

# Pavo2 SDK 编程指南 (2.1.0)

上海星秒光电科技有限公司  
(版权所有, 翻版必究)

# 目 录

|                                        |    |
|----------------------------------------|----|
| 1 概述 .....                             | 6  |
| 1.1 系统要求 .....                         | 6  |
| 1.2 Pavo2 体系结构 .....                   | 6  |
| 2 核心数据结构 .....                         | 8  |
| 3 接口说明 .....                           | 10 |
| 3.1 pavo2_driver .....                 | 11 |
| 3.2 set_local_address .....            | 11 |
| 3.3 set_conn_protocol .....            | 11 |
| 3.4 pavo2_open .....                   | 12 |
| 3.5 pavo2_is_open .....                | 12 |
| 3.6 pavo2_close .....                  | 12 |
| 3.7 get_scans_available .....          | 13 |
| 3.8 get_ptp_deviations_available ..... | 13 |
| 3.9 get_scanned_data .....             | 13 |
| 3.10 recv_point_cloud .....            | 15 |
| 3.11 recv_ptp_deviation_data .....     | 15 |
| 3.12 get_device_sn .....               | 15 |
| 3.13 get_device_pn .....               | 16 |
| 3.14 get_sdk_ver .....                 | 16 |
| 3.15 get_os_ver .....                  | 16 |
| 3.16 get_firmware_ver .....            | 16 |
| 3.17 get_os_error .....                | 17 |
| 3.18 get_error_code .....              | 17 |
| 3.19 enable_data .....                 | 17 |
| 3.20 set_intensity_mode .....          | 18 |
| 3.21 get_intensity_mode .....          | 18 |
| 3.22 set_angle_range .....             | 18 |
| 3.23 get_angle_range .....             | 18 |
| 3.24 get_max_angle_range .....         | 19 |
| 3.25 set_motor_speed .....             | 19 |
| 3.26 get_motor_speed .....             | 19 |
| 3.27 get_motor_speed_enable .....      | 20 |
| 3.28 set_angle_resolution .....        | 20 |
| 3.29 get_angle_resolution .....        | 20 |
| 3.30 get_angle_resolution_enable ..... | 20 |
| 3.31 set_device_ip .....               | 21 |
| 3.32 get_device_ip .....               | 21 |
| 3.33 set_tcp_port .....                | 21 |
| 3.34 get_tcp_port .....                | 21 |
| 3.35 set_device_mask .....             | 22 |
| 3.36 get_device_mask .....             | 22 |
| 3.37 set_device_gateway .....          | 22 |

|      |                                      |    |
|------|--------------------------------------|----|
| 3.38 | get_device_gateway .....             | 22 |
| 3.39 | get_device_mac .....                 | 23 |
| 3.40 | set_os_restart .....                 | 23 |
| 3.41 | set_echo_mode .....                  | 23 |
| 3.42 | get_echo_mode .....                  | 23 |
| 3.43 | set_outdoor_mode .....               | 24 |
| 3.44 | get_outdoor_mode .....               | 24 |
| 3.45 | set_calibrate_horizontal_angle ..... | 24 |
| 3.46 | get_calibrate_horizontal_angle ..... | 25 |
| 3.47 | get_max_angle_distance .....         | 25 |
| 3.48 | enable_tail_filter .....             | 25 |
| 3.49 | enable_front_filter .....            | 25 |
| 3.50 | enable_far_hr_filter .....           | 26 |
| 3.51 | enable_near_hr .....                 | 26 |
| 3.52 | is_near_hr_enabled .....             | 26 |
| 3.53 | is_near_hr_supported .....           | 26 |
| 3.54 | set_warn_angle_range .....           | 27 |
| 3.55 | get_warn_angle_range .....           | 27 |
| 3.56 | set_warn_intensity_distance .....    | 27 |
| 3.57 | get_warn_intensity_distance .....    | 28 |
| 3.58 | set_warn_point_number .....          | 28 |
| 3.59 | get_warn_point_number .....          | 28 |
| 3.60 | enable_ptp .....                     | 28 |
| 3.61 | is_ptp_enabled .....                 | 29 |
| 3.62 | is_ptp_supported .....               | 29 |
| 3.63 | get_temperature .....                | 29 |
| 3.64 | get_heartbeat .....                  | 30 |

## 修订历史

| 内容                                                                                      | 版本    | 修订人 |
|-----------------------------------------------------------------------------------------|-------|-----|
| 1. 文档创建                                                                                 | 1.0.0 | 张洪波 |
| 1. 去掉转速设置接口<br>2. 增加二回波模式设置和获取接口                                                        | 1.0.1 | 张洪波 |
| 1. 修正获取 pavo_response_pcd_t 格式点云数据不正确的问题<br>2. 修正 pavo2_is_open 接口在断开连接后仍返回 true 的问题    | 1.0.2 | 张洪波 |
| 1. 兼容雷达 360 度检测范围                                                                       | 1.0.3 | 张洪波 |
| 1. 修复点云数据乱序的问题<br>2. 修复部分指令回复接受不到的问题                                                    | 1.0.4 | 张洪波 |
| 1. 修复噪点问题                                                                               | 1.0.5 | 张洪波 |
| 1. 修复关闭点云强度后部分点云信息错误的问题                                                                 | 1.0.6 | 张洪波 |
| 1. 添加高反滤波处理                                                                             | 1.0.7 | 张洪波 |
| 1. 增加室外增强模式<br>2. 修复校验模式点云不正确 bug                                                       | 1.0.8 | 张洪波 |
| 1. 新增读取最大测距接口<br>2. 新增读取转速使能接口<br>3. 新增读取角分辨率使能接口<br>4. 新增读取最大角度范围使能接口<br>5. 新增读写水平校准接口 | 1.0.9 | 张洪波 |
| 1. 增加时间延时接口                                                                             | 1.1.0 | 张洪波 |
| 1. 增加障碍物报警相关接口<br>2. 修复滤波时距离为 0 的时候强度不为 0 的 bug                                         | 1.1.1 | 张洪波 |

|                                                                   |       |     |
|-------------------------------------------------------------------|-------|-----|
| 1. 去掉对 Nigle 算法的禁用<br>2. 发查询命令之前，删除缓存中同一个寄存器的所有查询回应               | 1.1.2 | 张洪波 |
| 1. 优化点云解析逻辑处理                                                     | 1.1.3 | 张洪波 |
| 1. 修复接收缓冲区内容较大时解析错误                                               | 1.1.4 | 张洪波 |
| 1. 添加 ptp 相关接口                                                    | 1.2.0 | 郭强  |
| 1. 添加获取温度接口                                                       | 1.2.1 | 郭强  |
| 1. 增加绑定本地 ip 地址和端口接口<br>2. 增加设置/获取雷达 mask 和 gateway 接口            | 1.2.2 | 郭强  |
| 1. 增加近端高反识别接口<br>2. 将接口 enable_hr_filter 更改为 enable_far_hr_filter | 1.2.3 | 郭强  |
| 1. 增加 set_conn_protocol 接口，提供用户选择连接雷达采用协议的功能。                     | 2.0.0 | 郭强  |
| 1. 增加接口 is_ptp_supported<br>2. 增加接口 is_near_hr_supported          | 2.1.0 | 郭强  |

# 1 概述

本文档描述了 SIMINICS Pavo2 SDK 的功能与使用方法，与 Pavo2 Lidar 同时发布。

本 SDK 不能用于对其他设备或系统的控制，请勿移作他用。

## 1.1 系统要求

本 SDK 为源码发布，请用户自行集成到目标系统。

SDK 使用了同时支持 Windows 和 Linux 操作系统。

SDK 使用了多线程和 Socket 技术，编译时请确保操作系统支持多线程和 Socket 库。

SDK 由 C++编写，遵循 C++11 标准，请确保编译器支持 C++11 标准。

## 1.2 Pavo2 体系结构

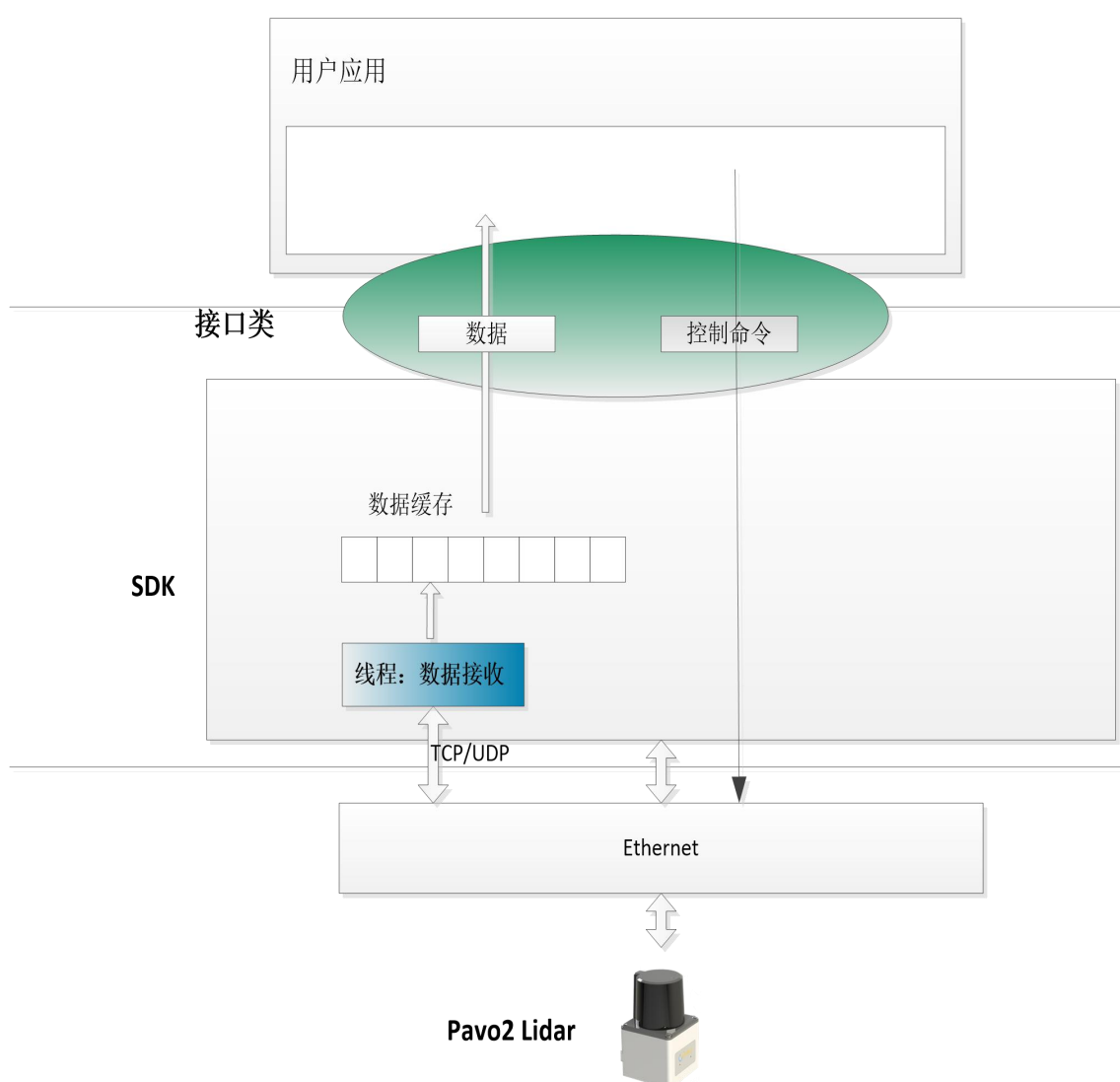


图 1: Pavo2 系统基本结构

Pavo2 数据采集系统从软件角度包括以下几个部分:

1. **Pavo2 Lidar:** 采集环境数据。
2. **Ethernet:** Pavo2 Lidar 通过 Ethernet 与上位机连接。Pavo2 Lidar 采集到的数据经由 TCP/UDP 数据包发送到上位机。
3. **SDK:** 上位机程序可以通过 SDK 提供的接口接收 Pavo2 Lidar 采集到的数据或对 Pavo2 Lidar 进行控制:
  - 1) 获取设备数据线程: 读取设备数据, 保证数据传输速度。  
SDK 通过 TCP/UDP 与雷达进行连接, 并启动一专门线程接收雷达数据。
  - 2) 数据缓存: 接收到的数据先缓存在内存中。  
用户通过 SDK 接口类获得的数据都是最新一帧的数据, 该帧以前的数据如未及时读取, 会被丢弃。

视 Pavo2 型号不同, 雷达与上位机的数据通信有两种方式

1. 只提供 TCP 连接。
  2. 同时支持 TCP 和 UDP 连接, 用户可以选择 TCP 或 UDP 连接方式。
- 具体型号支持何种协议, 请咨询销售人员。

## 2 核心数据结构

```
enum class PAVO2_PROTO_TYPE : int
{
    DEFAULT = 0,
    TCP = 1,
    UDP = 2
};

typedef struct pavo_response_scan
{
    uint16_t angle;
    uint16_t distance;
    uint8_t intensity;
} __DATA_ALIGN__ pavo_response_scan_t;

typedef struct pavo_response_pcd
{
    double x;
    double y;
    double z; //fixed as 0 for single-laser device
    uint8_t intensity;
} __DATA_ALIGN__ pavo_response_pcd_t;

typedef struct pavo_packet_info_pcd
{
    vector<pavo_response_pcd_t> vec_buff;
    uint32_t time_stamp;
    uint32_t time_delay;
} __DATA_ALIGN__ pavo_packet_info_pcd_t;

typedef struct pavo_packet_info_scan
{
    vector<pavo_response_scan_t> vec_buff;
    uint32_t time_stamp;
    uint32_t time_delay;
} __DATA_ALIGN__ pavo_packet_info_scan_t;

typedef struct pavo_point_cloud_scan
{
    uint32_t packet_id;
    bool ptp_en;
    uint16_t process_delay;
    uint16_t angle_resolution;
    uint16_t start_angle;
    uint16_t end_angle;
    vector<pavo_response_scan_t> data;
    uint64_t time_stamp;
} pavo_point_cloud_scan_t;

typedef struct pavo_ptp_deviation
{
    int64_t last_sync_time;
    int64_t time_offset;
} pavo_ptp_deviation_t;
```

PAVO2\_PROTO\_TYPE: 选择 pavo2\_driver 连接 Pavo2 Lidar 所使用的协议。

pavo\_response\_scan\_t: 角度和距离坐标。

angle: 单位为 0.01 度。

distance: 单位为 0.002 米。

intensity: 值在 0~4500 之间，为一相对值。

pavo\_response\_pcd\_t: 笛卡尔坐标。

x,y,z: 单位为 0.002 米。

intensity: 值在 0~4500 之间，为一相对值。

pavo\_packet\_info\_pcd\_t: 一圈点云数据

vec\_buff: 点云数据集合

time\_stamp: 时间戳

time\_delay: 时间延时

pavo\_packet\_info\_scan\_t: 一圈点云数据

vec\_buff: 点云数据集合

time\_stamp: 时间戳

time\_delay: 时间延时

这些数据结构定义在 pavo2\_types.h 头文件中。

pavo\_point\_cloud\_scan\_t: 一圈点云数据，包括点云头部信息

packet\_id: 一圈点云数据 id

ptp\_en: 是否开启 ptp 功能

process\_delay: 点云在雷达固件中的处理延时

angle\_resolution: 角分辨率

start\_angle: 点云起始角度

end\_angle: 点云结束角度

Data: 点云数据

time\_stamp: 点云时间戳

pavo\_ptp\_deviation\_t: 雷达与 ptp 主机之间的时间偏差信息

last\_sync\_time: 雷达与 ptp 主机之间的上一次同步时间。

time\_offset: 雷达与主机之间的时间偏差。

### 3 接口说明

用户通过类“class pavo2\_driver”对 Pavo2 Lidar 进行访问。该类定义在 pavo2\_driver.h 头文件中，SDK 的版本号通过 SDK\_VER 宏定义进行获取，主要函数有：

```
class pavo2_driver
{
public:
    pavo2_driver();
    ~pavo2_driver();
    void set_local_address(std::string local_ip, uint16_t local_port);
    void set_conn_protocol(PAVO2_PROTO_TYPE proto_type);

    bool pavo2_open(string lidar_ip, uint16_t lidar_port, uint32_t timeout = 0);
    bool pavo2_is_open();
    bool pavo2_close();
    std::size_t get_scans_available() const;
    std::size_t get_ptp_deviations_available() const;
    bool get_scanned_data(pavo_response_scan_t* data_buffer, uint16_t count, uint16_t timeout = 0);
    bool get_scanned_data(std::vector<pavo_response_scan_t>& vec_buff, uint16_t timeout = 0);
    bool get_scanned_data(pavo_response_pcd_t* data_buffer, uint16_t count, uint16_t timeout = 0);
    bool get_scanned_data(std::vector<pavo_response_pcd_t>& vec_buff, uint16_t timeout = 0);
    bool recv_point_cloud(pavo_point_cloud_scan_t& scan, uint16_t timeout = 0);
    bool recv_ptp_deviation_data(pavo_ptp_deviation_t& ptp_deviation, uint16_t timeout = 0);

    bool get_device_sn(string & sn);
    bool get_device_pn(string & pn);
    bool get_sdk_ver(string & ver);
    bool get_os_ver(string & ver);
    bool get_firmware_ver(string & ver);
    bool get_os_error(uint32_t& errCode, uint8_t iIndex = 0);
    int get_error_code();
    bool enable_data(bool en);
    bool set_intensity_mode(bool en);
    bool get_intensity_mode(bool & en);
    bool set_angle_range(uint16_t start, uint16_t end);
    bool get_angle_range(uint16_t & start, uint16_t & end);
    bool set_motor_speed(uint8_t val);
    bool get_motor_speed(uint8_t & val);
    bool get_motor_speed_enable(uint32_t & val);
    bool set_angle_resolution(uint16_t val);
    bool get_angle_resolution(uint16_t & val);
    bool get_angle_resolution_enable(uint32_t & val);

    bool set_device_ip(string ip);
    bool get_device_ip(string & ip);
    bool set_tcp_port(uint16_t val);
    bool get_tcp_port(uint16_t & val);
    bool set_device_mask(const std::string& mask);
    bool get_device_mask(std::string& mask);
    bool set_device_gateway(const std::string& gateway);
    bool get_device_gateway(std::string& gateway);
    bool get_device_mac(std::string& mac);

    bool set_os_restart();
    bool set_echo_mode(bool en);
    bool get_echo_mode(bool& en);
    bool set_outdoor_mode(bool en);
    bool get_outdoor_mode(bool& en);

    bool set_calibrate_horizontal_angle(uint32_t angle);
    bool get_calibrate_horizontal_angle(uint32_t& angle);
    bool get_max_angle_distance(uint32_t & val);
    bool enable_tail_filter(uint16_t method);
    bool enable_front_filter(uint16_t iLevel);
    bool enable_far_hr_filter(uint16_t hrLevel);
```

```

    bool enable_near_hr_filter(bool en);
    bool get_near_hr_filter(bool &en);
    bool set_warn_angle_range(uint16_t start,uint16_t end);
    bool get_warn_angle_range(uint16_t& start,uint16_t& end);
    bool set_warn_intensity_distance(uint16_t distance,uint16_t intensity);
    bool get_warn_intensity_distance(uint16_t& distance,uint16_t& intensity);
    bool set_warn_point_number(uint16_t number);
    bool get_warn_point_number(uint16_t& number);
    bool enable_ptp(bool en);
    bool is_ptp_enabled(bool &en);
    bool get_temperature(float& temperature);
    bool get_heartbeat();

    //其后省略.....
}

```

### 3. 1 pavo2\_driver

Pavo2 Driver 的构造函数。

```
pavo2_driver();
```

SDK 核心类，在创建时，会启动一个 TCP 通信节点，用于和 Pavo2 雷达通信。

采用雷达被动上传数据模式。

### 3. 2 set\_local\_address

绑定和雷达连接的本地 IP 地址和端口。

```
void set_local_address(const std::string& local_ip, uint16_t local_port);
```

参数：

local\_ip: 输入参数，本地 IP 地址。

返回值：

无。

### 3. 3 set\_conn\_protocol

选择所创建的 pavo2\_driver 连接 Pavo2 Lidar 所使用的协议。

```
void set_conn_protocol(PAVO2_PROTO_TYPE proto_type);
```

参数：

proto\_type: 输入参数，选择所需要使用的协议。

DEFAULT:使用默认协议，当前默认协议是 TCP；

TCP:使用 TCP 协议；

UDP:使用 UDP 协议。

返回值：

无。

PAVO2\_PROTO\_TYPE 定义如下：

```

enum class PAVO2_PROTO_TYPE : int
{
    DEFAULT = 0,

```

```
TCP = 1,  
UDP = 2  
};
```

### 3.4 pavo2\_open

和 pavo2 雷达建立通信连接。

```
bool pavo2_open(const std::string& lidar_ip, uint16_t lidar_port, uint32_t timeout = 0);
```

参数：

**lidar\_ip**: 输入参数，雷达的 IP 地址。

**lidar\_port**: 输入参数，雷达的发送数据端口。

**timeout**: 输入参数，超时时间，单位 ms。默认值为 0。如果该值为 0，则该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有成功建立通信，则函数仍然返回，返回值为 false。

返回值：

是否成功建立连接：

**true**: 建立连接成功；

**false**: 建立连接失败或超时。

### 3.5 pavo2\_is\_open

判断是否与 Pavo2 Lidar 成功连接。常用于连接建立后，对连接通信是否仍然正常进行判断。

```
bool pavo2_is_open();
```

参数：

无。

返回值：

连接是否成功：

**true**: 连接成功；

**false**: 连接失败。

### 3.6 pavo2\_close

关闭和雷达的通信连接。

```
bool pavo2_close();
```

参数：

无。

返回值：

**true**: 关闭成功；

**false**: 关闭失败。

### 3.7 get\_scans\_available

获取 SDK 中缓存的点云圈数。

```
std::size_t get_scans_available() const;
```

参数：

无。

返回值：

缓存的点云圈数。

### 3.8 get\_ptp\_deviations\_available

获取 SDK 中缓存的 ptp 时间偏差信息数量。

```
std::size_t get_ptp_deviations_available() const;
```

参数：

无。

返回值：

缓存的 ptp 时间偏差信息数量。

### 3.9 get\_scanned\_data

该接口用于获取 Pavo2 Lidar 扫描到的数据，根据数据类型的不同，函数重载四次。

```
bool get_scanned_data(pavo_response_scan_t* data_buffer, uint16_t& count, uint16_t timeout = 0);
```

参数：

**data\_buffer:** 输出参数，用户提供的数据返回数组。用户须保证该参数的有效性。

**count:** 输入/输出参数，返回的数据个数。

输入：data\_buffer 大小；

输出：返回数据个数，如返回数据大于 data\_buffer 大小，则表示缓冲大小不足。

**timeout:** 输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数仍然返回，返回值为 false。

返回值：

是否获得有效数据：

true: 获得有效数据；

false: 获取数据失败。

```
bool get_scanned_data(std::vector<pavo_response_scan_t>& vec_buff, uint16_t timeout = 0);
```

参数：

**vec\_buff:** 输出参数，返回数据。

**timeout:** 输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数

数仍然返回, 返回值为 `false`;

返回值:

是否获得有效数据:

`true`: 获得有效数据;

`false`: 获取数据失败。

```
bool get_scanned_data(pavo_response_pcd_t* data_buffer, uint16_t& count, uint16_t timeout = 0);
```

参数:

`data_buffer`: 输出参数, 用户提供的数据返回数组。用户须保证该参数的有效性。

`count`: 输入/输出参数, 返回的数据个数。

输入: `data_buffer` 大小;

输出: 返回数据个数, 如返回数据大于 `data_buffer` 大小, 则表示缓冲大小不足。

`timeout`: 输入参数, 超时时间, 单位 `ms`。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 `false`。

返回值:

是否获得有效数据:

`true`: 获得有效数据;

`false`: 获取数据失败。

```
bool get_scanned_data(std::vector<pavo_response_pcd_t>& vec_buff, uint16_t timeout = 0);
```

参数:

`vec_buff`: 输出参数, 返回数据。

`timeout`: 输入参数, 超时时间, 单位 `ms`。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 `false`;

返回值:

是否获得有效数据:

`true`: 获得有效数据;

`false`: 获取数据失败。

```
bool get_scanned_data(pavo_packet_info_scan_t& packet_info_scan, uint16_t timeout = 0);
```

参数:

`packet_info_scan`: 输出参数, 用户提供的数据返回数组。用户须保证该参数的有效性。

`timeout`: 输入参数, 超时时间, 单位 `ms`。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 `false`。

返回值:

是否获得有效数据。

```
bool get_scanned_data(pavo_packet_info_pcd_t& packet_info_pcd, uint16_t timeout = 0);
```

参数:

**packet\_info\_pcd:** 输出参数, 用户提供的数据返回数组。用户须保证该参数的有效性。

**timeout:** 输入参数, 超时时间, 单位 ms。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 false。

返回值:

是否获得有效数据。

## 3.10 recv\_point\_cloud

该接口用于获取 Pavo2 Lidar 扫描到的点云数据, 与 `get_scanned_data`, 该接口获取的点云数据还包括头部信息。

```
bool recv_point_cloud(pavo_point_cloud_scan_t& scan, uint16_t timeout=0);
```

参数:

**scan:** 输出参数, 点云数据。

**timeout:** 输入参数, 超时时间, 单位 ms。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 false。

返回值:

是否获得有效数据:

**true:** 获得有效数据;

**false:** 获取数据失败。

## 3.11 recv\_ptp\_deviation\_data

获取 Pavo2 Lidar 与 ptp 主机之间的时间偏差信息。

```
bool recv_ptp_deviation_data(pavo_ptp_deviation_t& ptp_deviation, uint16_t timeout=0);
```

参数:

**ptp\_deviation:** 输出参数, ptp 时间偏差信息。

**timeout:** 输入参数, 超时时间, 单位 ms。如果该值为 0, 则在获得有效数据之前, 该函数会一直阻塞。如果该值大于零, 则如果超过该时间还没有获得有效数据, 则函数仍然返回, 返回值为 false。

返回值:

是否获得有效数据:

**true:** 获得有效数据;

**false:** 获取数据失败。

## 3.12 get\_device\_sn

获取设备 SN 序列号。

```
bool get_device_sn(std::string& sn);
```

参数:

sn: 获取到的设备 SN 序列号。

返回值:

true: 调用成功;

false: 调用失败。

### 3.13 get\_device\_pn

获取设备 PN 序列号。

```
bool get_device_pn(std::string& pn);
```

参数:

pn: 获取到的设备 PN 序列号,以十六进制显示。

返回值:

true: 调用成功;

false: 调用失败。

### 3.14 get\_sdk\_ver

获取 sdk 版本号。

```
bool get_sdk_ver(std::string& ver);
```

参数:

ver: 输出参数。Sdk 版本号。

返回值:

true: 调用成功;

false: 调用失败。

### 3.15 get\_os\_ver

获取雷达操作系统版本号。

```
bool get_os_ver(std::string& ver);
```

参数:

ver: 输出参数。雷达操作系统版本号。

返回值:

true: 调用成功;

false: 调用失败。

### 3.16 get\_firmware\_ver

获取雷达固件版本号。

```
bool get_firmware_ver(string& ver);
```

参数：

**ver:** 输出参数。雷达固件版本号。

返回值：

**true:** 调用成功；

**false:** 调用失败。

### 3.17 get\_os\_error

获取雷达操作系统错误码。

```
bool get_os_error(uint32_t& errCode, uint8_t iIndex = 0)
```

参数：

**errCode:** 输出参数。雷达操作系统错误码。

**iIndex :** 输入参数，0~3 表示获取对应序号的错误码，其他值表示获取全部错误码

返回值：

**true:** 调用成功。

**false:** 调用失败。

### 3.18 get\_error\_code

获取雷达连接网络错误。

```
int get_error_code();
```

参数：

无。

返回值：

网络错误码。

### 3.19 enable\_data

开启/停止 Pavo2 Lidar 数据传输。

```
bool enable_data(bool en);
```

参数：

**en:** 输入参数。

**true:** 开启数据传输；

**false:** 停止数据传输。

返回值：

**true:** 调用成功；

**false:** 调用失败。

## 3.20 set\_intensity\_mode

设置雷达数据是否带强度信息。

```
bool set_intensity_mode(bool en);
```

参数：

en: 输入参数。

true: 雷达数据带强度信息；

false: 雷达数据不带强度信息。

返回值：

true: 调用成功；

false: 调用失败。

## 3.21 get\_intensity\_mode

获取雷达数据是否带强度信息。

```
bool get_intensity_mode(bool& en);
```

参数：

en: 输出参数，雷达数据是否带强度信息。

true: 雷达数据带强度信息；

false: 雷达数据不带强度信息。

返回值：

true: 调用成功；

false: 调用失败。

## 3.22 set\_angle\_range

设置雷达传输角度范围。

```
bool set_angle_range(uint16_t start, uint16_t end);
```

参数：

start: 输入参数，起始角度。取值范围[4000, 32000]。单位是 0.01°。

end: 输入参数，终止角度。取值范围[4000, 32000]。单位是 0.01°。

注意：start 值不能大于 end。

返回值：

true: 调用成功；

false: 调用失败。

## 3.23 get\_angle\_range

获取传输的角度范围。

```
bool get_angle_range(uint16_t& start, uint16_t& end);
```

参数：

start: 输出参数。起始角度。

end: 输出参数。终止角度。

返回值：

true: 调用成功；

false: 调用失败。

## 3.24 get\_max\_angle\_range

获取雷达角度范围使能配置。

```
bool get_max_angle_range(uint32_t& val);
```

参数：

val: 输出参数，雷达角度范围使能配置。

返回值：

true: 调用成功；

false: 调用失败。

## 3.25 set\_motor\_speed

设置电机转速。

```
bool set_motor_speed(uint8_t val);
```

参数：

val: 输入参数，电机转速，单位 Hz。

返回值：

true: 调用成功；

false: 调用失败。

## 3.26 get\_motor\_speed

获取电机转速。

```
bool get_motor_speed(uint8_t& val);
```

参数：

val: 输出参数，电机转速，单位 Hz。

返回值：

true: 调用成功；

false: 调用失败。

### 3.27 get\_motor\_speed\_enable

获取电机转速可配置范围。

```
bool get_motor_speed_enable(uint32_t& val);
```

参数：

val: 输出参数，电机转速可配置范围。

返回值：

true: 调用成功；

false: 调用失败。

### 3.28 set\_angle\_resolution

设置雷达角分辨率。

```
bool set_angle_resolution(uint16_t val);
```

参数：

val: 输入参数，角分辨，可配置 8、16、32。

返回值：

true: 调用成功；

false: 调用失败。

### 3.29 get\_angle\_resolution

获取雷达角分辨率。

```
bool get_angle_resolution(uint16_t& val);
```

参数：

val: 输出参数，角分辨。

返回值：

true: 调用成功；

false: 调用失败。

### 3.30 get\_angle\_resolution\_enable

获取雷达角分辨率可配置范围。

```
bool get_angle_resolution_enable(uint32_t& val);
```

参数：

val: 输出参数，角分辨可配置范围。

返回值：

true: 调用成功；

false: 调用失败。

### 3.31 set\_device\_ip

配置 Pavo2 Lidar 数据的源 IP 地址。

```
bool set_device_ip(const std::string& ip);
```

参数:

ip: 输入参数。Pavo2 Lidar IP 地址。

返回值:

true: 调用成功;

false: 调用失败。

### 3.32 get\_device\_ip

获取 Pavo2 Lidar 数据的源 IP 地址。

```
bool get_device_ip(std::string& ip);
```

参数:

ip: 输出参数。Pavo2 Lidar 地址。

返回值:

true: 调用成功;

false: 调用失败。

### 3.33 set\_tcp\_port

设置雷达 TCP 服务端口。

```
bool set_tcp_port(uint16_t val);
```

参数:

val: 输入参数。雷达 TCP 服务端口。

返回值:

true: 调用成功;

false: 调用失败。

### 3.34 get\_tcp\_port

获取雷达 TCP 服务端口。

```
bool get_tcp_port(uint16_t& val);
```

参数:

val: 输出参数。雷达 TCP 服务端口。

返回值:

true: 调用成功;

**false:** 调用失败。

### 3.35 set\_device\_mask

设置雷达子网掩码。

```
bool set_device_mask(const std::string& mask);
```

参数:

**mask:** 输入参数。雷达子网掩码。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.36 get\_device\_mask

获取雷达子网掩码。

```
bool get_device_mask(std::string& mask);
```

参数:

**mask:** 输出参数。雷达子网掩码。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.37 set\_device\_gateway

设置雷达网关。

```
bool set_device_gateway(const std::string& gateway);
```

参数:

**gateway:** 输入参数。雷达网关。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.38 get\_device\_gateway

获取雷达网关。

```
bool get_device_gateway(std::string& gateway);
```

参数:

**gateway:** 输出参数。雷达网关。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.39 get\_device\_mac

获取雷达 mac 地址。

```
bool get_device_mac(std::string& mac);
```

参数:

**mac:** 输出参数。雷达 MAC 地址。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.40 set\_os\_restart

设置系统重启模式。

```
bool set_os_restart();
```

参数:

无。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.41 set\_echo\_mode

设置雷达是否开启二回波模式。

```
bool set_echo_mode(bool en);
```

参数:

**en:** 输入参数。

**true:** 开启二回波模式;

**false:** 不开启二回波模式。

返回值:

**true:** 调用成功;

**false:** 调用失败。

### 3.42 get\_echo\_mode

获取雷达是否开启二回波模式。

```
bool get_echo_mode(bool& en);
```

参数：

**en:** 输出参数，雷达是否开启二回波模式。

**true:** 开启二回波模式；

**false:** 不开启二回波模式。

返回值：

**true:** 调用成功；

**false:** 调用失败。

### 3.43 set\_outdoor\_mode

设置雷达是否开启室外增强模式。

```
bool set_outdoor_mode(bool en);
```

参数：

**en:** 输入参数，是否开启室外增强模式。

**true:** 开启室外增强模式；

**false:** 不开启室外增强模式。

返回值：

**true:** 调用成功；

**false:** 调用失败。

### 3.44 get\_outdoor\_mode

获取雷达是否开启室外增强模式。

```
bool get_outdoor_mode(bool& en);
```

参数：

**en:** 输出参数，雷达是否开启室外增强模式。

**true:** 开启室外增强模式；

**false:** 不开启室外增强模式。

返回值：

**true:** 调用成功；

**false:** 调用失败。

### 3.45 set\_calibrate\_horizontal\_angle

设置雷达水平校准角度。

```
bool set_calibrate_horizontal_angle(uint32_t angle);
```

参数：

**angle:** 输入参数。雷达水平校准角度：0-35999。

返回值：

true: 调用成功;  
false: 调用失败。

### 3.46 get\_calibrate\_horizontal\_angle

获取雷达水平校准角度。

```
bool get_calibrate_horizontal_angle(uint32_t& val);
```

参数:

val: 输出参数, 雷达水平校准角度。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.47 get\_max\_angle\_distance

获取雷达最大测距。

```
bool get_max_angle_distance(uint32_t& val);
```

参数:

val: 输出参数, 雷达最大测距。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.48 enable\_tail\_filter

开启点云数据拖尾处理模式。

```
bool enable_tail_filter(uint16_t method);
```

参数:

method: 设置使用的去除拖尾的算法。取值范围[0,1,2,3]。0 表示不启动去除拖尾算法, 1 开启的去除拖尾算法效果最弱, 2 中等, 3 最强。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.49 enable\_front\_filter

开启点云数据前冲滤波处理模式。

```
bool enable_front_filter(uint16_t iLevel);
```

参数:

iLevel: 设置使用的前冲滤波的算法等级。取值范围[0,1,2,3]。0 表示不启动前冲滤波算法,

1 开启的前冲滤波算法效果最弱，2 中等，3 最强。

返回值：

true: 调用成功；

false: 调用失败。

### 3.50 enable\_far\_hr\_filter

开启点云数据远端高反滤波处理模式。

```
bool enable_far_hr_filter(uint16_t iLevel);
```

参数：

iLevel: 设置使用的高反滤波的算法。取值范围[0,1,2]。0 表示不启动高反滤波算法，1 开启的高反滤波算法效果最弱，2 最强。

返回值：

true: 调用成功；

false: 调用失败。

### 3.51 enable\_near\_hr

开启点云近端高反识别。

```
bool enable_near_hr(bool en);
```

参数：

en: 输入参数，是否开启近场高反识别。

返回值：

true: 调用成功；

false: 调用失败。

### 3.52 is\_near\_hr\_enabled

确认近端高反识别是否已开启。

```
bool is_near_hr_enabled(bool& en);
```

参数：

en: 输出参数，近端高反识别是否已开启。

返回值：

true: 调用成功；

false: 调用失败。

### 3.53 is\_near\_hr\_supported

确认近端高反识别是否支持。

```
bool is_near_hr_supported(bool& supported);
```

参数:

supported: 输出参数, 近端高反识别是否支持。

返回值:

true: 调用成功;

false: 调用失败。

### 3.54 set\_warn\_angle\_range

设置点云异常警告角度范围。

```
bool set_warn_angle_range(uint16_t start, uint16_t end);
```

参数:

start: 输入参数, 起始角度。取值范围[4000, 32000]。单位是  $0.01^{\circ}$ 。

end: 输入参数, 终止角度。取值范围[4000, 32000]。单位是  $0.01^{\circ}$ 。

注意: start 值不大于 end。

返回值:

true: 调用成功;

false: 调用失败。

### 3.55 get\_warn\_angle\_range

获取点云异常警告角度范围。

```
bool get_warn_angle_range(uint16_t& start, uint16_t& end);
```

参数:

start: 输出参数。起始角度。

end: 输出参数。终止角度。

返回值:

true: 调用成功;

false: 调用失败。

### 3.56 set\_warn\_intensity\_distance

设置点云异常警告距离强度阈值。

```
bool set_warn_intensity_distance(uint16_t distance, uint16_t intensity);
```

参数:

distance: 输入参数, 距离。

intensity: 输入参数, 强度。

返回值:

true: 调用成功;

false: 调用失败。

### 3.57 get\_warn\_intensity\_distance

获取点云异常警告距离强度阈值。

```
bool get_warn_intensity_distance(uint16_t& distance, uint16_t& intensity);
```

参数:

distance: 输出参数。距离。

intensity: 输出参数。强度。

返回值:

true: 调用成功;

false: 调用失败。

### 3.58 set\_warn\_point\_number

设置点云异常警告点云数量。

```
bool set_warn_point_number(uint16_t number);
```

参数:

number: 输入参数, 点云数量。

返回值:

true: 调用成功;

false: 调用失败。

### 3.59 get\_warn\_point\_number

获取点云异常警告点云数量。

```
bool get_warn_point_number(uint16_t& number);
```

参数:

number: 输出参数。点云数量。

返回值:

true: 调用成功;

false: 调用失败。

### 3.60 enable\_ptp

启用雷达 ptp 功能。

```
bool enable_ptp(bool en);
```

参数:

en: 是否启用雷达 ptp 功能。

true: 开启 ptp 功能;  
false: 不开启 ptp 功能。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.61 is\_ptp\_enabled

查询雷达 ptp 功能是否启用。

```
bool is_ptp_enabled(bool& en);
```

参数:

en: 输出参数。表示雷达 ptp 功能是否启用。

true: ptp 功能已开启;  
false: ptp 功能未开启。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.62 is\_ptp\_supported

查询雷达 ptp 功能是否支持。

```
bool is_ptp_supported(bool& supported);
```

参数:

supported: 输出参数。表示雷达 ptp 功能是否支持。

true: ptp 功能支持;  
false: ptp 功能不支持。

返回值:

true: 调用成功;  
false: 调用失败。

### 3.63 get\_temperature

获取雷达内部温度。

```
bool get_temperature(float& temperature);
```

参数:

temperature: 输出参数。表示雷达内部温度。

返回值:

true: 调用成功;

**false:** 调用失败。

## 3.64 get\_heartbeat

获取雷达连接状态。

```
bool get_heartbeat();
```

参数：

无。

返回值：

**true:** 连接成功；

**false:** 连接失败。