# Talis 30 MJ Rack Management System

**Introduction**

Talis 30 MJ Rack Management is a device that capable to capture data from up to 8 Battery Packs. In this document we will refer Talis 30 MJ Rack Management System as RMS. RMS implement a REST API as an interface to exchange data thus easily integrate into systems.

**How it works**

When RMS powered up, the first thing it does is connect to an access point with predefined SSID. Once it is connected, RMS will setting up a webserver as a way for other devices to communicate using REST API. There are many features which is provided and each feature has its own endpoint. The usage of each endpoints will be explained later. To start the main routine of RMS, there are 2 important steps that user must take:

1. Addressing
    - User must do addressing first before doing any other tasks, this will ensure that RMS could get the exact number of battery pack. This will also write the address into BMS on each pack to avoid conflicting data during transmission since RMS implements a half-duplex data transmission using RS-485
    - Addressing may take some time, wait for around 2 – 3s before doing any other tasks or user can check the progress using get-addressing-status endpoint. The later method is more preferable because it has more accurate timing. Addressing is a crucial step because it determines the flow of data, make sure that this step is not disturbed by any other tasks / skipped

2. Data Collection
    - Once the addressing is completed, user can begin a data collection from each battery pack by send a correspond command to set-data-collection endpoint. It is better to make sure that the number of pack match with the result from get-addressing-status
    - When data collection is started, RMS will update its own register with the data from each BMS. User can get all the data using get-cms-data endpoints or a single data using get-single-cms-data. The non-updated data will have default value and should be eliminated from processing

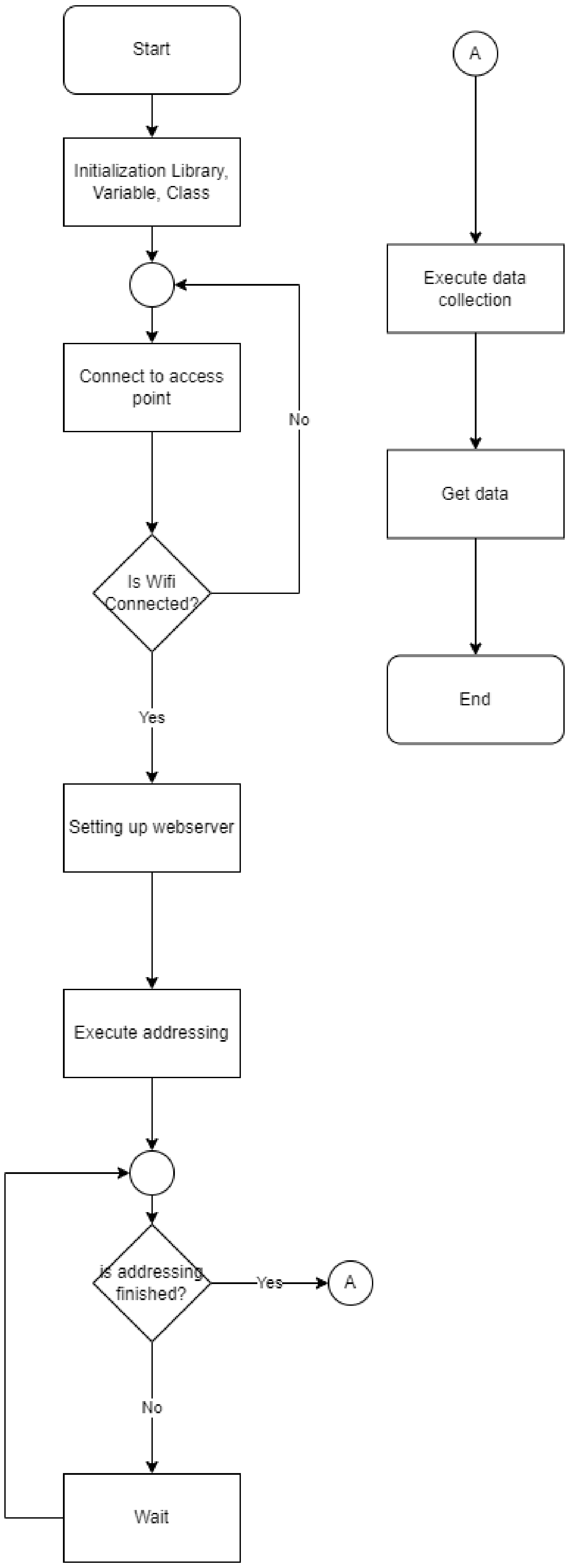

*Figure 1. RMS Flowchart*

**REST API**

RMS implements a REST API to exchange data. This will make user could easily integrate it into their own systems. RMS implements a HTTP POST and HTTP GET method, and doesn't support HTTPS right now, make sure to send a request using HTTP and not the later. The list of each endpoints of POST and GET method are described below

**GET Method**

| No | Endpoint | Method | Param | Usage |
|---|---|---|---|---|
| 1 | ./get-single-cms-data | GET | bid | To get the single cms data defined by bid |
| 2 | ./get-cms-data | GET | - | To get 8 cms data |
| 3 | ./get-device-general-info | GET | - | To get rms info such as code, ip, mac, etc |
| 4 | ./get-alarm-parameter | GET | - | To get the alarm parameter such as min-max voltage & temperature |
| 5 | ./get-addressing-status | GET | - | To get the number of connected BMS |

Note : fill the "." With ip of device (ex : http://192.168.2.200/get-cms-data)

Endpoint usage and explanation for GET Method :

1. get-single-cms-data
   - Send a HTTP GET request with parameter bid (ex : 192.168.2.200/get-single-cms-data?bid=1)
   - RMS will respond with formatted JSON :

```
{
  "msg_count": 13,
  "frame_name": "FRAME-32-NA",
  "cms_code": "CMS-32-NA",
  "base_code": "BASE-32-NA",
  "mcu_code": "MCU-32-NA",
  "site_location": "SITE-32-NA",
  "bid": 1,
  "vcell": [
    3644,
    3851,
    3917,
    49,
    3657,
    3722,
    3692,
    51,
    49,
    3917,
    3589,
    3741,
    51,
    51,
    3684,
    3643,
    3983,
    528,
    46,
    3975,
    3530,
    3930,
    46,
    48,
    3863,
    3747,
    3657,
    47,
    47,
    3892,
    3633,
    3666,
    3663,
    48,
    3682,
    4003,
    3634,
    3489,
    48,
    3627,
    3686,
    3693,
    3633,
    47,
    3660
  ],
  "temp": [
    24000,
    27100,
    29500,
    24000,
    27100,
    32800,
    27100,
    27100,
    27100
  ],
  "pack": [
    37420,
    34763,
    44075
  ],
  "wake_status": 1,
  "door_status": 0
}
```

- msg_count : ideally, this value will be different each time, if this value is same for a very long time, then there is problem with RS485 communication
- frame_name : the serial number of pack
- cms_code : the serial number of cms
- base_code : the serial number of base pcb
- mcu_code : the serial number of MCU
- site_location : the location its placed
- bid : the id of pack, it shows which stack. Smaller id means its position is at the bottom of pack
- vcell : the cell data voltage
- wake_status : the status of CMS, 1 means its in wake state
- door_status : the door status

2. get-cms-data
    - Send a HTTP GET request without any parameter
    - RMS will respond with formatted JSON :

```
{
  "cms-data" : [
  {
      1*
  },
  {
      2*
  },
  {
      3*
  },
  {
      4*
  },
  {
      5*
  },
  {
      6*
  },
  {
      7*
  },
  {
      8*
  }
  ]
}
```

    - Each asterisk(*) field will be filled with formatted JSON text like when requested single cms data

3. get-device-general-info
    - Send a HTTP GET request without any parameter
    - RMS will respond with formatted JSON :

```
{
  "rms_code": "RMS-000000030",
  "ver": "1.0.0",
  "ip": "192.168.2.212",
  "mac": "0C:B8:15:D8:E7:F8",
  "dev_type": "RMS"
}
```

4. get-alarm-parameter
    - Send a HTTP GET request without any parameter
    - RMS will respond with formatted JSON :

```
{
  "vcell_max": 3700,
  "vcell_min": 3000,
  "temp_max": 80000,
  "temp_min": 20000
}
```

5. get-addressing-status
    - Send a HTTP GET request without any parameter
    - RMS will respond with formatted JSON

```
{
  "num_of_device": 8,
  "device_address_list": [
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8
  ],
  "status": 0
}
```

- num_of_device : the number of detected devices
- device_address_list : the list of detected bid
- Status : 0 means the addressing is still ongoing, 1 means the addressing is finished

**POST Method**

| No | Endpoint | Method | Usage |
|---|---|---|---|
| 1 | ./set-balancing | POST | Set cell balancing |
| 2 | ./set-addressing | POST | Activate the RMS addressing procedure |
| 3 | ./set-alarm | POST | Activate output alarm of RMS |
| 4 | ./set-data-collection | POST | Start data capture of RMS |
| 5 | ./set-led | POST | Set the led of each BMS |
| 6 | ./set-alarm-parameter | POST | Set max-min alarm cell voltage & temperature |
| 7 | ./set-hardware-alarm | POST | Activate hardware alarm |
| 8 | ./set-sleep | POST | Sleep the CMS |
| 9 | ./set-wakeup | POST | Wake the CMS |
| 10 | ./restart-cms | POST | Restart the CMS |
| 11 | ./restart | POST | Restart the RMS |
| 12 | ./set-rms-code | POST | Set the RMS serial number |
| 13 | ./set-frame | POST | Set the frame serial number of BMS |
| 14 | ./set-cms-code | POST | Set CMS code |
| 15 | ./set-base-code | POST | Set base code |
| 16 | ./set-mcu-code | POST | Set mcu code |
| 17 | ./set-site-location | POST | Set site location |

Note : fill the "." With ip of device (ex : http://192.168.2.200/set-balancing)

Endpoint usage and explanation for POST Method :

1. set-balancing
    - To activate balancing feature on RMS, send a HTTP POST with following JSON format:

```
{
  "balancing_command": {
    "bid": 1,
    "sbal": 1,
    "cball": [
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0
      ]
    }
}
```

- Each index on cball array represent the cell position, to set balancing on first cell, set the first array index cball[0] to 1. To disable balancing, set the correspond array index to 0
- When the POST request successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

  - Status 1 indicate that the request is successfully received, any other value indicate abnormality

2. set-addressing
   - To begin addressing on BMS, send the HTTP POST request to RMS with following JSON format :

```
{
  "addr" : 1
}
```

   - Please wait a while, preferebaly around 2 – 3s before sending another request since the RMS need some time to do an addressing
   - The addressing status can be checked on ./get-addressing-status. It lists all the detected device and field “status” represent whether the addressing process is finished(1) or still ongoing(0)
   - When POST request successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

3. set-alarm
   - To set the alarm output (buzzer, relay, etc), send the HTTP POST request to RMS with following JSON format :

```
{
  "alarm": {
    "buzzer": 0,
    "power_relay": 0,
    "batt_relay": 0
  }
}
```

- When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

4. set-data-collection
   - To begin data capture from BMS, send the HTTP POST request to RMS with following JSON format :

```
{
  "data_collection" : 1
}
```

   - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

5. set-led
   - To set the led on each BMS, send HTTP POST request to RMS with following JSON format :

```
{
  "bid": 1,
  "ledset": 1,
  "num_of_led" : 8,
  "led_rgb" : [
       [0,0,0],
       [0,0,0],
       [0,0,0],
       [0,0,0],
       [0,0,0],
       [0,0,0],
       [0,0,0],
       [0,0,0]
     ]
}
```

   - bid : determine the pack number id
   - ledset : 1 to execute, 0 to cancel
   - num_of_led : number of led in 1 pack
   - led_rgb : array contains the RGB value, each array index represent the position of led (ex : to set the first led into green, set the first array to [0, 255, 0])
   - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

6. set-alarm-parameter

- To set alarm parameter such as max – min voltage, max – min temperature, send HTTP POST request to RMS with following JSON :

```
{
  "vcell_max" : 3600,
  "vcell_min" : 3000,
  "temp_max" : 80000,
  "temp_min" : 20000
}
```

- When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

7. set-hardware-alarm
   - To enable / disable internal hardware alarm, send a HTTP POST request to RMS with following JSON format :

```
{
  "hardware_alarm" : 0
}
```

   - Hardware alarm feature is an internal abnormality checker, if it is enabled RMS will perform a routine check on cell abnormality (voltage and temperature). If an abnormality happens, the RMS will trigger output on buzzer pin. Default value is 0 (disabled) and the value will not be retained when powered off
   - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

8. set-sleep
   - To make CMS sleep, send HTTP POST request to RMS with following JSON format :

```
{
  "bid": 1,
  "shutdown" : 1
}
```

   - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

9. set-wakeup
   - To wake CMS, send HTTP POST request to RMS with following JSON format :

```
{
  "bid": 1,
  "wakeup" : 1
}
```

   - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

10. restart-cms
    - To restart CMS, send HTTP POST request to RMS with following JSON format :

```
{
  "bid": 1,
  "restart" : 1
}
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
{
  "status": 1
}
```

11. Restart
    - To restart RMS, send HTTP POST request to RMS with following JSON format :

```
    {
      "restart" : 1
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

12. set-rms-code
    - To set RMS code, send HTTP POST request to RMS with following JSON format :

```
    {
      "rms_code_write" : 1,
      "rms_code" : "RMS-000000030"
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

13. set-frame
    - To set frame code, send HTTP POST request to RMS with following JSON format :

```
    {
      "bid" : 1,
      "frame_write" : 1,
      "frame_name" : "611800016"
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

14. set-cms-code
    - To set cms code, send HTTP POST request to RMS with following JSON format :

```
    {
      "bid" : 1,
      "cms_write" : 1,
      "cms_code" : "0622400013"
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

15. set-base-code
    - To set base code, send HTTP POST request to RMS with following JSON format :

```
    {
      "bid" : 1,
      "base_write" : 1,
      "base_code" : "b0622400013"
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

16. set-mcu-code
    - To set base code, send HTTP POST request to RMS with following JSON format :

```
    {
      "bid" : 1,
      "mcu_write" : 1,
      "mcu_code" : "m0622400013"
    }
```

    - When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```

17. set-site-location
    - To set base code, send HTTP POST request to RMS with following JSON format :

```
    {
      "bid" : 1,
      "site_write" : 1,
      "site_location" : "l0622400013"
    }
```

- When POST request is successfully received, RMS will respond with JSON format :

```
    {
      "status": 1
    }
```