tr>
<td></td>
<td>2. When the load exceeds 20kg, it is recommended to use 'set_chassis_load_state' and switch the car control mode to overload mode</td>
<td></td>
<td>2. When recoverability abnormality (such as communication timeout, communication disconnection, etc.) occurs, enters to locked car mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Press the emergency stop button to enter the emergency stop mode</td>
</tr>
<tr>
<td>Emergency stop mode</td>
<td>In non abnormal mode, press the emergency stop button</td>
<td>Relief force, shield speed and enable</td>
<td>The emergency stop button pops up and enters the locked car</td>
</tr>
</tbody>
</table>

1. Controls power on
2. Default mode after emergency stop recovery
3. In control car mode, when recoverability abnormality (such as communication timeout, communication disconnection, etc.) occurs, enters the locked car mode.
<table>
<thead>
<tr>
<th>Mode</th>
<th>Status and Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error mode</td>
<td>Unrecoverable exception error code detected. Braking, releasing force, shielding speed and enable commands. The indicator light is keeps red.</td>
</tr>
<tr>
<td></td>
<td>1. Restart</td>
</tr>
</tbody>
</table>
Appendix II Connector welding instructions

(I) Preparatory work

1. Tools

   Electric soldering iron, solder wire

2. Materials

   8pin connector, 2pin connector, 2 AWG16 cables, 8 AWG26 cables, as shown in Figure 1.

(II) Welding instructions (take 8pin connector as an example)

1. Figure 2 shows the 8pin connector received by the customer.

   Screw the connector from the position shown by the red arrow to disassemble it into the state shown in Figure 3;
2. Take out the part shown in Figure 4, which is the part that needs to be welded;

3. As shown in Figure 5, the pin angle number of the connector can be seen from one side of the component, and then rotate it 180°, which is the part needs to be welded;

4. Use the AWG 26 cables to weld according to the pin angle definition in the welding manual (see appendix III for details). After the welding is completed, as shown in Figure 6;
5. Take out the two parts shown in Figure 7 and put them on the welded parts, as shown in Figure 8;

![Figure 7](image1)
![Figure 8](image2)

6. Take out the part shown in Figure 9, put them on the previously assembled parts, and tighten them, as shown in Figure 10;

![Figure 9](image3)
![Figure 10](image4)

7. Then connect the remote control receiver and serial port, as shown in Figure 11;

![Figure 11](image5)

8. The welding method of 2pin connector is the same as that of 8pin connector.
## Appendix III Connector pin angle definition instructions

<table>
<thead>
<tr>
<th>Connector</th>
<th>Pin number</th>
<th>Define</th>
<th>Wire size</th>
<th>Remark</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>8pin</td>
<td>1</td>
<td>CANH</td>
<td>AWG26</td>
<td></td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CANL</td>
<td>AWG26</td>
<td></td>
<td>Gray</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>TX</td>
<td>AWG26</td>
<td></td>
<td>Blue</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>RX</td>
<td>AWG26</td>
<td></td>
<td>Green</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>GND</td>
<td>AWG26</td>
<td></td>
<td>White</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5V</td>
<td>AWG26</td>
<td></td>
<td>Brown</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>GND</td>
<td>AWG26</td>
<td></td>
<td>Black</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>S.B PPM</td>
<td>AWG26</td>
<td></td>
<td>Yellow</td>
</tr>
<tr>
<td>2pin</td>
<td>1</td>
<td>Power+</td>
<td>AWG16</td>
<td>power supply for upper system</td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Power-</td>
<td>AWG16</td>
<td></td>
<td>Black</td>
</tr>
</tbody>
</table>
Appendix IV C/C++ APIReference documents

int Init_Comcore(void)

Function: initialization of host computer dynamic link library

Parameter: none

Return value: 0: initialization succeed;

Other: initialization fail

void exit_Comcore(void)

Function: exit initialization of host computer dynamic link library

Parameter: none

Return value: none

void aprctrl_datastamped_jni_register(saprctrl_datastamped_t* f)
Function: registration via the callback function provided by parameter, and this callback function conducts the sensor data processing.

Parameter: f is a struct pointer, and this struct includes the unique function pointer member variables.

Return value: none

```c
void aprctrl_eventcallback_jni_register(aprctrl_event* f)
```

Function: registration via the callback function provided by parameter, and this callback function conducts the processing of event code.

Parameter: f is a struct pointer, and this struct includes the unique function pointer member variables.

Return value: none

```c
uint16_t get_err_state(boardname e boardname)
```

Function: acquire the software/firmware runtime error code

Parameter: board name refers to the software/firmware ID
Parameter is one of the following values:

- Host computer ID
- Motor board ID
- Central board ID
- BMS ID

Return value: error code

```c
int16_t get_bat_soc(void)
```

Function: acquire percentage of battery remaining capacity

Parameter: none

Return value: percentage of battery remaining capacity (1~100)

```c
int16_t get_bat_charging(void)
```

Function: inquire whether the battery is in charging state

Parameter: none

Return value: 0: not in charging state
1: in charging state
int16 get_bat_mvol(void)

Function: acquire real-time voltage of battery

Parameter: none

Return value: voltage value, unit mV

int16 get_bat_mcurrent(void)

Function: acquire real-time current of battery

Parameter: none

Return value: current value, unit mA

int16 get_bat_temp(void)

Function: acquire battery temperature

Parameter: none

Return value: temperature value, unit degree Celsius

int16 get_chassis_work_model(void)

Function: acquire working state of chassis motor
Parameter: none

Return value: 1: motor in augmentation;
0: motor not in augmentation

#include<stdio.h>

int16_t get_chassis_load_state(void)

Function: acquire setting value of chassis based on controlling parameter of different loading

Parameter: none

Return value:
0: no-load control parameter;
1: full load controlling parameter

#include<stdio.h>

int16_t get_chassis_mode(void)

Function: acquire working mode of chassis finite state machine (FSM)

Parameter: none

Return value: 0 locking mode;
1 vehicle control mode;

2 pushing mode;

3 emergency stop mode;

4 error mode

int16 t_get_ctrl_cmd_src(void)

Function: acquire command origin of motor chassis control

Parameter: none

Return value:

0: control vehicle with remote controller;

1: control vehicle with host computer

int16 t_get_vehicle_meter(void)

Function: acquire the mileage since the chassis is power up

Parameter: none

Return value: mileage value, unit meter
uint16_t get_host_version(void)

Function: acquire the host computer software version

Parameter: none

Return value: host computer software version number

uint16_t get_chassis_central_version(void)

Function: acquire the central board firmware version

Parameter: none

Return value: the central board firmware version number

uint16_t get_chassis_motor_version(void)

Function: acquire the motor board firmware version

Parameter: none

Return value: the motor board firmware version number

int16_t get_line_forward_max_vel_fb (void)
Function: acquire the forward speed limiting feedback value of the chassis

Parameter: None

Return value: the forward speed limiting feedback value of the chassis

```c
int16_t get_line_backward_max_vel_fb (void)
```

Function: acquire the backward speed limiting feedback value of the chassis

Parameter: None

Return value: the backward speed limiting feedback value of the chassis

```c
int16_t get_angular_max_vel_fb (void)
```

Function: acquire the angular speed limiting feedback value of the chassis

Parameter: None
Return value: the angular speed limiting feedback value of the chassis

```c
int16_t getIapTotalProgress (void)
```

Function: Get the progress of IAP upgrades

Parameter: None

Return value:

-1: IAP upgrade failed

0: IAP upgrades are idle or started or interrupted

100: IAP upgrade completed

Other: Percentage of IAP upgrade progress

```c
void iapCentralBoard (void)
```

Function: IAP upgrade of the central board firmware of the chassis

Parameter: None

Return value: none
Note: You need to place the central board firmware "central.bin" in the path of "/sdcard/firmware/" in advance.

```c
void iapMotorBoard (void)
```

Function:  IAP upgrade of the motor board firmware of the chassis  
Parameter:  None  
Return value: none  

Note: You need to place the motor board firmware "motor.bin" in the path of "/sdcard/firmware/" in advance.

```c
bool isHostIapOver (void)
```

Function:  Query if the IAP upgrade process has ended  
Parameter:  None  
Return value: true: the IAP completes or fails or is interrupted  
False: IAP not started or in progress  

```c
Int16_t getHostIapResult (void)
```
Function: acquire the reason for the end of IAP

Parameter: None

Return value:

3: IAP completes
4: IAP fails
5: IAP is interrupted

Others: IAP not started or in progress

Int16_t getHostIapErrorCode (void)

Function: Gets the error code for IAP failure

Parameter: None

Return value: the error code for IAP failure

int16_t get_chassis_hang_mode(void)

Function: Gets the setting state of chassis hang_mode

Parameter: None

Return value:
1: The chassis is in hang_mode

0: The chassis is not in hang_mode

int16_t get_charge_mos_ctrl_status (void)

Function: Gets the status of switch for charging MOS on the central board.

Parameter: None

Return value:

1: The MOS opened

0: The MOS closed

void set_cmd_vel(double linear_x, double angular_z)

Function: set up the command value of chassis target speed, which needs to be regular transmit once the chassis is enabled. It will be determined as communication failure if the chassis can’t receive the command value in continuous 150ms in controlling mode.

Parameter: linear_x: linear velocity command value, unit m/s;

angular_z: angular velocity command value, unit rad/s
Return value: none

```c
void set_line_forward_max_vel(double linearforwardmax_x)
```

Function: set up the max forward linear velocity value of chassis.

Parameter: linearforwardmax_x: max forward linear velocity value of chassis, unit m/s, range 0–3

Return value: none

```c
void set_line_backward_max_vel(double linearbackwardmax_x)
```

Function: set up the max backward linear velocity value of chassis.

Parameter: linearbackwardmax_x: max backward linear velocity value of chassis, unit m/s, range –2–0

Return value: none

```c
void set_angular_max_vel(double angularmax_z)
```

Function: set up the max angular velocity command value of chassis.
Parameter: angularmaxz: the max angular velocity command value, unit rad/s, range 0–3

Return value: none

void set_enable_ctrl(uint16 enableflag)

Function: set up to enable the chassis to control the vehicle.

Parameter: enable_flag:

1 enable the vehicle control;
0 exit the vehicle control

Return value: none

void set_smart_car_serial(const char *serialno)

Function: set up the terminal name of serial port of host computer, e.g. ttyUSB0.

Parameter: serial_no: terminal name of serial port, under the path /dev/ by default, e.g. “ttyUSB0”

Return value: none
void set_comu_interface (comu_choice_e comu_choice)

Function: Set up the communication interface between the host computer and the chassis, including serial communication and CAN communication

Parameter: comu_choice:

‘comu_serial’ Use a serial port for communication
‘comu_can’ Use a CAN port for communication

Return value: none

void set_chassis_load_state(int16 newLoadSet)

Function: set up the parameter of chassis control based on the different chassis load.

Parameter: newLoadSet:

0: no-load parameter;
1: full load parameter

Return value: none
void set_chassis_poweroff (void)

Function: chassis power off controlled by host computer.

Parameter: none

Return value: none

void set_remove_push_cmd(void)

Function: when the chassis is in the pushing mode, it can exit this mode under the control of host computer.

Parameter: none

Return value: none

void setHostlapCanceled (void)

Function: Interrupt the IAP upgrade process.

Parameter: none

Return value: none
void set_chassis_hang_mode(int16_t enterHand)

Function: The chassis is configured to be in hang test mode. When in hang_ode, the chassis can be controlled normally.

Parameter: enterHand:

1: Config the chassis to be in hang_mode;

0: Config the chassis to be in non hang_mode.

Return value: none

void set_charge_mos_ctrl (bool on)

Function: Sets the switch for charging MOS on the central board.

Parameter: enterHand:

1: Turn on the charging MOS switch;

0: Turn off the charging MOS switch.

Return value: none