YDLIDAR S2-Pro SDK API

V1.0.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

YDLIDAR DATASET

LIDAR	Model	Baudrate	Sample↩ Rate(K)	Range(m)	Frequency(HZ)	Intenstiy(bi	Single↩ Channel	voltage(↩ V)
S2-Pro	4	115200	4	0.10∼8.0	5∼8	false	false	4.8∼5.2

Chapter 2

FlowChart

```
st=>start: Start
op=>operation: Set Paramamters and Initialize
op1=>operation: TrunOn
tr=>operation: Try Again
op2=>operation: doProcessSimple
op3=>operation: TrunOff
op4=>operation: disconnecting
cond=>condition: success Yes or No?
cond1=>condition: success Yes or No?
cond2=>condition: success Yes or No?
cond3=>condition: LOOP Yes or No?
cond4=>condition: TryAgain Yes or No?
e=>end: End
en=>end: End

st(left)->op->cond
cond(yes)->op1->cond1
cond(no)->op3->op4->e
cond1(yes)->op2->cond3
cond3(yes)->op2
cond3(no, left)->op3->op4->e
cond1(no,right)->tr(bottom)->cond4
cond4(yes)->op3
cond4(no)->op3(right)->op4(right)->e
```

sequenceDiagram

```
sequenceDiagram
note over UserProgram: Set Paramters
note over UserProgram: Initialize SDK
UserProgram->Command: Get LiDAR Information
Command-->UserProgram: Device connected and Devce Information recevied
note over UserProgram: TurnOn
UserProgram->Command: Start LiDAR
Command-->UserProgram: LiDAR Started successfully
UserProgram->LaserScan: Get Laser Scan Data
LaserScan-->UserProgram:  Laser Scan Data Recvied
note over UserProgram: doProcessSimple
loop Laser Scan Data
  LaserScan->UserProgram: doProcessSimple
end
note over UserProgram: TurnOff
UserProgram->Command: TurnOff
note over UserProgram: disconnecting
UserProgram->Command: disconnecting
```

Chapter 3

General FAQs

I am new to the S2-Pro SDK project, where do I start?

You have several options:

- To build S2-Pro your computer, start by reviewing the https://github.com/YDLIDAR/S2-Pro/blob/master/README.md "README.md"
- To install and build YDLIDAR SDK on a robot Project, go to: https://github.com/YDLIDAR/S2-Pro/blob/master/doc/Tutorials.md "S2-Pro SDK quick start".

How do I send a pull request?

Sending a pull request is simple.

1. Fork the S2-Pro Repository into your GitHub.
2. Create a Developer Branch in your Repository.
3. Commit your change in your Developer Branch.
4. Send the pull request from your GitHub Repository Webpage.

More General FAQs to follow.

Chapter 4

General FAQs_cn

请问我怎么样使用**pull request**?

使用pull request非常简单。

1. 将S2-Pro Repository fork到你自己的Github中。
2. 在你的Repository中建立一个开发者 Branch。
3. 在开发者Branch中commit你做的任何的改变
4. 在你的github网页中使用pull request

参考更多的FAQs

Chapter 5

Hardware FAQs

Which types of YD LiDAR are supported by S2-Pro?

please visit https://github.com/YDLIDAR/S2-Pro/blob/master/doc/Dataset.md "this" page.

More Hardware FAQs to follow.

Chapter 6

硬件FAQs:

YDLIDAR雷达需要什么硬件支持?

- 芯片主频大于30MHz.
- 如果芯片主频太低， 数据不能实时解析，数据将会丢失，一些角度范围会丢失，比如:Arduino UNO(16 MHz).
- 推荐最小主频大于30MHz才能实时解析雷达数据，如果是TG30这种采样率20K， 需更高的主频.
- S2-Pro 不支持控制器芯片，如STM32, Arduino.

YDLIDAR雷达可以在什么样的开发板上使用?

- 雷达采样率小于6K的，开发板主频大于30MHz就可以.
- 更改采样率雷达，开发板主频大于100MHz.

S2-Pro支持哪些雷达型号?

S2-Pro 支持S2-Pro雷达，定制版本请联系EAI

Chapter 7

FAQs

Chapter 8

Software FAQs

Can other operating systems besides Ubuntu and windows be used?

We have only tested on Ubuntu and windows which means it's the only operating system we currently officially support. Users are always welcome to try different operating systems and can share their patches with the community if they are successfully able to use them.

More Software FAQs to follow.

Chapter 9

软件FAQ

除了Ubuntu和Windows之外，其他操作系统还能使用吗?

我们只对Ubuntu和Windows进行了测试，这意味着它是我们目前正式支持的操作系统。欢迎开发者尝试不同的操作系统，如果能够成功地使用它们，可以分享补丁与社区。

更多的软件常见问题。

Chapter 10

How to Build and Debug using VSCode

Visual Studio Code (hereafter referred to as VSCode) is Microsoft's first lightweight code editor for Linux. Find below a few configuration files that allow the use of VSCode to compile and debug the S2-Pro project. I will elaborate on it below, hoping to bring some help to the developers.

Compile the S2-Pro project using VSCode

You could first set up the S2-Pro project using the build and release document under **Build in Visual Studio Code**. Only follow the steps until the `Build the S2-Pro Project in VSCode title`

In the pop-up window, select the corresponding The options are as shown below:

Chapter 11

How to Build and Install

Install CMake

The installation procedures in Ubuntu 18.04/16.04/14.04 LTS and Windows 7/10 are shown here as examples. For Ubuntu 18.04/16.04/14.04 32-bit LTS and Mac, you can get it in S2-Pro wiki. S2-Pro requires CMake 2.8.2+ as dependencies.

Ubuntu 18.04/16.04/14.04 LTS

You can install these packages using apt:

```
1 sudo apt install cmake pkg-config
```

Windows 7/10

vcpkg is recommended for building the dependency libraries as follows: For the 32-bit project:

```
1 .\vcpkg install cmake
2 .\vcpkg integrate install
```

For the 64-bit project:

```
1 .\vcpkg install cmake:x64-windows
2 .\vcpkg integrate install
```

Build S2-Pro

Ubuntu 18.04/16.04/14.04 LTS

In the YDLidar S2-Pro SDK directory, run the following commands to compile the project:

```
1 git clone https://github.com/YDLIDAR/S2-Pro.git
2 cd S2-Pro/build
3 cmake ..
4 make
```

Windows 7/10

Then, in the YDLidar S2-Pro SDK directory, run the following commands to create the Visual Studio solution file. Please replace [vcpkgroot] with your vcpkg installation path. Generate the 32-bit project:

```
1 cd build && \
2 cmake .. "-DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]\scripts\buildsystems\vcpkg.cmake"
```

Generate the 64-bit project:

```
1 cd build && \
2 cmake .. -G "Visual Studio 15 2017 Win64"
       "-DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]\scripts\buildsystems\vcpkg.cmake"
```

Compile S2-Pro

You can now compile the YDLidar SDK in Visual Studio. Note:

- For more windows build and Run, Please refer to How to gerenrate Vs Project by CMake
- For VS2017 or higher, Please refer to `CMake projects in visual studio`

Run S2-Pro Sample

Three samples are provided in samples, which demonstrate how to configure YDLidar LiDAR units and receive the laser scan data when directly connecting YDLidar S2-Pro SDK to LiDAR units or by using a YDLidar Adapter board, respectively. The sequence diagram is shown as below:

Ubuntu 18.04/16.04 /14.04 LTS

For Ubuntun 18.04/16.04/14.04 LTS, run the *ydlidar_test* if connect with the Triangle LiDAR unit(s) or TOF LiDAR unit(s):

```
1 ./ydlidar_test
```

Windows 7/10

After compiling the YDLidar SDK as shown in section 4.1.2, you can find `ydlidar_test.exe` in the {S2-Pro} or {S2-Pro} folder, respectively, which can be run directly.

Then you can see SDK initializing the information as below:

Then you can see SDK Scanning the information as below:

Build in Visual Studio Code

Install VSCode

The easiest way to install for Debian/Ubuntu based distributions is to download from https://code.visualstudio.com and install the .deb package (64-bit) either through the graphical software center if it's available or through the command line with:

```
1 sudo dpkg -i <file>.deb
2 sudo apt-get install -f # Install dependencies
```

Start VSCode

Start VSCode with the following command:

```
1 code
```

Open the S2-Pro project in VSCode

Use the keyboard shortcut ∗∗(Ctrl+K Ctrl+O)∗∗ to open the S2-Pro project.

Build the S2-Pro project in VSCode

Use the keyboard shortcut ∗∗(Ctrl+Shift+B)∗∗ to build the S2-Pro project.

Run all unit tests for the S2-Pro project in VSCode

Select the "Tasks->Run Tasks..." menu command and click "run all unit tests for the S2-Pro project" from a popup menu to check the code style for the S2-Pro project.

Run a code style check task for the S2-Pro project in VSCode

Select the "Tasks->Run Tasks..." menu command and click "code style check for the S2-Pro project" from a popup menu to check the code style for the S2-Pro project.

Clean the S2-Pro project in VSCode

Select the "Tasks->Run Tasks..." menu command and click "clean the S2-Pro project" from a popup menu to clean the S2-Pro project.

Chapter 12

How to create a pull request

You can follow the standard `github approach` to contribute code to S2-Pro. Here is a sample setup:

- Fork a new repo with your GitHub username.
- Set up your GitHub personal email and user name

```
1 git config user.name "XXX"
2 git config user.email "XXX@[XXX.com]"
```

- Clone your fork (Please replace "USERNAME" with your GitHub user name.)

```
1 (Use SSH) git clone git@github.com:USERNAME/S2-Pro.git
2 (Use HTTPS) git clone https://github.com/USERNAME/S2-Pro.git
```

- Add S2-Pro repository as upstream

```
1 (Use SSH) git remote add upstream git@github.com:YDLIDAR/S2-Pro.git
2 (Use HTTPS) git remote add upstream https://github.com/YDLIDAR/S2-Pro.git
```

- Confirm that the upstream branch has been added

```
1 git remote -v
```

- Create a new branch, make changes and commit

```
1 git checkout -b  "my_dev"
```

- Sync up with the YDLIDAR/S2-Pro repo

```
1 git pull --rebase  upstream master
```

- Push local developments to your own forked repository

```
1 git push -f -u origin "my_dev"
```

- Generate a new pull request between "YDLIDAR/S2-Pro:master" and "forked repo:my_dev"
- Collaborators will review and merge the commit (this may take some time, please be patient)

Thanks a lot for your contributions!

Chapter 13

How to create a udev rules

Introduction

The serial port is used under Linux. The serial port number will change with the insertion order of multiple serial ports. This problem can be solved by setting the serial port alias.

Create The New UDEV Rules

Create a new `ydlidar_ports.rules` file and write the corresponding serial port rules to the file.

Create new udev file

```
1 sudo gedit /etc/udev/rules.d/ydlidar_ports.rules
```

or

```
1 sudo vim /etc/udev/rules.d/ydlidar_ports.rules
```

Query serial port number through udevadm

```
1 udevadm info -a -n /dev/ttyUSB0 | grep KERNELS
```

result as follows:

```
1 udevadm info -a -n /dev/ttyUSB1 | grep KERNELS
```

result as follows:

Create UDEV Permission Rule For tty Devices

Write the first KERNELS queried above into the new `ydlidar_ports.rules` file. Add these two following rules in it.

```
1 {ydlidar_ports.rules}
2 SUBSYSTEM=="tty", KERNELS=="1-1:1.0", SYMLINK+="ydlidar", MODE="0666", GROUP:="dialout"
3 SUBSYSTEM=="tty", KERNELS=="1-2:1.0", SYMLINK+="ydlidar1", MODE="0666", GROUP:="dialout"
```

Restart The UDEV Service

Save the file and close it. Then as root, tell systemd-udevd to reload the rules files (this also reloads other databases such as the kernel module index), by running.

```
1 sudo udevadm control --reload
```

and

```
1 sudo service udev reload
2 sudo service udev restart
```

Note: If it doesn't work, plug and unplug the USB or restart the computer

You can query the corresponding results with the following command

```
1 ls -l /dev/ydlidar*
2
3 lrwxrwxrwx 1 root dialout 7 Feb 17 13:27 /dev/ydlidar -> ttyUSB0
4 lrwxrwxrwx 1 root dialout 7 Feb 17 13:27 /dev/ydlidar1 -> ttyUSB1
```

Chapter 14

Introduction

The Visual Studio version recommended by S2-Pro to use is Visual Studio 2017. This document decribes steps to run S2-Pro on Visual Studio 2017.

The S2-Pro version used in this document is the lastest release version which is 1.0.0. And this document focuses on How to Install Software, and conforms to the steps and rules provided by S2-Pro.

Download S2-Pro

please refer to https://github.com/YDLIDAR/S2-Pro/blob/master/doc/quickstart/s2_pro_software_installation_↩guide.md "S2-Pro Software Installation"， download S2-Pro version 1.0.0 source code onto the computer.

Install CMake

Please follow the official guide to install the cmake.

Build and Run S2-Pro

Please refer to https://github.com/YDLIDAR/S2-Pro/blob/master/doc/howto/how_to_build_and_release.md "How to build and release".

Chapter 15

Github访问慢解决方案

浏览器打开如下网站

http://github.global.ssl.fastly.net.ipaddress.com/

找到对应IP地址，例如：151.101.xx.xx

浏览器打开另外一个网站

http://github.com.ipaddress.com/

找到对应IP地址。例如：192.30.xx.xx

编辑hosts文件

```
1 sudo vim /etc/hosts
```

在文件中加入如下两行

```
1 192.30.xx.xx github.com
2 151.101.xx.xx github.global.ssl.fastly.net
```

如果使用mac，还需更新DNS缓存

```
1 sudo dscacheutil -flushcache
```

Chapter 16

Howto Guides

Build

- How to build and install
- How to build and debug using VSCode
- How to create a csharp project

Contribution

- How to create a pull request

Others

- How to create a udev rules

Chinese versions

- How to install ubuntu
- How to solve slow pull from cn

Chapter 17

Quick Start Guides

S2-Pro 1.0.0

- S2-Pro 1.0.0 quick start
- S2-Pro 1.0.0 quick start cn
- S2-Pro 1.0.0 hardware system installation guide
- S2-Pro 1.0.0 quick start developer

Others

- S2-Pro software installation guide

Chapter 18

Software Overview of S2-Pro

S2-Pro has been initiated to provide an open, comprehensive, and reliable software platform for its partners in the robot, Large screen interaction and mapping industries.

S2-Pro Software Installation

This section includes:

- Download the S2-Pro Release Package
- Run S2-Pro

Before getting started, please make sure you have installed Linux or Windows.

∗New - Git∗

git Installation

ubuntu 14.04 / 16.04 / 18.04

```
1 sudo apt-get install -y git
```

Windows

Installation

Download S2-Pro Source

1. Download S2-Pro source code from the github source and check out the correct branch:
   “‘ git clone git@github.com:YDLIDAR/S2-Pro.git cd S2-Pro git checkout [release_branch_name] “‘

Run S2-Pro

Please refer to https://github.com/YDLIDAR/S2-Pro/blob/master/doc/howto/how_to_build_and_release.md "How to build and release"

Chapter 19

YDLIDAR S2-Pro SDK Documents

Quick Start Guide

README - A hardware and software guide to setting up S2-Pro, segregated by versions

API

YDLIDAR S2-Pro SDK API for Developers - All you need to know about YDLiDAR S2-Pro SDK API

Howto Guides

README - Brief technical solutions to common problems that developers face during the installation and use of the S2-Pro

FAQs

README - Commonly asked questions about S2-Pro's setup

Chapter 20

S2-Pro SDK API for Developers

set lidar properties

This document provides an extensive technical deep dive into how to create, manipulate and use YDLIDAR SDK's API.

Table of Contents

Samples

The first part of demonstrating YDLIDAR S2-Pro SDK API is to understand the ydlidar_test/ydlidar_test example. Following are one optinal concepts: `ydlidar::init(int argc, char *argv[])` (basic unit) of the example.

Create A System State

In the YDLIDAR S2-Pro SDK, the `ydlidar::init(int argc, char *argv[])` is optinal unit, If you need to accept `Ctrl + C` or other system abnormal signals. you can use it to create a system state, and check whether the system is normal by `ydlidar::ok()`. The system signal creation interface is as follows:

```
ydlidar::init(int argc, char *argv[]);
```

- when `ydlidar::init(int argc, char *argv[])` has called, the system is in an initialized state, able to accept `Ctrl + C` and `ydlidar::shutdown()` signals.

Code Example

S2-Pro LiDAR (../samples/ydlidar_test.cpp)

```
#include "CYdLidar.h"
#include <iostream>
#include <string>
#include <algorithm>
#include <cctype>
using namespace std;
using namespace ydlidar;
#if defined(_MSC_VER)
#pragma comment(lib, "ydlidar_sdk.lib")
#endif

int main(int argc, char *argv[]) {
  // init system signal
  ydlidar::init(argc, argv);

  CYdLidar laser;
  std::string port = "/dev/ydlidar
  laser.setlidaropt(LidarPropSerialPort, port.c_str(), port.size());
  std::string ignore_array;
  ignore_array.clear();
  laser.setlidaropt(LidarPropIgnoreArray, ignore_array.c_str(),
                    ignore_array.size());

  int optval = 115200;
  laser.setlidaropt(LidarPropSerialBaudrate, &optval, sizeof(int));
  optval = 4;
  laser.setlidaropt(LidarPropAbnormalCheckCount, &optval, sizeof(int));

  bool b_optvalue = false;
  laser.setlidaropt(LidarPropFixedResolution, &b_optvalue, sizeof(bool));
  laser.setlidaropt(LidarPropReversion, &b_optvalue, sizeof(bool));
  laser.setlidaropt(LidarPropInverted, &b_optvalue, sizeof(bool));
  b_optvalue = true;
  laser.setlidaropt(LidarPropAutoReconnect, &b_optvalue, sizeof(bool));

  float f_optvalue = 180.0f;
  laser.setlidaropt(LidarPropMaxAngle, &f_optvalue, sizeof(float));
  f_optvalue = -180.0f;
  laser.setlidaropt(LidarPropMinAngle, &f_optvalue, sizeof(float));
  f_optvalue = 16.f;
  laser.setlidaropt(LidarPropMaxRange, &f_optvalue, sizeof(float));
  f_optvalue = 0.1f;
  laser.setlidaropt(LidarPropMinRange, &f_optvalue, sizeof(float));
  f_optvalue = 10.f;
  laser.setlidaropt(LidarPropScanFrequency, &f_optvalue, sizeof(float));

  // initialize SDK and LiDAR
  bool ret = laser.initialize();
  if (ret) {//success
    //Start the device scanning routine which runs on a separate thread and enable motor.
    ret = laser.turnOn();
  } else {
    fprintf(stderr, "%s\n", laser.DescribeError());
    fflush(stderr);
  }

  LaserScan scan;
  // Turn On success and loop
  while (ret && ydlidar::ok()) {
    bool hardError;
    scan.points.clear();

    if (laser.doProcessSimple(scan, hardError)) {
      fprintf(stdout, "Scan received[%llu]: %u ranges is [%f]Hz\n",
              scan.stamp,
              (unsigned int)scan.points.size(), 1.0 / scan.config.scan_time);
      fflush(stdout);
    } else {
      printf("[YDLIDAR ERROR]: %s\n",ydlidar::protocol::DescribeError(laser.getDriverError()));
      fflush(stdout);
    }
  }
  // Stop the device scanning thread and disable motor.
  laser.turnOff();
  // Uninitialize the SDK and Disconnect the LiDAR.
  laser.disconnecting();
  return 0;
}
```

CMake BUILD file(../samples/CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.8)
PROJECT(ydlidar_test)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
add_definitions(-std=c++11) # Use C++11

#Include directories
INCLUDE_DIRECTORIES(
      ${CMAKE_SOURCE_DIR}
      ${CMAKE_SOURCE_DIR}/../
       ${CMAKE_CURRENT_BINARY_DIR}
)

SET(EXECUTABLE_OUTPUT_PATH ${CMAKE_BINARY_DIR})
ADD_EXECUTABLE(${PROJECT_NAME}
                main.cpp)

# Add the required libraries for linking:
TARGET_LINK_LIBRARIES(${PROJECT_NAME} ydlidar_driver)
```

Build and Run

- Build: cd build & cmake ../ & make
- Run ydlidar_test in terminals:
  - ./ydlidar_test
- Examine the results: you should see message printing out in terminals.

Development Flow

FlowChart

Sequence

API Directory

CYDLidar API

For additional information and examples, refer to CYDLidar

API List

C++ API

```
{C++}
 /**
   *   * @param optname        option name
   * \xrefitem todo 1.@note set string property example
   * @code
   * CYdLidar laser;
   * std::string lidar_port = "/dev/ydlidar";
   * laser.setlidaropt(LidarPropSerialPort,lidar_port.c_str(), lidar_port.size());
   *
```

- **Todo** int properties

Note

set int property example

```
1 * CYdLidar laser;
2 * int lidar_baudrate = 230400;
3 * laser.setlidaropt(LidarPropSerialPort,&lidar_baudrate, sizeof(int));
4 *
```

- **Todo** bool properties

Note

set bool property example

```
1 * CYdLidar laser;
2 * bool lidar_fixedresolution = true;
3 * laser.setlidaropt(LidarPropSerialPort,&lidar_fixedresolution, sizeof(bool));
4 *
```

- **Todo** float properties

Note

set float property example, Must be float type, not double type.

```
1 * CYdLidar laser;
2 * float lidar_maxrange = 16.0f;
3 * laser.setlidaropt(LidarPropSerialPort,&lidar_maxrange, sizeof(float));
4 *
```

Parameters

optval	option value

-
- - std::string(or char∗)
- - int
- - bool
- - float

Parameters

optlen	option length

-
- - data type size

Returns

- true if the Property is set successfully, otherwise false.

See also

- LidarProperty ∗/ bool setlidaropt(int optname, const void ∗optval, int optlen);

/∗∗

- get lidar property

Parameters

optname	option name

-

- **Todo** string properties

  Note

  get string property example

- ```
  1 * CYdLidar laser;
  2 * char lidar_port[30];
  3 * laser.getlidaropt(LidarPropSerialPort,lidar_port, sizeof(lidar_port));
  4 *
  ```

- **Todo** int properties

  Note

  get int property example

- ```
  1 * CYdLidar laser;
  2 * int lidar_baudrate;
  3 * laser.getlidaropt(LidarPropSerialPort,&lidar_baudrate, sizeof(int));
  4 *
  ```

- **Todo** bool properties

  Note

  get bool property example

- ```
  1 * CYdLidar laser;
  2 * bool lidar_fixedresolution;
  3 * laser.getlidaropt(LidarPropSerialPort,&lidar_fixedresolution, sizeof(bool));
  4 *
  ```

- **Todo** float properties

  Note

  set float property example

- ```
  1 * CYdLidar laser;
  2 * float lidar_maxrange;
  3 * laser.getlidaropt(LidarPropSerialPort,&lidar_maxrange, sizeof(float));
  4 *
  ```

Parameters

optval	option value

-

- - std::string(or char*)
- - int
- - bool
- - float

Parameters

optlen	option length

-

- - data type size

Returns

- true if the Property is get successfully, otherwise false.

See also

- LidarProperty */ bool getlidaropt(int optname, void *optval, int optlen);

/**

- Initialize the SDK and LiDAR.

Returns

- true if successfully initialized, otherwise false. */ bool initialize();

/**

Return LiDAR's version information in a numeric form.

Parameters

version	Pointer to a version structure for returning the version information. */ void GetLidarVersion(LidarVersion &version);

- /**
  - Start the device scanning routine which runs on a separate thread and enable motor.

    Returns
  - true if successfully started, otherwise false. */ bool turnOn(); //!< See base class docs

    /**
  - Get the LiDAR Scan Data. turnOn is successful before doProcessSimple scan data.

Parameters

out	*outscan*	LiDAR Scan Data

  -

Parameters

out	*hardwareError*	hardware error status

  -

    Returns
  - true if successfully started, otherwise false. */ bool doProcessSimple(LaserScan &scan_msg, bool &hardwareError);

    /**
  - Stop the device scanning thread and disable motor.

    Returns
  - true if successfully Stoped, otherwise false. */ bool turnOff(); //!< See base class docs

    /**
  - Uninitialize the SDK and Disconnect the LiDAR. */ void disconnecting(); //!< Closes the comms with the laser. Shouldn't have to be directly needed by the user

/**

- Get the last error information of a (lidar or serial)

  Returns

- a human-readable description of the given error information

- or the last error information of a (lidar or serial) */ lidar_error_t getDriverError() const;

YDlidarDriver API

For additional information and examples, refer to YDlidarDriver

API List

```
result_t connect(const char *port_path, uint32_t baudrate);

void disconnect();

static std::string getSDKVersion();


lidar_error_t getDriverError();
result_t getHealth(device_health &health, uint32_t timeout = DEFAULT_TIMEOUT);

result_t getDeviceInfo(device_info &info, uint32_t timeout = DEFAULT_TIMEOUT);

result_t getScanFrequency(scan_frequency_t &frequency,
                          uint32_t timeout = DEFAULT_TIMEOUT);

result_t setScanFrequencyAdd(scan_frequency_t &frequency,
                             uint32_t timeout = DEFAULT_TIMEOUT);

result_t setScanFrequencyDis(scan_frequency_t &frequency,
                             uint32_t timeout = DEFAULT_TIMEOUT);

result_t setScanFrequencyAddMic(scan_frequency_t &frequency,
                                uint32_t timeout = DEFAULT_TIMEOUT);

result_t setScanFrequencyDisMic(scan_frequency_t &frequency,
                                uint32_t timeout = DEFAULT_TIMEOUT);

result_t getZeroOffsetAngle(offset_angle_t &angle,
                            uint32_t timeout =  DEFAULT_TIMEOUT);

static std::map<std::string, std::string> lidarPortList();

bool isConnected() const;

bool isScanning() const;

uint32_t getPointIntervalTime() const;

uint32_t getPackageTransferTime() const;

void setAutoReconnect(const bool &enable);

void setSingleChannel(bool enable);

result_t startScan(uint32_t timeout = DEFAULT_TIMEOUT) ;

result_t stopScan(uint32_t timeout = DEFAULT_TIMEOUT);


result_t stop();


result_t grabScanData(LaserFan *fan, uint32_t timeout = DEFAULT_TIMEOUT) ;

result_t startMotor();

result_t stopMotor();

void flush();
```

Parameter Table

The Table that the user uses to perform parameter related operations:

- Set the parameter related API by table.

For additional information and examples, refer to `Parameter`

Table List - Models

LIDAR	Model	Baudrate	Sample↩ Rate(K)	Range(m)	Frequency(HZ)	Intenstiy(bi	Single↩ Channel	voltage(↩ V)
S2-Pro	4	115200	3	0.12∼8.0	5∼8	false	false	4.8∼5.2

Chapter 21

YDlidarDriver

YDlidarDriver API

Library	YDlidarDriver
File	ydlidar_driver.h
Author	Tony [code at ydlidar com]
Source	https://github.com/ydlidar/S2-Pro
Version	1.0.0

Chapter 22

YDLIDAR S2-Pro SDK PACKAGE V1.0.0

Table of Contents

Introduction

YDLidar S2-Pro SDK is the software development kit designed for YDLIDAR S2-Pro products. It is developed based on C++ following YDLidar S2-Pro SDK Communication Protocol, and provides easy-to-use C++,style API. With YDLidar S2-Pro SDK, users can quickly connect to YDLidar S2-Pro products and receive Laser scan data.

YDLidar S2-Pro SDK consists of YDLidar S2-Pro SDK communication protocol, YDLidar S2-Pro SDK core, YDLidar S2-Pro SDK API, Linux/windows samples.

Prerequisites

- Linux
- Windows 7/10, Visual Studio 2015/2017
- C++11 compiler

Installation

- Fork and then Clone S2-Pro's GitHub code
- Build and Install - This step is required

Documents

- LiDAR Dataset: All you need to know about LiDAR Models.
- SDK FlowChart: Development flowchart.
- YDLIDAR SDK API for Developers: All you need to know about S2-Pro API
- HowTo: Brief technical solutions to common problems that developers face during the installation and use of the S2-Pro
- FAQs

Support

You can get support from YDLidar with the following methods:

- Send email to support@ydlidar.com with a clear description of your problem and your setup
- Github Issues

Contact EAI

If you have any extra questions, please feel free to contact us

Chapter 23

Todo List

Member CYdLidar::getlidaropt (int optname, void *optval, int optlen)

string properties

- LidarPropSerialPort
- LidarPropIgnoreArray

int properties

- LidarPropSerialBaudrate
- LidarPropLidarType
- LidarPropDeviceType
- LidarPropSampleRate

bool properties

- LidarPropFixedResolution
- LidarPropReversion
- LidarPropInverted
- LidarPropAutoReconnect
- LidarPropSingleChannel
- LidarPropIntenstiy

float properties

- LidarPropMaxRange
- LidarPropMinRange
- LidarPropMaxAngle
- LidarPropMinAngle
- LidarPropScanFrequency

Member CYdLidar::setlidaropt (int optname, const void *optval, int optlen)

string properties

- LidarPropSerialPort
- LidarPropIgnoreArray

int properties

- LidarPropSerialBaudrate
- LidarPropLidarType

- LidarPropDeviceType
- LidarPropSampleRate

bool properties

- LidarPropFixedResolution
- LidarPropReversion
- LidarPropInverted
- LidarPropAutoReconnect
- LidarPropSingleChannel
- LidarPropIntenstiy

float properties

- LidarPropMaxRange
- LidarPropMinRange
- LidarPropMaxAngle
- LidarPropMinAngle
- LidarPropScanFrequency

Page S2-Pro SDK API for Developers

string properties

- - LidarPropSerialPort
- - LidarPropIgnoreArray
-

int properties

- - LidarPropSerialBaudrate
- - LidarPropLidarType
- - LidarPropDeviceType
- - LidarPropSampleRate
-

bool properties

- - LidarPropFixedResolution
- - LidarPropReversion
- - LidarPropInverted
- - LidarPropAutoReconnect
- - LidarPropSingleChannel
- - LidarPropIntenstiy
-

float properties

- - LidarPropMaxRange
- - LidarPropMinRange
- - LidarPropMaxAngle
- - LidarPropMinAngle
- - LidarPropScanFrequency
-

string properties

- - LidarPropSerialPort
- - LidarPropIgnoreArray
-

int properties

- - LidarPropSerialBaudrate
- - LidarPropLidarType
- - LidarPropDeviceType
- - LidarPropSampleRate
-

bool properties

- - LidarPropFixedResolution
- - LidarPropReversion
- - LidarPropInverted
- - LidarPropAutoReconnect
- - LidarPropSingleChannel
- - LidarPropIntenstiy
-

float properties

- - LidarPropMaxRange
- - LidarPropMinRange
- - LidarPropMaxAngle
- - LidarPropMinAngle
- - LidarPropScanFrequency
-

Chapter 24

Namespace Index

24.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 25

Class Index

25.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 26

File Index

26.1 File List

Here is a list of all files with brief descriptions:

Chapter 27

Namespace Documentation

27.1 angles Namespace Reference

Functions

- static double from_degrees (double degrees)

  Convert degrees to radians.
- static double to_degrees (double radians)

  Convert radians to degrees.
- static double normalize_angle_positive (double angle)

  normalize_angle_positive
- static double normalize_angle (double angle)

  normalize
- static double shortest_angular_distance (double from, double to)

  shortest_angular_distance
- static double two_pi_complement (double angle)

  returns the angle in [-2∗M_PI, 2∗M_PI] going the other way along the unit circle.
- static bool find_min_max_delta (double from, double left_limit, double right_limit, double &result_min_delta, double &result_max_delta)

  This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".
- static bool shortest_angular_distance_with_limits (double from, double to, double left_limit, double right_limit, double &shortest_angle)

  Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by left_limit and right_limit. The valid interval of angular positions is [left_limit,right_limit]. E.g., [-0.25,0.25] is a 0.5 radians wide interval that contains 0. But [0.25,-0.25] is a 2∗M_PI-0.5 wide interval that contains M_PI (but not 0). The value of shortest_angle is the angular difference between "from" and "to" that lies within the defined valid interval. E.g. shortest_angular_distance_with_limits(-0.5,0.5,0.25,-0.25,ss) evaluates ss to 2∗M_PI-1.0 and returns true while shortest_angular_distance_with_limits(-0.5,0.5,-0.25,0.25,ss) returns false since -0.5 and 0.5 do not lie in the interval [-0.25,0.25].

27.1.1 Function Documentation

27.1.1.1 static bool angles::find_min_max_delta (double *from,* double *left_limit,* double *right_limit,* double & *result_min_delta,* double & *result_max_delta*) [static]

This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".

Returns

returns false if "from" angle does not lie in the interval [left_limit,right_limit]

Parameters

from	- "from" angle - must lie in [-M_PI, M_PI)
left_limit	- left limit of valid interval for angular position - must lie in [-M_PI, M_PI], left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
right_limit	- right limit of valid interval for angular position - must lie in [-M_PI, M_PI], left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
result_min_delta	- minimum (delta) angle (in radians) that can be moved from "from" position before hitting the joint stop
result_max_delta	- maximum (delta) angle (in radians) that can be movedd from "from" position before hitting the joint stop

Definition at line 140 of file angles.h.

27.1.1.2 static double angles::from_degrees (double *degrees*) [inline], [static]

Convert degrees to radians.

Definition at line 48 of file angles.h.

27.1.1.3 static double angles::normalize_angle (double *angle*) [inline], [static]

normalize

Normalizes the angle to be -M_PI circle to +M_PI circle It takes and returns radians.

Definition at line 81 of file angles.h.

27.1.1.4 static double angles::normalize_angle_positive (double *angle*) [inline], [static]

normalize_angle_positive

Normalizes the angle to be 0 to 2∗M_PI It takes and returns radians.

Definition at line 68 of file angles.h.

27.1.1.5 static double angles::shortest_angular_distance (double *from,* double *to*) [inline],[static]

shortest_angular_distance

Given 2 angles, this returns the shortest angular difference. The inputs and ouputs are of course radians.

The result would always be -pi <= result <= pi. Adding the result to "from" will always get you an equivelent angle to "to".

Definition at line 102 of file angles.h.

27.1.1.6 static bool angles::shortest_angular_distance_with_limits (double *from,* double *to,* double *left_limit,* double *right_limit,* double & *shortest_angle*) [inline],[static]

Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by left_limit and right_limit. The valid interval of angular positions is [left_limit,right_limit]. E.g., [-0.25,0.25] is a 0.5 radians wide interval that contains 0. But [0.25,-0.25] is a 2∗M_PI-0.5 wide interval that contains M_PI (but not 0). The value of shortest_angle is the angular difference between "from" and "to" that lies within the defined valid interval. E.g. shortest_angular_distance_with_limits(-0.5,0.5,0.25,-0.25,ss) evaluates ss to 2∗M_PI-1.0 and returns true while shortest_angular_distance_with_limits(-0.5,0.5,-0.25,0.25,ss) returns false since -0.5 and 0.5 do not lie in the interval [-0.25,0.25].

Returns

true if "from" and "to" positions are within the limit interval, false otherwise

Parameters

from	- "from" angle
to	- "to" angle
left_limit	- left limit of valid interval for angular position, left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
right_limit	- right limit of valid interval for angular position, left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
shortest_angle	- result of the shortest angle calculation

Definition at line 215 of file angles.h.

27.1.1.7 static double angles::to_degrees (double *radians*) [inline],[static]

Convert radians to degrees.

Definition at line 56 of file angles.h.

27.1.1.8 static double angles::two_pi_complement (double *angle*) [inline],[static]

returns the angle in [-2∗M_PI, 2∗M_PI] going the other way along the unit circle.

Parameters

angle	The angle to which you want to turn in the range [-2∗M_PI, 2∗M_PI] E.g. two_pi_complement(-M_PI/4) returns 7_M_PI/4 two_pi_complement(M_PI/4) returns -7∗M_PI/4

Definition at line 116 of file angles.h.

27.2 impl Namespace Reference

Functions

- uint32_t getHDTimer ()
- uint64_t getCurrentTime ()

27.2.1 Function Documentation

27.2.1.1 uint64_t impl::getCurrentTime ( )

Definition at line 11 of file unix_timer.cpp.

27.2.1.2 uint32_t impl::getHDTimer ( )

Definition at line 5 of file unix_timer.cpp.

27.3 response_health_error Namespace Reference

Enumerations

- enum bits : uint8_t {
  SensorError = 1 << 0, EncodeError = 1 << 1, PWRError = 1 << 2, PDError = 1 << 3,
  LDError = 1 << 4, DataError = 1 << 5, CSError = 1 << 6 }

27.3.1 Enumeration Type Documentation

27.3.1.1 enum response_health_error::bits : uint8_t

Enumerator

SensorError

EncodeError

PWRError

PDError

LDError

DataError

CSError

Definition at line 114 of file ydlidar_def.h.

27.4 response_scan_packet_sync Namespace Reference

Enumerations

- enum bits : uint8_t {
  sync = 1 << 0, reserved1 = 1 << 1, reserved2 = 1 << 2, reserved3 = 1 << 3,
  reserved4 = 1 << 4, reserved5 = 1 << 5, reserved6 = 1 << 6, reserved7 = 1 << 7 }

27.4.1 Enumeration Type Documentation

27.4.1.1 enum response_scan_packet_sync::bits : uint8_t

Enumerator

sync

reserved1

reserved2

reserved3

reserved4

reserved5

reserved6

reserved7

Definition at line 126 of file ydlidar_def.h.

27.5 serial Namespace Reference

Classes

- class MillisecondTimer
- struct PortInfo
- class Serial
- struct Timeout

Enumerations

- enum bytesize_t { fivebits = 5, sixbits = 6, sevenbits = 7, eightbits = 8 }
- enum parity_t {
  parity_none = 0, parity_odd = 1, parity_even = 2, parity_mark = 3,
  parity_space = 4 }
- enum stopbits_t { stopbits_one = 1, stopbits_two = 2, stopbits_one_point_five }
- enum flowcontrol_t { flowcontrol_none = 0, flowcontrol_software, flowcontrol_hardware }

Functions

- std::vector< PortInfo > list_ports ()

27.5.1 Enumeration Type Documentation

27.5.1.1 enum serial::bytesize_t

Enumeration defines the possible bytesizes for the serial port.

Enumerator

fivebits
sixbits
sevenbits
eightbits

Definition at line 16 of file serial.h.

27.5.1.2 enum serial::flowcontrol_t

Enumeration defines the possible flowcontrol types for the serial port.

Enumerator

flowcontrol_none
flowcontrol_software
flowcontrol_hardware

Definition at line 46 of file serial.h.

27.5.1.3 enum serial::parity_t

Enumeration defines the possible parity types for the serial port.

Enumerator

parity_none
parity_odd
parity_even
parity_mark
parity_space

Definition at line 26 of file serial.h.

27.5.1.4 enum serial::stopbits_t

Enumeration defines the possible stopbit types for the serial port.

Enumerator

stopbits_one
stopbits_two
stopbits_one_point_five

Definition at line 37 of file serial.h.

27.5.2 Function Documentation

27.5.2.1 std::vector<PortInfo> serial::list_ports ( )

27.6 ydlidar Namespace Reference

Namespaces

- protocol

Classes

- class YDlidarDriver

Functions

- void init (int argc, char *argv[])
- bool ok ()
- void shutdownNow ()
- std::vector< float > split (const std::string &s, char delim)

  split string to vector by delim format
- std::string format (const char *fmt,...)

27.6.1 Function Documentation

27.6.1.1 std::string ydlidar::format (const char * *fmt*, ...)

Definition at line 32 of file ydlidar_driver.cpp.

27.6.1.2 void ydlidar::init (int *argc*, char * *argv[]*) [inline]

Definition at line 198 of file v8stdint.h.

27.6.1.3 bool ydlidar::ok ( ) [inline]

Definition at line 215 of file v8stdint.h.

27.6.1.4 void ydlidar::shutdownNow ( ) [inline]

Definition at line 218 of file v8stdint.h.

27.6.1.5 std::vector<float> ydlidar::split (const std::string & *s*, char *delim*) [inline]

split string to vector by delim format

Parameters

s	string
delim	split format

Returns

split vector

Definition at line 228 of file v8stdint.h.

27.7 ydlidar::protocol Namespace Reference

Functions

- const char * DescribeError (const lidar_error_t &error)
- void reset_ct_packet_t (ct_packet_t &ct)
- lidar_error_t convert_ct_packet_to_error (const ct_packet_t &ct)
- result_t check_ct_packet_t (const ct_packet_t &ct)
- void write_command (Serial *serial, uint8_t cmd)
- result_t wait_for_data (Serial *serial, size_t data_count, uint32_t timeout=1000)
- result_t read_command (Serial *serial, uint8_t *buffer, size_t size, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_header_t (Serial *serial, lidar_ans_header_t &header, lidar_error_t &error, uint32_t timeout=1000)
- result_t check_ans_header_t (const lidar_ans_header_t &header, lidar_error_t &error)
- result_t read_response_health_t (Serial *serial, device_health &health, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_device_info_t (Serial *serial, device_info &info, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_sample_rate_t (Serial *serial, sampling_rate_t &rate, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_scan_frequency_t (Serial *serial, scan_frequency_t &frequency, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_offset_angle_t (Serial *serial, offset_angle_t &angle, lidar_error_t &error, uint32_t timeout=1000)
- result_t parse_payload (const scan_packet_t &scan, LaserFan &data)
- result_t parse_intensity_payload (const scan_intensity_packet_t &scan, LaserFan &data)
- result_t check_package_header_t (const node_package_header_t &header, lidar_error_t &error)
- result_t parse_ct_packet_t (const node_package_header_t &header, unsigned short error_count, ct_packet_t &ct)
- uint8_t crc8_t (uint8_t *ptr, uint16_t len, uint8_t default_crc=0x00, uint8_t poly=0x8c, uint8_t inverted=1)
- uint16_t checksum_response_scan_packet_t (const scan_packet_t &scan)
- uint16_t checksum_response_scan_intensity_packet_t (const scan_intensity_packet_t &scan)
- result_t read_response_scan_header_t (Serial *serial, node_package_header_t &header, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_scan_t (Serial *serial, scan_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t read_response_scan_intensity_t (Serial *serial, scan_intensity_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)

27.7.1 Function Documentation

27.7.1.1 result_t ydlidar::protocol::check_ans_header_t (const lidar_ans_header_t & *header,* lidar_error_t & *error*)

Definition at line 347 of file ydlidar_protocol.cpp.

27.7.1.2 result_t ydlidar::protocol::check_ct_packet_t (const ct_packet_t & *ct*)

Definition at line 221 of file ydlidar_protocol.cpp.

27.7.1.3 result_t ydlidar::protocol::check_package_header_t (const node_package_header_t & *header,* lidar_error_t & *error*)

Definition at line 581 of file ydlidar_protocol.cpp.

27.7.1.4 uint16_t ydlidar::protocol::checksum_response_scan_intensity_packet_t (const scan_intensity_packet_t & *scan*)

Definition at line 663 of file ydlidar_protocol.cpp.

27.7.1.5 uint16_t ydlidar::protocol::checksum_response_scan_packet_t (const scan_packet_t & *scan*)

Definition at line 639 of file ydlidar_protocol.cpp.

27.7.1.6 lidar_error_t ydlidar::protocol::convert_ct_packet_to_error (const ct_packet_t & *ct*)

Definition at line 171 of file ydlidar_protocol.cpp.

27.7.1.7 uint8_t ydlidar::protocol::crc8_t (uint8_t * *ptr,* uint16_t *len,* uint8_t *default_crc* = `0x00`, uint8_t *poly* = `0x8c`, uint8_t *inverted* = `1`)

Definition at line 611 of file ydlidar_protocol.cpp.

27.7.1.8 const char * ydlidar::protocol::DescribeError (const lidar_error_t & *error*)

Definition at line 34 of file ydlidar_protocol.cpp.

27.7.1.9 result_t ydlidar::protocol::parse_ct_packet_t (const node_package_header_t & *header,* unsigned short *error_count,* ct_packet_t & *ct*)

Definition at line 688 of file ydlidar_protocol.cpp.

27.7.1.10 **result_t ydlidar::protocol::parse_intensity_payload (const scan_intensity_packet_t & *scan,* LaserFan & *data*)**

Definition at line 527 of file ydlidar_protocol.cpp.

27.7.1.11 **result_t ydlidar::protocol::parse_payload (const scan_packet_t & *scan,* LaserFan & *data*)**

Definition at line 475 of file ydlidar_protocol.cpp.

27.7.1.12 **result_t ydlidar::protocol::read_command (Serial * *serial,* uint8_t * *buffer,* size_t *size,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 249 of file ydlidar_protocol.cpp.

27.7.1.13 **result_t ydlidar::protocol::read_response_device_info_t (Serial * *serial,* device_info & *info,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 389 of file ydlidar_protocol.cpp.

27.7.1.14 **result_t ydlidar::protocol::read_response_header_t (Serial * *serial,* lidar_ans_header_t & *header,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 270 of file ydlidar_protocol.cpp.

27.7.1.15 **result_t ydlidar::protocol::read_response_health_t (Serial * *serial,* device_health & *health,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 368 of file ydlidar_protocol.cpp.

27.7.1.16 **result_t ydlidar::protocol::read_response_offset_angle_t (Serial * *serial,* offset_angle_t & *angle,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 454 of file ydlidar_protocol.cpp.

27.7.1.17 **result_t ydlidar::protocol::read_response_sample_rate_t (Serial * *serial,* sampling_rate_t & *rate,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 411 of file ydlidar_protocol.cpp.

27.7.1.18 **result_t ydlidar::protocol::read_response_scan_frequency_t (Serial * *serial,* scan_frequency_t & *frequency,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 432 of file ydlidar_protocol.cpp.

27.7.1.19 **result_t ydlidar::protocol::read_response_scan_header_t (Serial * *serial,* node_package_header_t & *header,* ct_packet_t & *ct,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 729 of file ydlidar_protocol.cpp.

27.7.1.20 **result_t ydlidar::protocol::read_response_scan_intensity_t (Serial * *serial,* scan_intensity_packet_t & *scan,* ct_packet_t & *ct,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 840 of file ydlidar_protocol.cpp.

27.7.1.21 **result_t ydlidar::protocol::read_response_scan_t (Serial * *serial,* scan_packet_t & *scan,* ct_packet_t & *ct,* lidar_error_t & *error,* uint32_t *timeout* = 1000)**

Definition at line 813 of file ydlidar_protocol.cpp.

27.7.1.22 **void ydlidar::protocol::reset_ct_packet_t (ct_packet_t & *ct*)**

Definition at line 212 of file ydlidar_protocol.cpp.

27.7.1.23 **result_t ydlidar::protocol::wait_for_data (Serial * *serial,* size_t *data_count,* uint32_t *timeout* = 1000)**

Definition at line 242 of file ydlidar_protocol.cpp.

27.7.1.24 **void ydlidar::protocol::write_command (Serial * *serial,* uint8_t *cmd*)**

Definition at line 233 of file ydlidar_protocol.cpp.

Chapter 28

Class Documentation

28.1 cmd_packet_t Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t syncByte
- uint8_t cmd_flag
- uint8_t size
- uint8_t data

28.1.1 Detailed Description

Definition at line 202 of file ydlidar_def.h.

28.1.2 Member Data Documentation

28.1.2.1 uint8_t cmd_packet_t::cmd_flag

Definition at line 204 of file ydlidar_def.h.

28.1.2.2 uint8_t cmd_packet_t::data

Definition at line 206 of file ydlidar_def.h.

28.1.2.3 uint8_t cmd_packet_t::size

Definition at line 205 of file ydlidar_def.h.

28.1.2.4 uint8_t cmd_packet_t::syncByte

Definition at line 203 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.2 ct_packet_t Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t size
- uint8_t index
- uint8_t info [100]
- uint8_t crc
- uint8_t cs
- uint8_t valid

28.2.1 Detailed Description

Definition at line 221 of file ydlidar_def.h.

28.2.2 Member Data Documentation

28.2.2.1 uint8_t ct_packet_t::crc

Definition at line 225 of file ydlidar_def.h.

28.2.2.2 uint8_t ct_packet_t::cs

Definition at line 226 of file ydlidar_def.h.

28.2.2.3 uint8_t ct_packet_t::index

Definition at line 223 of file ydlidar_def.h.

28.2.2.4 uint8_t ct_packet_t::info[100]

Definition at line 224 of file ydlidar_def.h.

28.2.2.5 uint8_t ct_packet_t::size

Definition at line 222 of file ydlidar_def.h.

28.2.2.6 uint8_t ct_packet_t::valid

Definition at line 227 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.3 CYdLidar Class Reference

```
#include <CYdLidar.h>
```

Public Member Functions

- CYdLidar ()

  Constructor.
- virtual ~CYdLidar ()

  Destructor: turns the laser off.
- bool setlidaropt (int optname, const void *optval, int optlen)

  set lidar properties
- bool getlidaropt (int optname, void *optval, int optlen)

  get lidar property
- bool initialize ()

  Initialize the SDK and LiDAR.
- void GetLidarVersion (LidarVersion &version)

  Return LiDAR's version information in a numeric form.
- bool turnOn ()

  Start the device scanning routine which runs on a separate thread and enable motor.
- bool doProcessSimple (LaserScan &scan_msg, bool &hardwareError)

  Get the LiDAR Scan Data. turnOn is successful before doProcessSimple scan data.
- bool turnOff ()

  Stop the device scanning thread and disable motor.
- void disconnecting ()

  Uninitialize the SDK and Disconnect the LiDAR.
- lidar_error_t getDriverError () const

  Get the last error information of a (lidar or serial)

Protected Member Functions

- bool checkCOMMs ()
- bool checkStatus ()
- bool checkScanFrequency ()

  checkScanFrequency
- bool checkZeroOffsetAngle ()

  checkZeroOffsetAngle
- bool checkHardware ()
- bool checkHealth (const ct_packet_t &info)

  checkHealth
- bool checkLidarAbnormal ()
- bool getDeviceHealth (uint32_t timeout=500)
- bool getDeviceInfo (uint32_t timeout=500)

28.3.1 Detailed Description

"Dataset"

LIDAR	Model	Baudrate	Sample↩ Rate(K)	Range(m)	Frequency HZ)	Intenstiy(b	Single↩ Channel	voltage(↩ V)
S2-Pro	4	115200	3	0.12∼8	5∼8	false	false	4.8∼5.2

Dataset

example: S2-Pro LiDAR

```
CYdLidar laser;
std::string port = "/dev/ydlidar";
laser.setlidaropt(LidarPropSerialPort, port.c_str(), port.size());
std::string ignore_array;
ignore_array.clear();
laser.setlidaropt(LidarPropIgnoreArray, ignore_array.c_str(),
                  ignore_array.size());

int optval = 115200;
laser.setlidaropt(LidarPropSerialBaudrate, &optval, sizeof(int));
optval = 4;
laser.setlidaropt(LidarPropAbnormalCheckCount, &optval, sizeof(int));

bool b_optvalue = false;
laser.setlidaropt(LidarPropFixedResolution, &b_optvalue, sizeof(bool));
laser.setlidaropt(LidarPropReversion, &b_optvalue, sizeof(bool));
laser.setlidaropt(LidarPropInverted, &b_optvalue, sizeof(bool));
b_optvalue = true;
laser.setlidaropt(LidarPropAutoReconnect, &b_optvalue, sizeof(bool));

float f_optvalue = 180.0f;
laser.setlidaropt(LidarPropMaxAngle, &f_optvalue, sizeof(float));
f_optvalue = -180.0f;
laser.setlidaropt(LidarPropMinAngle, &f_optvalue, sizeof(float));

f_optvalue = 10.f;
laser.setlidaropt(LidarPropMaxRange, &f_optvalue, sizeof(float));
f_optvalue = 0.1f;
laser.setlidaropt(LidarPropMinRange, &f_optvalue, sizeof(float));
f_optvalue = 6.f
laser.setlidaropt(LidarPropScanFrequency, &f_optvalue, sizeof(float));
```

Definition at line 93 of file CYdLidar.h.

28.3.2 Constructor & Destructor Documentation

28.3.2.1 CYdLidar::CYdLidar ( )

Constructor.

Definition at line 39 of file CYdLidar.cpp.

28.3.2.2 CYdLidar::~CYdLidar ( ) [virtual]

Destructor: turns the laser off.

Definition at line 70 of file CYdLidar.cpp.

28.3.3 Member Function Documentation

28.3.3.1 bool CYdLidar::checkCOMMs ( ) [protected]

Returns true if communication has been established with the device. If it's not, try to create a comms channel.

Returns

false on error.

Definition at line 713 of file CYdLidar.cpp.

28.3.3.2 bool CYdLidar::checkHardware ( ) [protected]

Returns true if the normal scan runs with the device. If it's not,

Returns

false on error.

Definition at line 768 of file CYdLidar.cpp.

28.3.3.3 bool CYdLidar::checkHealth (const ct_packet_t & *info*) [protected]

checkHealth

Parameters

info	

Returns

Definition at line 428 of file CYdLidar.cpp.

28.3.3.4 bool CYdLidar::checkLidarAbnormal ( ) [protected]

returns true if the lidar data is normal, If it's not

Definition at line 507 of file CYdLidar.cpp.

28.3.3.5 bool CYdLidar::checkScanFrequency ( ) [protected]

checkScanFrequency

Returns

Definition at line 614 of file CYdLidar.cpp.

28.3.3.6 bool CYdLidar::checkStatus ( ) [protected]

Returns true if health status and device information has been obtained with the device. If it's not,

Returns

false on error.

Definition at line 690 of file CYdLidar.cpp.

28.3.3.7 bool CYdLidar::checkZeroOffsetAngle ( ) [protected]

checkZeroOffsetAngle

Returns

Definition at line 667 of file CYdLidar.cpp.

28.3.3.8 void CYdLidar::disconnecting ( )

Uninitialize the SDK and Disconnect the LiDAR.

Closes the comms with the laser. Shouldn't have to be directly needed by the user

Definition at line 300 of file CYdLidar.cpp.

28.3.3.9 bool CYdLidar::doProcessSimple (LaserScan & *scan_msg,* bool & *hardwareError*)

Get the LiDAR Scan Data. turnOn is successful before doProcessSimple scan data.

Parameters

out	*outscan*	LiDAR Scan Data
out	*hardwareError*	hardware error status

Returns

true if successfully started, otherwise false.

Definition at line 320 of file CYdLidar.cpp.

28.3.3.10 bool CYdLidar::getDeviceHealth (uint32_t *timeout* = 500) [protected]

Returns true if the device is in good health, If it's not

Definition at line 536 of file CYdLidar.cpp.

28.3.3.11 bool CYdLidar::getDeviceInfo (uint32_t *timeout* = 500) [protected]

Returns true if the device information is correct, If it's not

Definition at line 565 of file CYdLidar.cpp.

28.3.3.12 lidar_error_t CYdLidar::getDriverError ( ) const

Get the last error information of a (lidar or serial)

Returns

a human-readable description of the given error information or the last error information of a (lidar or serial)

Definition at line 309 of file CYdLidar.cpp.

28.3.3.13 bool CYdLidar::getlidaropt (int *optname*, void * *optval*, int *optlen*)

get lidar property

Parameters

optname	option name

Todo string properties

- LidarPropSerialPort
- LidarPropIgnoreArray

Note

get string property example

```
CYdLidar laser;
char lidar_port[30];
laser.getlidaropt(LidarPropSerialPort,lidar_port, sizeof(lidar_port));
```

Todo int properties

- LidarPropSerialBaudrate
- LidarPropLidarType
- LidarPropDeviceType
- LidarPropSampleRate

Note

get int property example

```
CYdLidar laser;
int lidar_baudrate;
laser.getlidaropt(LidarPropSerialPort,&lidar_baudrate, sizeof(int));
```

Todo bool properties

- LidarPropFixedResolution
- LidarPropReversion
- LidarPropInverted
- LidarPropAutoReconnect
- LidarPropSingleChannel
- LidarPropIntenstiy

Note

get bool property example

```
CYdLidar laser;
bool lidar_fixedresolution;
laser.getlidaropt(LidarPropSerialPort,&lidar_fixedresolution, sizeof(bool));
```

Todo float properties

- LidarPropMaxRange
- LidarPropMinRange
- LidarPropMaxAngle
- LidarPropMinAngle
- LidarPropScanFrequency

Note

set float property example

```
CYdLidar laser;
float lidar_maxrange;
laser.getlidaropt(LidarPropSerialPort,&lidar_maxrange, sizeof(float));
```

Parameters

optval	option value • std::string(or char∗) • int • bool • float
optlen	option length • data type size

Returns

true if the Property is get successfully, otherwise false.

See also

LidarProperty

Definition at line 187 of file CYdLidar.cpp.

28.3.3.14 void CYdLidar::GetLidarVersion (LidarVersion & *version*)

Return LiDAR's version information in a numeric form.

Parameters

version	Pointer to a version structure for returning the version information.

Definition at line 296 of file CYdLidar.cpp.

28.3.3.15 bool CYdLidar::initialize ( )

Initialize the SDK and LiDAR.

Returns

true if successfully initialized, otherwise false.

Definition at line 783 of file CYdLidar.cpp.

28.3.3.16 bool CYdLidar::setlidaropt (int *optname*, const void ∗ *optval*, int *optlen*)

set lidar properties

Parameters

optname	option name

Todo string properties

- LidarPropSerialPort
- LidarPropIgnoreArray

Note

set string property example

```
CYdLidar laser;
std::string lidar_port = "/dev/ydlidar";
laser.setlidaropt(LidarPropSerialPort,lidar_port.c_str(), lidar_port.size());
```

Todo int properties

- LidarPropSerialBaudrate
- LidarPropLidarType
- LidarPropDeviceType
- LidarPropSampleRate

Note

set int property example

```
CYdLidar laser;
int lidar_baudrate = 230400;
laser.setlidaropt(LidarPropSerialPort,&lidar_baudrate, sizeof(int));
```

Todo bool properties

- LidarPropFixedResolution
- LidarPropReversion
- LidarPropInverted
- LidarPropAutoReconnect
- LidarPropSingleChannel
- LidarPropIntenstiy

Note

set bool property example

```
CYdLidar laser;
bool lidar_fixedresolution = true;
laser.setlidaropt(LidarPropSerialPort,&lidar_fixedresolution, sizeof(bool));
```

Todo float properties

- LidarPropMaxRange
- LidarPropMinRange
- LidarPropMaxAngle
- LidarPropMinAngle
- LidarPropScanFrequency

Note

set float property example, Must be float type, not double type.

```
CYdLidar laser;
float lidar_maxrange = 16.0f;
laser.setlidaropt(LidarPropSerialPort,&lidar_maxrange, sizeof(float));
```

Parameters

optval	option value • std::string(or char*) • int • bool • float
optlen	option length • data type size

Returns

true if the Property is set successfully, otherwise false.

See also

LidarProperty

Definition at line 74 of file CYdLidar.cpp.

28.3.3.17 bool CYdLidar::turnOff ( )

Stop the device scanning thread and disable motor.

Returns

true if successfully Stoped, otherwise false.See base class docs

Definition at line 490 of file CYdLidar.cpp.

28.3.3.18 bool CYdLidar::turnOn ( )

Start the device scanning routine which runs on a separate thread and enable motor.

Returns

true if successfully started, otherwise false.See base class docs

Definition at line 445 of file CYdLidar.cpp.

The documentation for this class was generated from the following files:

- include/CYdLidar.h
- src/CYdLidar.cpp

28.4 device_health Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t status
  健康状体
- uint16_t error_code
  错误代码

28.4.1 Detailed Description

Definition at line 184 of file ydlidar_def.h.

28.4.2 Member Data Documentation

28.4.2.1 uint16_t device_health::error_code

错误代码

Definition at line 186 of file ydlidar_def.h.

28.4.2.2 uint8_t device_health::status

健康状体

Definition at line 185 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.5 device_info Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t model
  雷达型号
- uint16_t firmware_version
  固件版本号
- uint8_t hardware_version
  硬件版本号
- uint8_t serialnum [16]
  系列号

28.5.1 Detailed Description

Definition at line 175 of file ydlidar_def.h.

28.5.2 Member Data Documentation

28.5.2.1 uint16_t device_info::firmware_version

固件版本号

Definition at line 177 of file ydlidar_def.h.

28.5.2.2 uint8_t device_info::hardware_version

硬件版本号

Definition at line 178 of file ydlidar_def.h.

28.5.2.3 uint8_t device_info::model

雷达型号

Definition at line 176 of file ydlidar_def.h.

28.5.2.4 uint8_t device_info::serialnum[16]

系列号

Definition at line 179 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.6 Event Class Reference

```
#include <locker.h>
```

Public Types

- enum { EVENT_OK = 1, EVENT_TIMEOUT = 2, EVENT_FAILED = 0 }

Public Member Functions

- Event (bool isAutoReset=true, bool isSignal=false)
- ~Event ()
- void set (bool isSignal=true)
- unsigned long wait (unsigned long timeout=0xFFFFFFFF)

Protected Member Functions

- void release ()

Protected Attributes

- pthread_condattr_t _cond_cattr
- pthread_cond_t _cond_var
- pthread_mutex_t _cond_locker
- bool _is_signalled
- bool _isAutoReset

28.6.1 Detailed Description

Definition at line 184 of file locker.h.

28.6.2 Member Enumeration Documentation

28.6.2.1 anonymous enum

Enumerator

EVENT_OK

EVENT_TIMEOUT

EVENT_FAILED

Definition at line 187 of file locker.h.

28.6.3 Constructor & Destructor Documentation

28.6.3.1 Event::Event (bool *isAutoReset* = `true`, bool *isSignal* = `false`) [inline],[explicit]

Definition at line 193 of file locker.h.

28.6.3.2 Event::~Event ( ) [inline]

Definition at line 212 of file locker.h.

28.6.4 Member Function Documentation

28.6.4.1 void Event::release ( ) [inline], [protected]

Definition at line 310 of file locker.h.

28.6.4.2 void Event::set (bool *isSignal* = true) [inline]

Definition at line 216 of file locker.h.

28.6.4.3 unsigned long Event::wait (unsigned long *timeout* = 0xFFFFFFFF) [inline]

Definition at line 241 of file locker.h.

28.6.5 Member Data Documentation

28.6.5.1 pthread_condattr_t Event::_cond_cattr [protected]

Definition at line 323 of file locker.h.

28.6.5.2 pthread_mutex_t Event::_cond_locker [protected]

Definition at line 325 of file locker.h.

28.6.5.3 pthread_cond_t Event::_cond_var [protected]

Definition at line 324 of file locker.h.

28.6.5.4 bool Event::_is_signalled [protected]

Definition at line 326 of file locker.h.

28.6.5.5 bool Event::_isAutoReset [protected]

Definition at line 327 of file locker.h.

The documentation for this class was generated from the following file:

- include/locker.h

28.7 LaserConfig Struct Reference

A struct for returning configuration from the YDLIDAR.

`#include <ydlidar_def.h>`

Public Member Functions

- LaserConfig & operator= (const LaserConfig &data)

Public Attributes

- float min_angle

  Start angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
- float max_angle

  Stop angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
- float angle_increment

  angle resoltuion [rad]
- float time_increment

  Scan resoltuion [s].
- float scan_time

  Time between scans.
- float min_range

  Minimum range [m].
- float max_range

  Maximum range [m].

28.7.1 Detailed Description

A struct for returning configuration from the YDLIDAR.

Definition at line 273 of file ydlidar_def.h.

28.7.2 Member Function Documentation

28.7.2.1 LaserConfig& LaserConfig::operator= (const LaserConfig & *data*) [inline]

Definition at line 289 of file ydlidar_def.h.

28.7.3 Member Data Documentation

28.7.3.1 float LaserConfig::angle_increment

angle resoltuion [rad]

Definition at line 279 of file ydlidar_def.h.

28.7.3.2 float LaserConfig::max_angle

Stop angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.

Definition at line 277 of file ydlidar_def.h.

28.7.3.3 float LaserConfig::max_range

Maximum range [m].

Definition at line 287 of file ydlidar_def.h.

28.7.3.4 float LaserConfig::min_angle

Start angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.

Definition at line 275 of file ydlidar_def.h.

28.7.3.5 float LaserConfig::min_range

Minimum range [m].

Definition at line 285 of file ydlidar_def.h.

28.7.3.6 float LaserConfig::scan_time

Time between scans.

Definition at line 283 of file ydlidar_def.h.

28.7.3.7 float LaserConfig::time_increment

Scan resoltuion [s].

Definition at line 281 of file ydlidar_def.h.

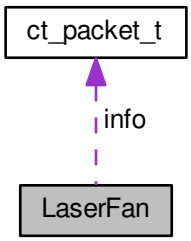
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.8 LaserFan Struct Reference

`#include <ydlidar_def.h>`

Collaboration diagram for LaserFan:



Public Member Functions

- LaserFan & operator= (const LaserFan &data)

Public Attributes

- uint8_t sync_flag
- ct_packet_t info

  Array of lidar points.
- std::vector< LaserPoint > points

28.8.1 Detailed Description

Definition at line 258 of file ydlidar_def.h.

28.8.2 Member Function Documentation

28.8.2.1 LaserFan& LaserFan::operator= (const LaserFan & *data*) [inline]

Definition at line 263 of file ydlidar_def.h.

28.8.3 Member Data Documentation

28.8.3.1 ct_packet_t LaserFan::info

Array of lidar points.

Definition at line 261 of file ydlidar_def.h.

28.8.3.2 std::vector<LaserPoint> LaserFan::points

Definition at line 262 of file ydlidar_def.h.

28.8.3.3 uint8_t LaserFan::sync_flag

Definition at line 259 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.9 LaserPoint Struct Reference

`#include <ydlidar_def.h>`

Public Member Functions

- LaserPoint & operator= (const LaserPoint &data)

Public Attributes

- float angle
- float range
- uint16_t intensity

28.9.1 Detailed Description

Definition at line 246 of file ydlidar_def.h.

28.9.2 Member Function Documentation

28.9.2.1 LaserPoint& LaserPoint::operator= (const LaserPoint & *data*) [inline]

Definition at line 250 of file ydlidar_def.h.

28.9.3 Member Data Documentation

28.9.3.1 float LaserPoint::angle

Definition at line 247 of file ydlidar_def.h.

28.9.3.2 uint16_t LaserPoint::intensity

Definition at line 249 of file ydlidar_def.h.

28.9.3.3 float LaserPoint::range

Definition at line 248 of file ydlidar_def.h.

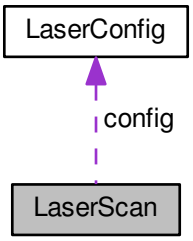
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.10 LaserScan Struct Reference

`#include <ydlidar_def.h>`

Collaboration diagram for LaserScan:



Public Member Functions

- LaserScan & operator= (const LaserScan &data)

Public Attributes

- uint64_t stamp

  System time when first range was measured in nanoseconds.
- std::vector< LaserPoint > points

  Array of laser point.
- LaserConfig config

  Configuration of scan.

28.10.1 Detailed Description

Definition at line 302 of file ydlidar_def.h.

28.10.2 Member Function Documentation

28.10.2.1 LaserScan& LaserScan::operator= (const LaserScan & *data*) [inline]

Definition at line 309 of file ydlidar_def.h.

28.10.3 Member Data Documentation

28.10.3.1 LaserConfig LaserScan::config

Configuration of scan.

Definition at line 308 of file ydlidar_def.h.

28.10.3.2 std::vector<LaserPoint> LaserScan::points

Array of laser point.

Definition at line 306 of file ydlidar_def.h.

28.10.3.3 uint64_t LaserScan::stamp

System time when first range was measured in nanoseconds.

Definition at line 304 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.11 lidar_ans_header_t Struct Reference

LiDAR response Header.

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t syncByte1
- uint8_t syncByte2
- uint32_t size: 30
- uint32_t subType: 2
- uint8_t type

28.11.1 Detailed Description

LiDAR response Header.

Definition at line 210 of file ydlidar_def.h.

28.11.2 Member Data Documentation

28.11.2.1 uint32_t lidar_ans_header_t::size

Definition at line 213 of file ydlidar_def.h.

28.11.2.2 uint32_t lidar_ans_header_t::subType

Definition at line 214 of file ydlidar_def.h.

28.11.2.3 uint8_t lidar_ans_header_t::syncByte1

Definition at line 211 of file ydlidar_def.h.

28.11.2.4 uint8_t lidar_ans_header_t::syncByte2

Definition at line 212 of file ydlidar_def.h.

28.11.2.5 uint8_t lidar_ans_header_t::type

Definition at line 215 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.12 LidarVersion Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t hardware
- uint8_t soft_major
- uint8_t soft_minor
- uint8_t soft_patch
- uint8_t sn [16]

28.12.1 Detailed Description

The numeric version information struct.

Definition at line 235 of file ydlidar_def.h.

28.12.2 Member Data Documentation

28.12.2.1 uint8_t LidarVersion::hardware

Hardware version

Definition at line 236 of file ydlidar_def.h.

28.12.2.2 uint8_t LidarVersion::sn[16]

serial number

Definition at line 240 of file ydlidar_def.h.

28.12.2.3 uint8_t LidarVersion::soft_major

major number

Definition at line 237 of file ydlidar_def.h.

28.12.2.4 uint8_t LidarVersion::soft_minor

minor number

Definition at line 238 of file ydlidar_def.h.

28.12.2.5 uint8_t LidarVersion::soft_patch

patch number

Definition at line 239 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.13 Locker Class Reference

```
#include <locker.h>
```

Public Types

- enum LOCK_STATUS { LOCK_OK = 0, LOCK_TIMEOUT = -1, LOCK_FAILED = -2 }

Public Member Functions

- Locker ()
- ~Locker ()
- Locker::LOCK_STATUS lock (unsigned long timeout=0xFFFFFFFF)
- void unlock ()
- pthread_mutex_t * getLockHandle ()

Protected Member Functions

- void init ()
- void release ()

Protected Attributes

- pthread_mutex_t _lock

28.13.1 Detailed Description

Definition at line 20 of file locker.h.

28.13.2 Member Enumeration Documentation

28.13.2.1 enum Locker::LOCK_STATUS

Enumerator

LOCK_OK

LOCK_TIMEOUT

LOCK_FAILED

Definition at line 22 of file locker.h.

28.13.3 Constructor & Destructor Documentation

28.13.3.1 Locker::Locker ( ) [inline]

Definition at line 28 of file locker.h.

28.13.3.2 Locker::~Locker ( ) [inline]

Definition at line 35 of file locker.h.

28.13.4 Member Function Documentation

28.13.4.1 pthread_mutex_t* Locker::getLockHandle ( ) [inline]

Definition at line 146 of file locker.h.

28.13.4.2 void Locker::init ( ) [inline], [protected]

Definition at line 154 of file locker.h.

28.13.4.3 Locker::LOCK_STATUS Locker::lock (unsigned long *timeout* = 0xFFFFFFFF) [inline]

Definition at line 39 of file locker.h.

28.13.4.4 void Locker::release ( ) [inline], [protected]

Definition at line 162 of file locker.h.

28.13.4.5 void Locker::unlock ( ) [inline]

Definition at line 133 of file locker.h.

28.13.5 Member Data Documentation

28.13.5.1 pthread_mutex_t Locker::_lock [protected]

Definition at line 179 of file locker.h.

The documentation for this class was generated from the following file:

- include/locker.h

28.14 serial::MillisecondTimer Class Reference

```
#include <unix_serial.h>
```

Public Member Functions

- MillisecondTimer (const uint32_t millis)
- int64_t remaining ()

28.14.1 Detailed Description

Definition at line 17 of file unix_serial.h.

28.14.2 Constructor & Destructor Documentation

28.14.2.1 MillisecondTimer::MillisecondTimer (const uint32_t *millis*) [explicit]

Definition at line 228 of file unix_serial.cpp.

28.14.3 Member Function Documentation

28.14.3.1 int64_t MillisecondTimer::remaining ( )

Definition at line 241 of file unix_serial.cpp.

The documentation for this class was generated from the following files:

- src/impl/unix/unix_serial.h
- src/impl/unix/unix_serial.cpp

28.15 node_package_header_t Struct Reference

LiDAR Intensity Nodes Package.

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t packageHeaderMSB
  package header MSB
- uint8_t packageHeaderLSB
  package header LSB
- uint8_t packageSync: 1
  package sync flag
- uint8_t packageCTInfo: 7
  package ct info
- uint8_t nowPackageNum
  package number
- uint16_t packageFirstSampleAngleSync: 1
- uint16_t packageFirstSampleAngle: 15
  first sample angle sync flag
- uint16_t packageLastSampleAngleSync: 1
  last sample angle sync flag
- uint16_t packageLastSampleAngle: 15
  last sample angle
- uint16_t checkSum
  checksum

28.15.1 Detailed Description

LiDAR Intensity Nodes Package.

Definition at line 96 of file ydlidar_def.h.

28.15.2 Member Data Documentation

28.15.2.1 uint16_t node_package_header_t::checkSum

checksum

Definition at line 106 of file ydlidar_def.h.

28.15.2.2 uint8_t node_package_header_t::nowPackageNum

package number

Definition at line 101 of file ydlidar_def.h.

28.15.2.3 uint8_t node_package_header_t::packageCTInfo

package ct info

Definition at line 100 of file ydlidar_def.h.

28.15.2.4 uint16_t node_package_header_t::packageFirstSampleAngle

first sample angle sync flag

first sample angle

Definition at line 103 of file ydlidar_def.h.

28.15.2.5 uint16_t node_package_header_t::packageFirstSampleAngleSync

Definition at line 102 of file ydlidar_def.h.

28.15.2.6 uint8_t node_package_header_t::packageHeaderLSB

package header LSB

Definition at line 98 of file ydlidar_def.h.

28.15.2.7 uint8_t node_package_header_t::packageHeaderMSB

package header MSB

Definition at line 97 of file ydlidar_def.h.

28.15.2.8 uint16_t node_package_header_t::packageLastSampleAngle

last sample angle

Definition at line 105 of file ydlidar_def.h.

28.15.2.9 uint16_t node_package_header_t::packageLastSampleAngleSync

last sample angle sync flag

Definition at line 104 of file ydlidar_def.h.

28.15.2.10 uint8_t node_package_header_t::packageSync

package sync flag

Definition at line 99 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.16 node_package_intensity_payload_t Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t PackageSampleIntensity
- uint16_t PackageSampleDistance

  intensity

28.16.1 Detailed Description

Definition at line 141 of file ydlidar_def.h.

28.16.2 Member Data Documentation

28.16.2.1 uint16_t node_package_intensity_payload_t::PackageSampleDistance

intensity

range

Definition at line 143 of file ydlidar_def.h.

28.16.2.2 uint8_t node_package_intensity_payload_t::PackageSampleIntensity

Definition at line 142 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.17 node_package_payload_t Struct Reference

package node info

```
#include <ydlidar_def.h>
```

Public Attributes

- uint16_t PackageSampleSi: 2
  si
- uint16_t PackageSampleDistance: 14
  range

28.17.1 Detailed Description

package node info

Definition at line 159 of file ydlidar_def.h.

28.17.2 Member Data Documentation

28.17.2.1 uint16_t node_package_payload_t::PackageSampleDistance

range

Definition at line 161 of file ydlidar_def.h.

28.17.2.2 uint16_t node_package_payload_t::PackageSampleSi

si

Definition at line 160 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.18 offset_angle_t Struct Reference

LiDAR Zero Offset Angle.

`#include <ydlidar_def.h>`

Public Attributes

- int32_t angle

28.18.1 Detailed Description

LiDAR Zero Offset Angle.

Definition at line 198 of file ydlidar_def.h.

28.18.2 Member Data Documentation

28.18.2.1 int32_t offset_angle_t::angle

Definition at line 199 of file ydlidar_def.h.

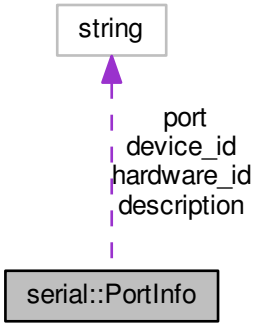
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.19 serial::PortInfo Struct Reference

`#include <serial.h>`

Collaboration diagram for serial::PortInfo:



Public Attributes

- std::string port
- std::string description
- std::string hardware_id
- std::string device_id

28.19.1 Detailed Description

Structure that describes a serial device.

Definition at line 581 of file serial.h.

28.19.2 Member Data Documentation

28.19.2.1 std::string serial::PortInfo::description

Human readable description of serial device if available.

Definition at line 587 of file serial.h.

28.19.2.2 std::string serial::PortInfo::device_id

Hardware Device ID or "" if not available.

Definition at line 593 of file serial.h.

28.19.2.3 std::string serial::PortInfo::hardware_id

Hardware ID (e.g. VID:PID of USB serial devices) or "n/a" if not available.

Definition at line 590 of file serial.h.

28.19.2.4 std::string serial::PortInfo::port

Address of the serial port (this can be passed to the constructor of Serial).

Definition at line 584 of file serial.h.

The documentation for this struct was generated from the following file:

- include/serial.h

28.20 sampling_rate_t Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint8_t rate
  采样频率

28.20.1 Detailed Description

Definition at line 189 of file ydlidar_def.h.

28.20.2 Member Data Documentation

28.20.2.1 uint8_t sampling_rate_t::rate

采样频率

Definition at line 190 of file ydlidar_def.h.

The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.21 scan_frequency_t Struct Reference

`#include <ydlidar_def.h>`

Public Attributes

- uint32_t frequency
  扫描频率

28.21.1 Detailed Description

Definition at line 193 of file ydlidar_def.h.

28.21.2 Member Data Documentation

28.21.2.1 uint32_t scan_frequency_t::frequency

扫描频率

Definition at line 194 of file ydlidar_def.h.

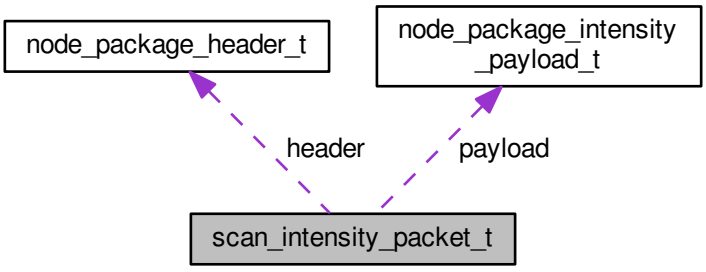
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.22 scan_intensity_packet_t Struct Reference

`#include <ydlidar_def.h>`

Collaboration diagram for scan_intensity_packet_t:



Public Attributes

- node_package_header_t header
- node_package_intensity_payload_t payload [40]

28.22.1 Detailed Description

Definition at line 149 of file ydlidar_def.h.

28.22.2 Member Data Documentation

28.22.2.1 node_package_header_t scan_intensity_packet_t::header

Definition at line 150 of file ydlidar_def.h.

28.22.2.2 node_package_intensity_payload_t scan_intensity_packet_t::payload[40]

Definition at line 151 of file ydlidar_def.h.

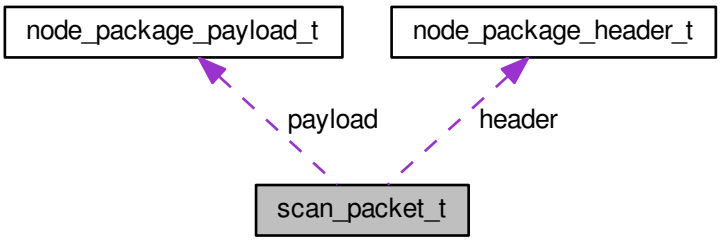
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.23 scan_packet_t Struct Reference

`#include <ydlidar_def.h>`

Collaboration diagram for scan_packet_t:



Public Attributes

- node_package_header_t header
- node_package_payload_t payload [40]

28.23.1 Detailed Description

Definition at line 166 of file ydlidar_def.h.

28.23.2 Member Data Documentation

28.23.2.1 node_package_header_t scan_packet_t::header

Definition at line 167 of file ydlidar_def.h.

28.23.2.2 node_package_payload_t scan_packet_t::payload[40]

Definition at line 168 of file ydlidar_def.h.

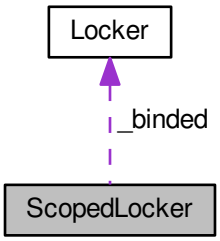
The documentation for this struct was generated from the following file:

- include/ydlidar_def.h

28.24 ScopedLocker Class Reference

`#include <locker.h>`

Collaboration diagram for ScopedLocker:



Public Member Functions

- ScopedLocker (Locker &l)
- void forceUnlock ()
- ~ScopedLocker ()

Public Attributes

- Locker & _binded

28.24.1 Detailed Description

Definition at line 331 of file locker.h.

28.24.2 Constructor & Destructor Documentation

28.24.2.1 ScopedLocker::ScopedLocker (Locker & *l*) [inline],[explicit]

Definition at line 333 of file locker.h.

28.24.2.2 ScopedLocker::~ScopedLocker ( ) [inline]

Definition at line 340 of file locker.h.

28.24.3 Member Function Documentation

28.24.3.1 void ScopedLocker::forceUnlock ( ) [inline]

Definition at line 337 of file locker.h.

28.24.4 Member Data Documentation

28.24.4.1 Locker& ScopedLocker::_binded

Definition at line 343 of file locker.h.

The documentation for this class was generated from the following file:

- include/locker.h

28.25 serial::Serial::ScopedReadLock Class Reference

Public Member Functions

- ScopedReadLock (Serial::SerialImpl *pimpl)
- ~ScopedReadLock ()

28.25.1 Detailed Description

Definition at line 28 of file serial.cpp.

28.25.2 Constructor & Destructor Documentation

28.25.2.1 serial::Serial::ScopedReadLock::ScopedReadLock (Serial::SerialImpl * *pimpl*) [inline],[explicit]

Definition at line 30 of file serial.cpp.

28.25.2.2 serial::Serial::ScopedReadLock::~ScopedReadLock ( ) [inline]

Definition at line 33 of file serial.cpp.

The documentation for this class was generated from the following file:

- src/serial.cpp

28.26 serial::Serial::ScopedWriteLock Class Reference

Public Member Functions

- ScopedWriteLock (Serial::SerialImpl *pimpl)
- ~ScopedWriteLock ()

28.26.1 Detailed Description

Definition at line 44 of file serial.cpp.

28.26.2 Constructor & Destructor Documentation

28.26.2.1 serial::Serial::ScopedWriteLock::ScopedWriteLock (Serial::SerialImpl * *pimpl*) [inline], [explicit]

Definition at line 46 of file serial.cpp.

28.26.2.2 serial::Serial::ScopedWriteLock::~ScopedWriteLock ( ) [inline]

Definition at line 49 of file serial.cpp.

The documentation for this class was generated from the following file:

- src/serial.cpp

28.27 serial::Serial Class Reference

```
#include <serial.h>
```

Classes

- class ScopedReadLock
- class ScopedWriteLock
- class SerialImpl

Public Member Functions

- Serial (const std::string &port="", uint32_t baudrate=9600, Timeout timeout=Timeout(), bytesize_t bytesize=eightbits, parity_t parity=parity_none, stopbits_t stopbits=stopbits_one, flowcontrol_t flowcontrol=flowcontrol_none)
- virtual ~Serial ()
- bool open ()
- bool isOpen ()
- void closePort ()
- size_t available ()
- bool waitReadable ()
- void waitByteTimes (size_t count)
- int waitfordata (size_t data_count, uint32_t timeout, size_t *returned_size)

  waitfordata
- size_t read (uint8_t *buffer, size_t size)
- size_t read (std::vector< uint8_t > &buffer, size_t size=1)
- size_t read (std::string &buffer, size_t size=1)
- std::string read (size_t size=1)
- size_t readline (std::string &buffer, size_t size=65536, std::string eol="\n")
- std::string readline (size_t size=65536, std::string eol="\n")
- std::vector< std::string > readlines (size_t size=65536, std::string eol="\n")
- size_t write (const uint8_t *data, size_t size)
- size_t write (const std::vector< uint8_t > &data)
- size_t write (const std::string &data)
- void setPort (const std::string &port)
- std::string getPort () const
- void setTimeout (Timeout &timeout)
- void setTimeout (uint32_t inter_byte_timeout, uint32_t read_timeout_constant, uint32_t read_timeout_multiplier, uint32_t write_timeout_constant, uint32_t write_timeout_multiplier)
- Timeout getTimeout () const
- bool setBaudrate (uint32_t baudrate)
- uint32_t getBaudrate () const
- bool setBytesize (bytesize_t bytesize)
- bytesize_t getBytesize () const
- bool setParity (parity_t parity)
- parity_t getParity () const
- bool setStopbits (stopbits_t stopbits)
- stopbits_t getStopbits () const
- bool setFlowcontrol (flowcontrol_t flowcontrol)
- flowcontrol_t getFlowcontrol () const
- void flush ()
- void flushInput ()
- void flushOutput ()
- void sendBreak (int duration)
- bool setBreak (bool level=true)
- bool setRTS (bool level=true)
- bool setDTR (bool level=true)
- bool waitForChange ()
- bool getCTS ()
- bool getDSR ()
- bool getRI ()
- bool getCD ()
- int getByteTime ()

28.27.1 Detailed Description

Class that provides a portable serial port interface.

Definition at line 107 of file serial.h.

28.27.2 Constructor & Destructor Documentation

28.27.2.1 serial::Serial::Serial (const std::string & *port* = " ", uint32_t *baudrate* = 9600, Timeout *timeout* = Timeout (), bytesize_t *bytesize* = eightbits, parity_t *parity* = parity_none, stopbits_t *stopbits* = stopbits_one, flowcontrol_t *flowcontrol* = flowcontrol_none) [explicit]

Creates a Serial object and opens the port if a port is specified, otherwise it remains closed until serial::Serial::open is called.

Parameters

port	A std::string containing the address of the serial port, which would be something like 'COM1' on Windows and '/dev/ttyS0' on Linux.
baudrate	An unsigned 32-bit integer that represents the baudrate
timeout	A serial::Timeout struct that defines the timeout conditions for the serial port.

See also

serial::Timeout

Parameters

bytesize	Size of each byte in the serial transmission of data, default is eightbits, possible values are: fivebits, sixbits, sevenbits, eightbits
parity	Method of parity, default is parity_none, possible values are: parity_none, parity_odd, parity_even
stopbits	Number of stop bits used, default is stopbits_one, possible values are: stopbits_one, stopbits_one_point_five, stopbits_two
flowcontrol	Type of flowcontrol used, default is flowcontrol_none, possible values are: flowcontrol_none, flowcontrol_software, flowcontrol_hardware

Exceptions

serial::PortNotOpenedException	
serial::IOException	
std::invalid_argument	

28.27.2.2 serial::Serial::~Serial ( ) [virtual]

Destructor

Definition at line 67 of file serial.cpp.

28.27.3 Member Function Documentation

28.27.3.1 size_t serial::Serial::available ( )

Return the number of characters in the buffer.

Definition at line 83 of file serial.cpp.

28.27.3.2 void serial::Serial::closePort ( )

Closes the serial port.

Definition at line 75 of file serial.cpp.

28.27.3.3 void serial::Serial::flush ( )

Flush the input and output buffers

Definition at line 297 of file serial.cpp.

28.27.3.4 void serial::Serial::flushInput ( )

Flush only the input buffer

Definition at line 303 of file serial.cpp.

28.27.3.5 void serial::Serial::flushOutput ( )

Flush only the output buffer

Definition at line 308 of file serial.cpp.

28.27.3.6 uint32_t serial::Serial::getBaudrate ( ) const

Gets the baudrate for the serial port.

Returns

An integer that sets the baud rate for the serial port.

See also

Serial::setBaudrate

\

Definition at line 261 of file serial.cpp.

28.27.3.7 bytesize_t serial::Serial::getBytesize ( ) const

Gets the bytesize for the serial port.

See also

Serial::setBytesize

\

Definition at line 269 of file serial.cpp.

28.27.3.8 int serial::Serial::getByteTime ( )

Returns the singal byte time.

Definition at line 349 of file serial.cpp.

28.27.3.9 bool serial::Serial::getCD ( )

Returns the current status of the CD line.

Definition at line 345 of file serial.cpp.

28.27.3.10 bool serial::Serial::getCTS ( )

Returns the current status of the CTS line.

Definition at line 333 of file serial.cpp.

28.27.3.11 bool serial::Serial::getDSR ( )

Returns the current status of the DSR line.

Definition at line 337 of file serial.cpp.

28.27.3.12 flowcontrol_t serial::Serial::getFlowcontrol ( ) const

Gets the flow control for the serial port.

See also

Serial::setFlowcontrol

\

Definition at line 293 of file serial.cpp.

28.27.3.13 parity_t serial::Serial::getParity ( ) const

Gets the parity for the serial port.

See also

Serial::setParity

\

Definition at line 277 of file serial.cpp.

28.27.3.14 string serial::Serial::getPort ( ) const

Gets the serial port identifier.

See also

Serial::setPort

Exceptions

std::invalid_argument	

Definition at line 245 of file serial.cpp.

28.27.3.15 bool serial::Serial::getRI ( )

Returns the current status of the RI line.

Definition at line 341 of file serial.cpp.

28.27.3.16 stopbits_t serial::Serial::getStopbits ( ) const

Gets the stopbits for the serial port.

See also

Serial::setStopbits

\

Definition at line 285 of file serial.cpp.

28.27.3.17 serial::Timeout serial::Serial::getTimeout ( ) const

Gets the timeout for reads in seconds.

Returns

A Timeout struct containing the inter_byte_timeout, and read and write timeout constants and multipliers.

See also

Serial::setTimeout

Definition at line 253 of file serial.cpp.

28.27.3.18 bool serial::Serial::isOpen ( )

Gets the open status of the serial port.

Returns

Returns true if the port is open, false otherwise.

Definition at line 79 of file serial.cpp.

28.27.3.19 bool serial::Serial::open ( )

Opens the serial port as long as the port is set and the port isn't already open.

If the port is provided to the constructor then an explicit call to open is not needed.

See also

Serial::Serial

Returns

Returns true if the port is open, false otherwise.

Definition at line 71 of file serial.cpp.

28.27.3.20 size_t serial::Serial::read (uint8_t * *buffer*, size_t *size*)

Read a given amount of bytes from the serial port into a given buffer.

The read function will return in one of three cases:

- The number of requested bytes was read.
  - In this case the number of bytes requested will match the size_t returned by read.
- A timeout occurred, in this case the number of bytes read will not match the amount requested, but no exception will be thrown. One of two possible timeouts occurred:
  - The inter byte timeout expired, this means that number of milliseconds elapsed between receiving bytes from the serial port exceeded the inter byte timeout.
  - The total timeout expired, which is calculated by multiplying the read timeout multiplier by the number of requested bytes and then added to the read timeout constant. If that total number of milliseconds elapses after the initial call to read a timeout will occur.
- An exception occurred, in this case an actual exception will be thrown.

Parameters

buffer	An uint8_t array of at least the requested size.
size	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

Definition at line 106 of file serial.cpp.

28.27.3.21 size_t serial::Serial::read (std::vector< uint8_t > & *buffer*, size_t *size* = 1)

Read a given amount of bytes from the serial port into a give buffer.

Parameters

buffer	A reference to a std::vector of uint8_t.
size	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

Definition at line 111 of file serial.cpp.

28.27.3.22 size_t serial::Serial::read (std::string & *buffer*, size_t *size* = 1)

Read a given amount of bytes from the serial port into a give buffer.

Parameters

buffer	A reference to a std::string.
size	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

Definition at line 119 of file serial.cpp.

28.27.3.23 string serial::Serial::read (size_t *size* = 1)

Read a given amount of bytes from the serial port and return a string containing the data.

Parameters

size	A size_t defining how many bytes to be read.

Returns

A std::string containing the data read from the port.

Definition at line 127 of file serial.cpp.

28.27.3.24 size_t serial::Serial::readline (std::string & *buffer*, size_t *size* = 65536, std::string *eol* = "\n")

Reads in a line or until a given delimiter has been processed.

Reads from the serial port until a single line has been read.

Parameters

buffer	A std::string reference used to store the data.
size	A maximum length of a line, defaults to 65536 (2^16)
eol	A string to match against for the EOL.

Returns

A size_t representing the number of bytes read.

28.27.3.25 std::string serial::Serial::readline (size_t *size* = 65536, std::string *eol* = "\n")

Reads in a line or until a given delimiter has been processed.

Reads from the serial port until a single line has been read.

Parameters

size	A maximum length of a line, defaults to 65536 (2^16)
eol	A string to match against for the EOL.

Returns

A std::string containing the line.

28.27.3.26 vector< string > serial::Serial::readlines (size_t *size* = 65536, std::string *eol* = "\n")

Reads in multiple lines until the serial port times out.

This requires a timeout > 0 before it can be run. It will read until a timeout occurs and return a list of strings.

Parameters

size	A maximum length of combined lines, defaults to 65536 (2^16)
eol	A string to match against for the EOL.

Returns

A vector<string> containing the lines.

Definition at line 167 of file serial.cpp.

28.27.3.27 void serial::Serial::sendBreak (int *duration*)

Sends the RS-232 break signal. See tcsendbreak(3).

Definition at line 313 of file serial.cpp.

28.27.3.28 bool serial::Serial::setBaudrate (uint32_t *baudrate*)

Sets the baudrate for the serial port.

Possible baudrates depends on the system but some safe baudrates include: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200 Some other baudrates that are supported by some comports: 128000, 153600, 230400, 256000, 460800, 921600

Parameters

baudrate	An integer that sets the baud rate for the serial port.

Definition at line 257 of file serial.cpp.

28.27.3.29 bool serial::Serial::setBreak (bool *level* = `true`)

Set the break condition to a given level. Defaults to true.

Definition at line 317 of file serial.cpp.

28.27.3.30 bool serial::Serial::setBytesize (bytesize_t *bytesize*)

Sets the bytesize for the serial port.

Parameters

bytesize	Size of each byte in the serial transmission of data, default is eightbits, possible values are: fivebits, sixbits, sevenbits, eightbits

\

Definition at line 265 of file serial.cpp.

28.27.3.31 bool serial::Serial::setDTR (bool *level* = `true`)

Set the DTR handshaking line to the given level. Defaults to true.

Definition at line 325 of file serial.cpp.

28.27.3.32 bool serial::Serial::setFlowcontrol (flowcontrol_t *flowcontrol*)

Sets the flow control for the serial port.

Parameters

flowcontrol	Type of flowcontrol used, default is flowcontrol_none, possible values are: flowcontrol_none, flowcontrol_software, flowcontrol_hardware

\

Definition at line 289 of file serial.cpp.

28.27.3.33 bool serial::Serial::setParity (parity_t *parity*)

Sets the parity for the serial port.

Parameters

parity	Method of parity, default is parity_none, possible values are: parity_none, parity_odd, parity_even

\

Definition at line 273 of file serial.cpp.

28.27.3.34 void serial::Serial::setPort (const std::string & *port*)

Sets the serial port identifier.

Parameters

port	A const std::string reference containing the address of the serial port, which would be something like 'COM1' on Windows and '/dev/ttyS0' on Linux.

Exceptions

std::invalid_argument	

Definition at line 229 of file serial.cpp.

28.27.3.35 bool serial::Serial::setRTS (bool *level* = `true`)

Set the RTS handshaking line to the given level. Defaults to true.

Definition at line 321 of file serial.cpp.

28.27.3.36 bool serial::Serial::setStopbits (stopbits_t *stopbits*)

Sets the stopbits for the serial port.

Parameters

stopbits	Number of stop bits used, default is stopbits_one, possible values are: stopbits_one, stopbits_one_point_five, stopbits_two

\

Definition at line 281 of file serial.cpp.

28.27.3.37 void serial::Serial::setTimeout (serial::Timeout & *timeout*)

Sets the timeout for reads and writes using the Timeout struct.

There are two timeout conditions described here:

- The inter byte timeout:
  - The inter_byte_timeout component of serial::Timeout defines the maximum amount of time, in milliseconds, between receiving bytes on the serial port that can pass before a timeout occurs. Setting this to zero will prevent inter byte timeouts from occurring.
- Total time timeout:
  - The constant and multiplier component of this timeout condition, for both read and write, are defined in serial::Timeout. This timeout occurs if the total time since the read or write call was made exceeds the specified time in milliseconds.
  - The limit is defined by multiplying the multiplier component by the number of requested bytes and adding that product to the constant component. In this way if you want a read call, for example, to timeout after exactly one second regardless of the number of bytes you asked for then set the read_timeout_constant component of serial::Timeout to 1000 and the read_timeout_multiplier to zero. This timeout condition can be used in conjunction with the inter byte timeout condition with out any problems, timeout will simply occur when one of the two timeout conditions is met. This allows users to have maximum control over the trade-off between responsiveness and efficiency.

Read and write functions will return in one of three cases. When the reading or writing is complete, when a timeout occurs, or when an exception occurs.

A timeout of 0 enables non-blocking mode.

Parameters

timeout	A serial::Timeout struct containing the inter byte timeout, and the read and write timeout constants and multipliers.

See also

serial::Timeout

Definition at line 249 of file serial.cpp.

28.27.3.38 void serial::Serial::setTimeout (uint32_t *inter_byte_timeout*, uint32_t *read_timeout_constant*, uint32_t *read_timeout_multiplier*, uint32_t *write_timeout_constant*, uint32_t *write_timeout_multiplier*) [inline]

Sets the timeout for reads and writes.

Definition at line 394 of file serial.h.

28.27.3.39 void serial::Serial::waitByteTimes (size_t *count*)

Block for a period of time corresponding to the transmission time of count characters at present serial settings. This may be used in con- junction with waitReadable to read larger blocks of data from the port.

Definition at line 92 of file serial.cpp.

28.27.3.40 bool serial::Serial::waitForChange ( )

Blocks until CTS, DSR, RI, CD changes or something interrupts it.

Can throw an exception if an error occurs while waiting. You can check the status of CTS, DSR, RI, and CD once this returns. Uses TIOCMIWAIT via ioctl if available (mostly only on Linux) with a resolution of less than +-1ms and as good as +-0.2ms. Otherwise a polling method is used which can give +-2ms.

Returns

Returns true if one of the lines changed, false if something else occurred.

Definition at line 329 of file serial.cpp.

28.27.3.41 int serial::Serial::waitfordata (size_t *data_count*, uint32_t *timeout*, size_t * *returned_size*)

waitfordata

Parameters

data_count	
timeout	
returned_size	

Returns

Definition at line 96 of file serial.cpp.

28.27.3.42 bool serial::Serial::waitReadable ( )

Block until there is serial data to read or read_timeout_constant number of milliseconds have elapsed. The return value is true when the function exits with the port in a readable state, false otherwise (due to timeout or select interruption).

Definition at line 87 of file serial.cpp.

28.27.3.43 size_t serial::Serial::write (const uint8_t * *data*, size_t *size*)

Write a string to the serial port.

Parameters

data	A const reference containing the data to be written to the serial port.
size	A size_t that indicates how many bytes should be written from the given data buffer.

Returns

A size_t representing the number of bytes actually written to the serial port.

Exceptions

serial::PortNotOpenedException	
serial::SerialException	
serial::IOException	

Definition at line 220 of file serial.cpp.

28.27.3.44 size_t serial::Serial::write (const std::vector< uint8_t > & *data*)

Write a string to the serial port.

Parameters

data	A const reference containing the data to be written to the serial port.

Returns

A size_t representing the number of bytes actually written to the serial port.

Definition at line 215 of file serial.cpp.

28.27.3.45 size_t serial::Serial::write (const std::string & *data*)

Write a string to the serial port.

Parameters

data	A const reference containing the data to be written to the serial port.

Returns

A size_t representing the number of bytes actually written to the serial port.

The documentation for this class was generated from the following files:

- include/serial.h
- src/serial.cpp

28.28 serial::Serial::SerialImpl Class Reference

`#include <unix_serial.h>`

Public Member Functions

- SerialImpl (const string &port, unsigned long baudrate, bytesize_t bytesize, parity_t parity, stopbits_t stopbits, flowcontrol_t flowcontrol)
- virtual ~SerialImpl ()
- bool open ()
- void close ()
- bool isOpen () const
- size_t available ()
- bool waitReadable (uint32_t timeout)
- void waitByteTimes (size_t count)
- int waitfordata (size_t data_count, uint32_t timeout, size_t *returned_size)
- size_t read (uint8_t *buf, size_t size=1)
- size_t write (const uint8_t *data, size_t length)
- void flush ()
- void flushInput ()
- void flushOutput ()
- void sendBreak (int duration)
- bool setBreak (bool level)
- bool setRTS (bool level)
- bool setDTR (bool level)
- bool waitForChange ()
- bool getCTS ()
- bool getDSR ()
- bool getRI ()
- bool getCD ()
- uint32_t getByteTime ()
- void setPort (const string &port)

- string getPort () const
- void setTimeout (Timeout &timeout)
- Timeout getTimeout () const
- bool setBaudrate (unsigned long baudrate)
- bool setStandardBaudRate (speed_t baudrate)
- bool setCustomBaudRate (unsigned long baudrate)
- unsigned long getBaudrate () const
- bool setBytesize (bytesize_t bytesize)
- bytesize_t getBytesize () const
- bool setParity (parity_t parity)
- parity_t getParity () const
- bool setStopbits (stopbits_t stopbits)
- stopbits_t getStopbits () const
- bool setFlowcontrol (flowcontrol_t flowcontrol)
- flowcontrol_t getFlowcontrol () const
- bool setTermios (const termios *tio)
- bool getTermios (termios *tio)
- int readLock ()
- int readUnlock ()
- int writeLock ()
- int writeUnlock ()

28.28.1 Detailed Description

Definition at line 27 of file unix_serial.h.

28.28.2 Constructor & Destructor Documentation

28.28.2.1 Serial::SerialImpl::SerialImpl (const string & *port*, unsigned long *baudrate*, bytesize_t *bytesize*, parity_t *parity*, stopbits_t *stopbits*, flowcontrol_t *flowcontrol*) [explicit]

Definition at line 637 of file unix_serial.cpp.

28.28.2.2 Serial::SerialImpl::~SerialImpl ( ) [virtual]

Definition at line 649 of file unix_serial.cpp.

28.28.3 Member Function Documentation

28.28.3.1 size_t Serial::SerialImpl::available ( )

Definition at line 751 of file unix_serial.cpp.

28.28.3.2 void Serial::SerialImpl::close ( )

Definition at line 732 of file unix_serial.cpp.

28.28.3.3 void Serial::SerialImpl::flush ( )

Definition at line 1319 of file unix_serial.cpp.

28.28.3.4 void Serial::SerialImpl::flushInput ( )

Definition at line 1329 of file unix_serial.cpp.

28.28.3.5 void Serial::SerialImpl::flushOutput ( )

Definition at line 1337 of file unix_serial.cpp.

28.28.3.6 unsigned long Serial::SerialImpl::getBaudrate ( ) const

Definition at line 1119 of file unix_serial.cpp.

28.28.3.7 serial::bytesize_t Serial::SerialImpl::getBytesize ( ) const

Definition at line 1238 of file unix_serial.cpp.

28.28.3.8 uint32_t Serial::SerialImpl::getByteTime ( )

Definition at line 1500 of file unix_serial.cpp.

28.28.3.9 bool Serial::SerialImpl::getCD ( )

Definition at line 1486 of file unix_serial.cpp.

28.28.3.10 bool Serial::SerialImpl::getCTS ( )

Definition at line 1444 of file unix_serial.cpp.

28.28.3.11 bool Serial::SerialImpl::getDSR ( )

Definition at line 1458 of file unix_serial.cpp.

28.28.3.12 serial::flowcontrol_t Serial::SerialImpl::getFlowcontrol ( ) const

Definition at line 1289 of file unix_serial.cpp.

28.28.3.13 serial::parity_t Serial::SerialImpl::getParity ( ) const

Definition at line 1255 of file unix_serial.cpp.

28.28.3.14 string Serial::SerialImpl::getPort ( ) const

Definition at line 1071 of file unix_serial.cpp.

28.28.3.15 bool Serial::SerialImpl::getRI ( )

Definition at line 1472 of file unix_serial.cpp.

28.28.3.16 serial::stopbits_t Serial::SerialImpl::getStopbits ( ) const

Definition at line 1272 of file unix_serial.cpp.

28.28.3.17 bool Serial::SerialImpl::getTermios (termios * *tio*)

Definition at line 1309 of file unix_serial.cpp.

28.28.3.18 serial::Timeout Serial::SerialImpl::getTimeout ( ) const

Definition at line 1079 of file unix_serial.cpp.

28.28.3.19 bool Serial::SerialImpl::isOpen ( ) const

Definition at line 747 of file unix_serial.cpp.

28.28.3.20 bool Serial::SerialImpl::open ( )

Definition at line 655 of file unix_serial.cpp.

28.28.3.21 size_t Serial::SerialImpl::read (uint8_t * *buf,* size_t *size* = 1)

Definition at line 885 of file unix_serial.cpp.

28.28.3.22 int Serial::SerialImpl::readLock ( )

Definition at line 1504 of file unix_serial.cpp.

28.28.3.23 int Serial::SerialImpl::readUnlock ( )

Definition at line 1509 of file unix_serial.cpp.

28.28.3.24 void Serial::SerialImpl::sendBreak (int *duration*)

Definition at line 1345 of file unix_serial.cpp.

28.28.3.25 bool Serial::SerialImpl::setBaudrate (unsigned long *baudrate*)

Definition at line 1083 of file unix_serial.cpp.

28.28.3.26 bool Serial::SerialImpl::setBreak (bool *level*)

Definition at line 1353 of file unix_serial.cpp.

28.28.3.27 bool Serial::SerialImpl::setBytesize (serial::bytesize_t *bytesize*)

Definition at line 1225 of file unix_serial.cpp.

28.28.3.28 bool Serial::SerialImpl::setCustomBaudRate (unsigned long *baudrate*)

Definition at line 1179 of file unix_serial.cpp.

28.28.3.29 bool Serial::SerialImpl::setDTR (bool *level*)

Definition at line 1391 of file unix_serial.cpp.

28.28.3.30 bool Serial::SerialImpl::setFlowcontrol (serial::flowcontrol_t *flowcontrol*)

Definition at line 1276 of file unix_serial.cpp.

28.28.3.31 bool Serial::SerialImpl::setParity (serial::parity_t *parity*)

Definition at line 1242 of file unix_serial.cpp.

28.28.3.32 void Serial::SerialImpl::setPort (const string & *port*)

Definition at line 1067 of file unix_serial.cpp.

28.28.3.33 bool Serial::SerialImpl::setRTS (bool *level*)

Definition at line 1371 of file unix_serial.cpp.

28.28.3.34 bool Serial::SerialImpl::setStandardBaudRate (speed_t *baudrate*)

Definition at line 1124 of file unix_serial.cpp.

28.28.3.35 bool Serial::SerialImpl::setStopbits (serial::stopbits_t *stopbits*)

Definition at line 1259 of file unix_serial.cpp.

28.28.3.36 bool Serial::SerialImpl::setTermios (const termios * *tio*)

Definition at line 1294 of file unix_serial.cpp.

28.28.3.37 void Serial::SerialImpl::setTimeout (serial::Timeout & *timeout*)

Definition at line 1075 of file unix_serial.cpp.

28.28.3.38 void Serial::SerialImpl::waitByteTimes (size_t *count*)

Definition at line 880 of file unix_serial.cpp.

28.28.3.39 bool Serial::SerialImpl::waitForChange ( )

Definition at line 1411 of file unix_serial.cpp.

28.28.3.40 int Serial::SerialImpl::waitfordata (size_t *data_count*, uint32_t *timeout*, size_t * *returned_size*)

Definition at line 798 of file unix_serial.cpp.

28.28.3.41 bool Serial::SerialImpl::waitReadable (uint32_t *timeout*)

Definition at line 765 of file unix_serial.cpp.

28.28.3.42 size_t Serial::SerialImpl::write (const uint8_t * *data*, size_t *length*)

Error

Timeout

Port ready to write

Definition at line 969 of file unix_serial.cpp.

28.28.3.43 int Serial::SerialImpl::writeLock ( )

Definition at line 1514 of file unix_serial.cpp.

28.28.3.44 int Serial::SerialImpl::writeUnlock ( )

Definition at line 1519 of file unix_serial.cpp.

The documentation for this class was generated from the following files:

- src/impl/unix/unix_serial.h
- src/impl/unix/unix_serial.cpp

28.29 termios2 Struct Reference

Public Attributes

- tcflag_t c_iflag
- tcflag_t c_oflag
- tcflag_t c_cflag
- tcflag_t c_lflag
- cc_t c_line
- cc_t c_cc [19]
- speed_t c_ispeed
- speed_t c_ospeed

28.29.1 Detailed Description

Definition at line 176 of file unix_serial.cpp.

28.29.2 Member Data Documentation

28.29.2.1 cc_t termios2::c_cc[19]

Definition at line 182 of file unix_serial.cpp.

28.29.2.2 tcflag_t termios2::c_cflag

Definition at line 179 of file unix_serial.cpp.

28.29.2.3 tcflag_t termios2::c_iflag

Definition at line 177 of file unix_serial.cpp.

28.29.2.4 speed_t termios2::c_ispeed

Definition at line 183 of file unix_serial.cpp.

28.29.2.5 tcflag_t termios2::c_lflag

Definition at line 180 of file unix_serial.cpp.

28.29.2.6 cc_t termios2::c_line

Definition at line 181 of file unix_serial.cpp.

28.29.2.7 tcflag_t termios2::c_oflag

Definition at line 178 of file unix_serial.cpp.

28.29.2.8 speed_t termios2::c_ospeed

Definition at line 184 of file unix_serial.cpp.

The documentation for this struct was generated from the following file:

- src/impl/unix/unix_serial.cpp

28.30 Thread Class Reference

`#include <thread.h>`

Public Member Functions

- Thread ()
- virtual ~Thread ()
- _size_t getHandle ()
- int terminate ()
- void * getParam ()
- int join (unsigned long timeout=-1)
- bool operator== (const Thread &right)

Static Public Member Functions

- template<class CLASS , int(CLASS::*)(void) PROC>
  static Thread ThreadCreateObjectFunctor (CLASS *pthis)
- template<class CLASS , int(CLASS::*)(void) PROC>
  static _size_t THREAD_PROC createThreadAux (void *param)
- static Thread createThread (thread_proc_t proc, void *param=NULL)

Protected Member Functions

- Thread (thread_proc_t proc, void *param)

Protected Attributes

- void * _param
- thread_proc_t _func
- _size_t _handle

28.30.1 Detailed Description

Definition at line 20 of file thread.h.

28.30.2 Constructor & Destructor Documentation

28.30.2.1 Thread::Thread ( ) [inline], [explicit]

Definition at line 57 of file thread.h.

28.30.2.2 virtual Thread::~Thread ( ) [inline], [virtual]

Definition at line 59 of file thread.h.

28.30.2.3 Thread::Thread (thread_proc_t *proc*, void * *param*) [inline], [explicit], [protected]

Definition at line 137 of file thread.h.

28.30.3 Member Function Documentation

28.30.3.1 static Thread Thread::createThread (thread_proc_t *proc*, void * *param* = NULL) [inline], [static]

Definition at line 35 of file thread.h.

28.30.3.2 template<class CLASS , int(CLASS::*)(void) PROC> static _size_t THREAD_PROC Thread::createThreadAux (void * *param*) [inline], [static]

Definition at line 30 of file thread.h.

28.30.3.3 _size_t Thread::getHandle ( ) [inline]

Definition at line 61 of file thread.h.

28.30.3.4 void* Thread::getParam ( ) [inline]

Definition at line 85 of file thread.h.

28.30.3.5 int Thread::join (unsigned long *timeout* = -1) [inline]

Definition at line 88 of file thread.h.

28.30.3.6 bool Thread::operator== (const Thread & *right*) [inline]

Definition at line 133 of file thread.h.

28.30.3.7 int Thread::terminate ( ) [inline]

Definition at line 64 of file thread.h.

28.30.3.8 template<class CLASS , int(CLASS::*)(void) PROC> static Thread Thread::ThreadCreateObjectFunctor (CLASS * *pthis*) [inline], [static]

Definition at line 24 of file thread.h.

28.30.4 Member Data Documentation

28.30.4.1 thread_proc_t Thread::_func [protected]

Definition at line 140 of file thread.h.

28.30.4.2 _size_t Thread::_handle [protected]

Definition at line 141 of file thread.h.

28.30.4.3 void* Thread::_param [protected]

Definition at line 139 of file thread.h.

The documentation for this class was generated from the following file:

- include/thread.h

28.31 serial::Timeout Struct Reference

`#include <serial.h>`

Public Member Functions

- Timeout (uint32_t inter_byte_timeout_=0, uint32_t read_timeout_constant_=0, uint32_t read_timeout_multiplier_=0, uint32_t write_timeout_constant_=0, uint32_t write_timeout_multiplier_=0)

Static Public Member Functions

- static uint32_t max ()
- static Timeout simpleTimeout (uint32_t timeout)

Public Attributes

- uint32_t inter_byte_timeout
- uint32_t read_timeout_constant
- uint32_t read_timeout_multiplier
- uint32_t write_timeout_constant
- uint32_t write_timeout_multiplier

28.31.1 Detailed Description

Structure for setting the timeout of the serial port, times are in milliseconds.

In order to disable the interbyte timeout, set it to Timeout::max().

Definition at line 58 of file serial.h.

28.31.2 Constructor & Destructor Documentation

28.31.2.1 serial::Timeout::Timeout (uint32_t *inter_byte_timeout_* = 0, uint32_t *read_timeout_constant_* = 0, uint32_t *read_timeout_multiplier_* = 0, uint32_t *write_timeout_constant_* = 0, uint32_t *write_timeout_multiplier_* = 0) [inline], [explicit]

Definition at line 91 of file serial.h.

28.31.3 Member Function Documentation

28.31.3.1 static uint32_t serial::Timeout::max ( ) [inline], [static]

Definition at line 62 of file serial.h.

28.31.3.2 static Timeout serial::Timeout::simpleTimeout (uint32_t *timeout*) [inline], [static]

Convenience function to generate Timeout structs using a single absolute timeout.

Parameters

timeout	A long that defines the time in milliseconds until a timeout occurs after a call to read or write is made.

Returns

Timeout struct that represents this simple timeout provided.

Definition at line 72 of file serial.h.

28.31.4 Member Data Documentation

28.31.4.1 uint32_t serial::Timeout::inter_byte_timeout

Number of milliseconds between bytes received to timeout on.

Definition at line 77 of file serial.h.

28.31.4.2 uint32_t serial::Timeout::read_timeout_constant

A constant number of milliseconds to wait after calling read.

Definition at line 79 of file serial.h.

28.31.4.3 uint32_t serial::Timeout::read_timeout_multiplier

A multiplier against the number of requested bytes to wait after calling read.

Definition at line 83 of file serial.h.

28.31.4.4 uint32_t serial::Timeout::write_timeout_constant

A constant number of milliseconds to wait after calling write.

Definition at line 85 of file serial.h.

28.31.4.5 uint32_t serial::Timeout::write_timeout_multiplier

A multiplier against the number of requested bytes to wait after calling write.

Definition at line 89 of file serial.h.

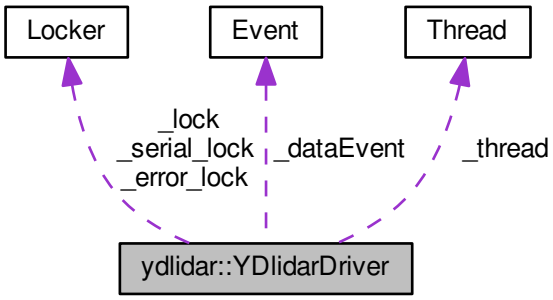
The documentation for this struct was generated from the following file:

- include/serial.h

28.32 ydlidar::YDlidarDriver Class Reference

`#include <ydlidar_driver.h>`

Collaboration diagram for ydlidar::YDlidarDriver:



Public Types

- enum { DEFAULT_TIMEOUT = 1000, MAX_SCAN_NODES = 2048, DEFAULT_TIMEOUT_COUNT = 2 }
- enum {
  YDLIDAR_F4 = 1, YDLIDAR_T1 = 2, YDLIDAR_F2 = 3, YDLIDAR_S4 = 4,
  YDLIDAR_G4 = 5, YDLIDAR_X4 = 6, YDLIDAR_G4PRO = 7, YDLIDAR_F4PRO = 8,
  YDLIDAR_R2 = 9, YDLIDAR_G10 = 10, YDLIDAR_S4B = 11, YDLIDAR_S2 = 12,
  YDLIDAR_G6 = 13, YDLIDAR_G2A = 14, YDLIDAR_G2B = 15, YDLIDAR_G2C = 16,
  YDLIDAR_G4B = 17, YDLIDAR_G4C = 18, YDLIDAR_G1 = 19, YDLIDAR_G5 = 20,
  YDLIDAR_G7 = 21, YDLIDAR_TG15 = 100, YDLIDAR_TG30 = 101, YDLIDAR_TG50 = 102,
  YDLIDAR_T15 = 200, YDLIDAR_Tail }

Public Member Functions

- YDlidarDriver ()
- virtual ~YDlidarDriver ()
- result_t connect (const char *port_path, uint32_t baudrate)

  Connecting Lidar
  After the connection if successful, you must use ::disconnect to close.
- void disconnect ()

  Disconnect the LiDAR.
- lidar_error_t getDriverError ()

  getDriverError
- result_t getHealth (device_health &health, uint32_t timeout=DEFAULT_TIMEOUT)

  get Health status
- result_t getDeviceInfo (device_info &info, uint32_t timeout=DEFAULT_TIMEOUT)

  get Device information

- result_t getScanFrequency (scan_frequency_t &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

  Get lidar scan frequency

  .
- result_t setScanFrequencyAdd (scan_frequency_t &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

  Increase the scanning frequency by 1.0 HZ

  .
- result_t setScanFrequencyDis (scan_frequency_t &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

  Reduce the scanning frequency by 1.0 HZ

  .
- result_t setScanFrequencyAddMic (scan_frequency_t &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

  Increase the scanning frequency by 0.1 HZ

  .
- result_t setScanFrequencyDisMic (scan_frequency_t &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

  Reduce the scanning frequency by 0.1 HZ

  .
- result_t getZeroOffsetAngle (offset_angle_t &angle, uint32_t timeout=DEFAULT_TIMEOUT)

  fetches zero angle tolerance values from lidar's internal memory while lidar assembly

- bool isConnected () const

  Is it connected to the lidar

  .
- bool isScanning () const

  Is the Lidar in the scan

  .
- uint32_t getPointIntervalTime () const

  getPointTime
- uint32_t getPackageTransferTime () const

  getPackageTime
- void setAutoReconnect (const bool &enable)

  whether to support hot plug

- void setSingleChannel (bool enable)

  setSingleChannel
- result_t startScan (uint32_t timeout=DEFAULT_TIMEOUT)

  Turn on scanning

  .
- result_t stopScan (uint32_t timeout=DEFAULT_TIMEOUT)

  stop Scanning state
- result_t stop ()

  turn off scanning

- result_t grabScanData (LaserFan *fan, uint32_t timeout=DEFAULT_TIMEOUT)

  Get a circle of laser data

  .
- result_t startMotor ()

  start motor

- result_t stopMotor ()

  stop motor

- void flush ()

  flush

Static Public Member Functions

- static std::string getSDKVersion ()

  Get SDK Version
  static function.
- static std::map< std::string, std::string > lidarPortList ()

  lidarPortList Get Lidar Port lists

Public Attributes

- bool m_isConnected

  LiDAR connected state.
- bool m_isScanning

  LiDAR Scanning state.
- bool isAutoReconnect

  auto reconnect
- bool isAutoconnting

  auto connecting state
- Event _dataEvent

  data event
- Locker _lock

  thread lock
- Locker _serial_lock

  serial lock
- Locker _error_lock

  error lock
- Thread _thread

  thread id

Protected Member Functions

- result_t createThread ()

  Data parsing thread
  .
- result_t startAutoScan (uint32_t timeout=DEFAULT_TIMEOUT)

  Automatically reconnect the lidar
  .
- result_t waitPackage (LaserFan &package, uint32_t timeout=DEFAULT_TIMEOUT)

  Unpacking
  .
- result_t waitScanData (LaserFan &package, uint32_t timeout=DEFAULT_TIMEOUT)

  get unpacked data
- int cacheScanData ()

  data parsing thread
- result_t sendCommand (uint8_t cmd, const void *payload=NULL, size_t payloadsize=0)

  send data to lidar
- result_t waitForData (size_t data_count, uint32_t timeout=DEFAULT_TIMEOUT, size_t *returned_size=NULL)

Waiting for the specified size data from the lidar
.

- result_t getData (uint8_t *data, size_t size)
  get data from serial
- result_t sendData (const uint8_t *data, size_t size)
  send data to serial
- void disableDataGrabbing ()
  disable Data scan channel
- void setDTR ()
  set DTR
- void clearDTR ()
  clear DTR
- void flushSerial ()
  flushSerial
- void setDriverError (const lidar_error_t &er)
  setDriverError

28.32.1 Detailed Description

Definition at line 60 of file ydlidar_driver.h.

28.32.2 Member Enumeration Documentation

28.32.2.1 anonymous enum

Enumerator

DEFAULT_TIMEOUT Default timeout.
MAX_SCAN_NODES Default Max Scan Count.
DEFAULT_TIMEOUT_COUNT

Definition at line 436 of file ydlidar_driver.h.

28.32.2.2 anonymous enum

Enumerator

YDLIDAR_F4 F4 LiDAR Model.
YDLIDAR_T1 T1 LiDAR Model.
YDLIDAR_F2 F2 LiDAR Model.
YDLIDAR_S4 S4 LiDAR Model.
YDLIDAR_G4 G4 LiDAR Model.
YDLIDAR_X4 X4 LiDAR Model.
YDLIDAR_G4PRO G4PRO LiDAR Model.

YDLIDAR_F4PRO F4PRO LiDAR Model.
YDLIDAR_R2 R2 LiDAR Model.
YDLIDAR_G10 G10 LiDAR Model.
YDLIDAR_S4B S4B LiDAR Model.
YDLIDAR_S2 S2 LiDAR Model.
YDLIDAR_G6 G6 LiDAR Model.
YDLIDAR_G2A G2A LiDAR Model.
YDLIDAR_G2B G2 LiDAR Model.
YDLIDAR_G2C G2C LiDAR Model.
YDLIDAR_G4B G4B LiDAR Model.
YDLIDAR_G4C G4C LiDAR Model.
YDLIDAR_G1 G1 LiDAR Model.
YDLIDAR_G5 G5 LiDAR Model.
YDLIDAR_G7 G7 LiDAR Model.
YDLIDAR_TG15 TG15 LiDAR Model.
YDLIDAR_TG30 T30 LiDAR Model.
YDLIDAR_TG50 TG50 LiDAR Model.
YDLIDAR_T15 T15 LiDAR Model.
YDLIDAR_Tail

Definition at line 441 of file ydlidar_driver.h.

28.32.3 Constructor & Destructor Documentation

28.32.3.1 ydlidar::YDlidarDriver::YDlidarDriver ( )

A constructor. A more elaborate description of the constructor.

Definition at line 64 of file ydlidar_driver.cpp.

28.32.3.2 ydlidar::YDlidarDriver::~YDlidarDriver ( ) [virtual]

A destructor. A more elaborate description of the destructor.

Definition at line 82 of file ydlidar_driver.cpp.

28.32.4 Member Function Documentation

28.32.4.1 int ydlidar::YDlidarDriver::cacheScanData ( ) [protected]

data parsing thread

Definition at line 638 of file ydlidar_driver.cpp.

28.32.4.2 void ydlidar::YDlidarDriver::clearDTR ( ) [protected]

clear DTR

Definition at line 140 of file ydlidar_driver.cpp.

28.32.4.3 result_t ydlidar::YDlidarDriver::connect (const char * *port_path*, uint32_t *baudrate*)

Connecting Lidar
After the connection if successful, you must use ::disconnect to close.

Parameters

in	*port_path*	serial port
in	*baudrate*	serial baudrate，S2-Pro：115200

Returns

connection status

Return values

0	success
<	0 failed

Note

After the connection if successful, you must use ::disconnect to close

See also

function ::YDlidarDriver::disconnect ()

Definition at line 104 of file ydlidar_driver.cpp.

28.32.4.4 result_t ydlidar::YDlidarDriver::createThread ( ) [protected]

Data parsing thread
.

Note

Before you create a dta parsing thread, you must use the ::startScan function to start the lidar scan successfully.

Definition at line 918 of file ydlidar_driver.cpp.

28.32.4.5 void ydlidar::YDlidarDriver::disableDataGrabbing ( ) [protected]

disable Data scan channel

Definition at line 238 of file ydlidar_driver.cpp.

28.32.4.6 void ydlidar::YDlidarDriver::disconnect ( )

Disconnect the LiDAR.

Definition at line 216 of file ydlidar_driver.cpp.

28.32.4.7 void ydlidar::YDlidarDriver::flush ( )

flush

Definition at line 204 of file ydlidar_driver.cpp.

28.32.4.8 void ydlidar::YDlidarDriver::flushSerial ( ) [protected]

flushSerial

Definition at line 150 of file ydlidar_driver.cpp.

28.32.4.9 result_t ydlidar::YDlidarDriver::getData (uint8_t * *data*, size_t *size*) [protected]

get data from serial

Parameters

in	*data*	data
in	*size*	date size

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 606 of file ydlidar_driver.cpp.

28.32.4.10 result_t ydlidar::YDlidarDriver::getDeviceInfo (device_info & *info*, uint32_t *timeout* = DEFAULT_TIMEOUT)

get Device information

Parameters

in	*info*	Device information
in	*timeout*	timeout

Returns

result status

Return values

RESULT_OK	success
RESULT_FAILE	or RESULT_TIMEOUT failed

Definition at line 303 of file ydlidar_driver.cpp.

28.32.4.11 lidar_error_t ydlidar::YDlidarDriver::getDriverError ( )

getDriverError

Returns

Definition at line 268 of file ydlidar_driver.cpp.

28.32.4.12 result_t ydlidar::YDlidarDriver::getHealth (device_health & *health*, uint32_t *timeout* = DEFAULT_TIMEOUT)

get Health status

Returns

result status

Return values

RESULT_OK	success
RESULT_FAILE	or RESULT_TIMEOUT failed

Definition at line 273 of file ydlidar_driver.cpp.

28.32.4.13 uint32_t ydlidar::YDlidarDriver::getPackageTransferTime ( ) const

getPackageTime

Returns

Definition at line 264 of file ydlidar_driver.cpp.

28.32.4.14 uint32_t ydlidar::YDlidarDriver::getPointIntervalTime ( ) const

getPointTime

Returns

Definition at line 260 of file ydlidar_driver.cpp.

28.32.4.15 result_t ydlidar::YDlidarDriver::getScanFrequency (scan_frequency_t & *frequency*, uint32_t *timeout* = DEFAULT_TIMEOUT)

Get lidar scan frequency
.

Parameters

in	*frequency*	scanning frequency
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Non-scan state, perform currect operation.

Definition at line 336 of file ydlidar_driver.cpp.

28.32.4.16 std::string ydlidar::YDlidarDriver::getSDKVersion ( ) [static]

Get SDK Version
static function.

Returns

Version

Definition at line 972 of file ydlidar_driver.cpp.

28.32.4.17 result_t ydlidar::YDlidarDriver::getZeroOffsetAngle (offset_angle_t & *angle,* uint32_t *timeout =* DEFAULT_TIMEOUT)

fetches zero angle tolerance values from lidar's internal memory while lidar assembly

Parameters

in	*angle*	zero offset angle
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_TIMEOUT	Failed
RESULT_FAILE	Angle is not calibrated

Note

Non-scan state, perform currect operation.

Definition at line 511 of file ydlidar_driver.cpp.

28.32.4.18 result_t ydlidar::YDlidarDriver::grabScanData (LaserFan * *fan,* uint32_t *timeout =* DEFAULT_TIMEOUT)

Get a circle of laser data
.

Parameters

in	*fan*	Laser data
in	*count*	one circle of laser points
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Before starting, you must start the start the scan successfully with the ::startScan function

Definition at line 822 of file ydlidar_driver.cpp.

28.32.4.19 bool ydlidar::YDlidarDriver::isConnected ( ) const

Is it connected to the lidar
.

Returns

connection status

Return values

true	connected
false	Non-connected

Definition at line 252 of file ydlidar_driver.cpp.

28.32.4.20 bool ydlidar::YDlidarDriver::isScanning ( ) const

Is the Lidar in the scan
.

Returns

scanning status

Return values

true	scanning
false	non-scanning

Definition at line 256 of file ydlidar_driver.cpp.

28.32.4.21 std::map< std::string, std::string > ydlidar::YDlidarDriver::lidarPortList ( ) [static]

lidarPortList Get Lidar Port lists

Returns

online lidars

Definition at line 976 of file ydlidar_driver.cpp.

28.32.4.22 result_t ydlidar::YDlidarDriver::sendCommand (uint8_t *cmd*, const void * *payload* = NULL, size_t *payloadsize* = 0) [protected]

send data to lidar

Parameters

in	*cmd*	command code
in	*payload*	payload
in	*payloadsize*	payloadsize

Returns

result status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 544 of file ydlidar_driver.cpp.

28.32.4.23 result_t ydlidar::YDlidarDriver::sendData (const uint8_t * *data*, size_t *size*) [protected]

send data to serial

Parameters

in	*data*	data
in	*size*	data size

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 582 of file ydlidar_driver.cpp.

28.32.4.24 void ydlidar::YDlidarDriver::setAutoReconnect (const bool & *enable*)

whether to support hot plug

设置雷达异常自动重新连接

Parameters

in	*enable*	hot plug : true support false no support
in	*enable*	是否开启自动重连: true 开启 false 关闭

Definition at line 852 of file ydlidar_driver.cpp.

28.32.4.25 void ydlidar::YDlidarDriver::setDriverError (const lidar_error_t & *er*) [protected]

setDriverError

Parameters

er	

Definition at line 163 of file ydlidar_driver.cpp.

28.32.4.26 void ydlidar::YDlidarDriver::setDTR ( ) [protected]

set DTR

Definition at line 129 of file ydlidar_driver.cpp.

28.32.4.27 result_t ydlidar::YDlidarDriver::setScanFrequencyAdd (scan_frequency_t & *frequency,* uint32_t *timeout* = DEFAULT_TIMEOUT)

Increase the scanning frequency by 1.0 HZ
.

Parameters

in	*frequency*	scanning frequency
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Non-scan state, perform currect operation.

Definition at line 371 of file ydlidar_driver.cpp.

28.32.4.28 result_t ydlidar::YDlidarDriver::setScanFrequencyAddMic (scan_frequency_t & *frequency,* uint32_t *timeout* = DEFAULT_TIMEOUT)

Increase the scanning frequency by 0.1 HZ
.

Parameters

in	*frequency*	scanning frequency
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Non-scan state, perform currect operation.

Definition at line 441 of file ydlidar_driver.cpp.

28.32.4.29 result_t ydlidar::YDlidarDriver::setScanFrequencyDis (scan_frequency_t & *frequency,* uint32_t *timeout* = DEFAULT_TIMEOUT)

Reduce the scanning frequency by 1.0 HZ
.

Parameters

in	*frequency*	scanning frequency
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Non-scan state, perform currect operation.

Definition at line 406 of file ydlidar_driver.cpp.

28.32.4.30 result_t ydlidar::YDlidarDriver::setScanFrequencyDisMic (scan_frequency_t & *frequency,* uint32_t *timeout* = DEFAULT_TIMEOUT)

Reduce the scanning frequency by 0.1 HZ
.

Parameters

in	*frequency*	scanning frequency
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Non-scan state, perform currect operation.

Definition at line 476 of file ydlidar_driver.cpp.

28.32.4.31 void ydlidar::YDlidarDriver::setSingleChannel (bool *enable*)

setSingleChannel

Parameters

enable	

Definition at line 856 of file ydlidar_driver.cpp.

28.32.4.32 result_t ydlidar::YDlidarDriver::startAutoScan (uint32_t *timeout* = DEFAULT_TIMEOUT) [protected]

Automatically reconnect the lidar
.

Parameters

in	*force*	scan model
in	*timeout*	timeout

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Lidar abnormality automatically reconnects.

Definition at line 931 of file ydlidar_driver.cpp.

28.32.4.33 result_t ydlidar::YDlidarDriver::startMotor ( )

start motor

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 168 of file ydlidar_driver.cpp.

28.32.4.34 result_t ydlidar::YDlidarDriver::startScan (uint32_t *timeout* = DEFAULT_TIMEOUT)

Turn on scanning
.

Parameters

in	*force*	Scan mode
in	*timeout*	timeout

Returns

result status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Note

Just turn it on once

Definition at line 863 of file ydlidar_driver.cpp.

28.32.4.35 result_t ydlidar::YDlidarDriver::stop ( )

turn off scanning

Returns

result status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 960 of file ydlidar_driver.cpp.

28.32.4.36 result_t ydlidar::YDlidarDriver::stopMotor ( )

stop motor

Returns

return status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 186 of file ydlidar_driver.cpp.

28.32.4.37 result_t ydlidar::YDlidarDriver::stopScan (uint32_t *timeout* = DEFAULT_TIMEOUT)

stop Scanning state

Parameters

timeout	timeout

Returns

status

Return values

RESULT_OK	success
RESULT_FAILE	failed

Definition at line 903 of file ydlidar_driver.cpp.

28.32.4.38 result_t ydlidar::YDlidarDriver::waitForData (size_t *data_count*, uint32_t *timeout* = DEFAULT_TIMEOUT, size_t * *returned_size* = NULL) [protected]

Waiting for the specified size data from the lidar
.

Parameters

in	*data_count*	wait max data size
in	*timeout*	timeout
in	*returned_size*	really data size

Returns

return status

Return values

RESULT_OK	success
RESULT_TIMEOUT	wait timeout
RESULT_FAILE	failed

Note

when timeout = -1, it will block...

Definition at line 627 of file ydlidar_driver.cpp.

28.32.4.39 result_t ydlidar::YDlidarDriver::waitPackage (LaserFan & *package*, uint32_t *timeout* = DEFAULT_TIMEOUT) [protected]

Unpacking
.

Parameters

in	*package*	lidar point information
in	*timeout*	timeout

Definition at line 783 of file ydlidar_driver.cpp.

28.32.4.40 result_t ydlidar::YDlidarDriver::waitScanData (LaserFan & *package*, uint32_t *timeout* = DEFAULT_TIMEOUT) [protected]

get unpacked data

Parameters

in	*package*	laser node
in	*count*	lidar points size
in	*timeout*	timeout

Returns

result status

Return values

RESULT_OK	success
RESULT_TIMEOUT	timeout
RESULT_FAILE	failed

Definition at line 813 of file ydlidar_driver.cpp.

28.32.5 Member Data Documentation

28.32.5.1 Event ydlidar::YDlidarDriver::_dataEvent

data event

Definition at line 471 of file ydlidar_driver.h.

28.32.5.2 Locker ydlidar::YDlidarDriver::_error_lock

error lock

Definition at line 474 of file ydlidar_driver.h.

28.32.5.3 Locker ydlidar::YDlidarDriver::_lock

thread lock

Definition at line 472 of file ydlidar_driver.h.

28.32.5.4 Locker ydlidar::YDlidarDriver::_serial_lock

serial lock

Definition at line 473 of file ydlidar_driver.h.

28.32.5.5 Thread ydlidar::YDlidarDriver::_thread

thread id

Definition at line 475 of file ydlidar_driver.h.

28.32.5.6 bool ydlidar::YDlidarDriver::isAutoconnnting

auto connecting state

Definition at line 434 of file ydlidar_driver.h.

28.32.5.7 bool ydlidar::YDlidarDriver::isAutoReconnect

auto reconnect

Definition at line 432 of file ydlidar_driver.h.

28.32.5.8 bool ydlidar::YDlidarDriver::m_isConnected

LiDAR connected state.

Definition at line 428 of file ydlidar_driver.h.

28.32.5.9 bool ydlidar::YDlidarDriver::m_isScanning

LiDAR Scanning state.

Definition at line 430 of file ydlidar_driver.h.

The documentation for this class was generated from the following files:

- include/ydlidar_driver.h
- src/ydlidar_driver.cpp

Chapter 29

File Documentation

29.13 doc/FAQs/Software_FAQs_cn.md File Reference

29.14 doc/howto/how_to_build_and_debug_using_vscode.md File Reference

29.15 doc/howto/how_to_build_and_install.md File Reference

29.16 doc/howto/how_to_create_a_pull.md File Reference

29.17 doc/howto/how_to_create_a_udev_rules.md File Reference

29.18 doc/howto/how_to_gerenrate_vs_project_by_cmake.md File Reference

29.19 doc/howto/how_to_solve_slow_pull_from_cn.md File Reference

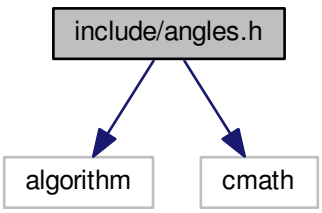
29.20 doc/quickstart/s2_pro_software_installation_guide.md File Reference

29.21 doc/S2_Pro_SDK_API_for_Developers.md File Reference

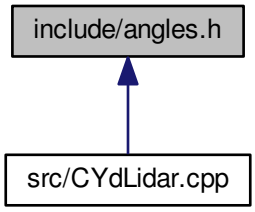
29.22 include/angles.h File Reference

```
#include <algorithm>
#include <cmath>
```

Include dependency graph for angles.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- angles

Functions

- static double angles::from_degrees (double degrees)

  Convert degrees to radians.
- static double angles::to_degrees (double radians)

  Convert radians to degrees.
- static double angles::normalize_angle_positive (double angle)

  normalize_angle_positive
- static double angles::normalize_angle (double angle)

  normalize
- static double angles::shortest_angular_distance (double from, double to)

  shortest_angular_distance
- static double angles::two_pi_complement (double angle)

  returns the angle in [-2∗M_PI, 2∗M_PI] going the other way along the unit circle.
- static bool angles::find_min_max_delta (double from, double left_limit, double right_limit, double &result_↩ min_delta, double &result_max_delta)

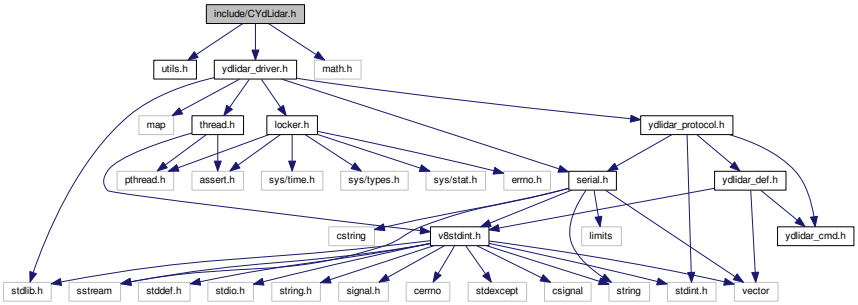
  This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".
- static bool angles::shortest_angular_distance_with_limits (double from, double to, double left_limit, double right_limit, double &shortest_angle)

  Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by left_limit and right_limit. The valid interval of angular positions is [left_limit,right_limit]. E.g., [-0.25,0.25] is a 0.5 radians wide interval that contains 0. But [0.25,-0.25] is a 2∗M_PI-0.5 wide interval that contains M_PI (but not 0). The value of shortest_angle is the angular difference between "from" and "to" that lies within the defined valid interval. E.↩ g. shortest_angular_distance_with_limits(-0.5,0.5,0.25,-0.25,ss) evaluates ss to 2∗M_PI-1.0 and returns true while shortest_angular_distance_with_limits(-0.5,0.5,-0.25,0.25,ss) returns false since -0.5 and 0.5 do not lie in the interval [-0.25,0.25].

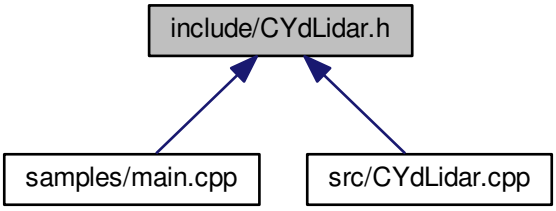
29.23 include/CYdLidar.h File Reference

```
#include "utils.h"
#include "ydlidar_driver.h"
#include <math.h>
```

Include dependency graph for CYdLidar.h:



This graph shows which files directly or indirectly include this file:



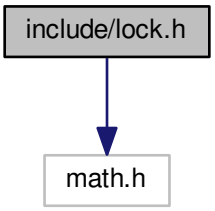
Classes

- class CYdLidar

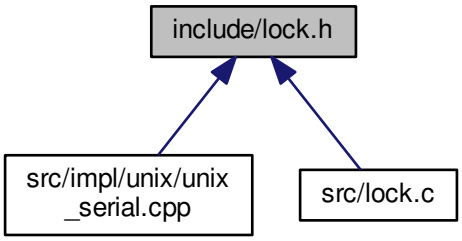
29.24 include/lock.h File Reference

```
#include <math.h>
```

Include dependency graph for lock.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define LOCK system_does_not_lock
- #define UNLOCK system_does_not_unlock

Functions

- int check_group_uucp ()
- int check_lock_pid (const char *file, int openpid)
- int lock_device (const char *)
- void unlock_device (const char *)
- int is_device_locked (const char *)
- int check_lock_status (const char *)
- int lfs_unlock (const char *, int)
- int lfs_lock (const char *, int)
- int lib_lock_dev_unlock (const char *, int)
- int lib_lock_dev_lock (const char *, int)
- void fhs_unlock (const char *, int)
- int fhs_lock (const char *, int)
- void uucp_unlock (const char *, int)
- int uucp_lock (const char *, int)

29.24.1 Macro Definition Documentation

29.24.1.1 #define LOCK system_does_not_lock

Definition at line 206 of file lock.h.

29.24.1.2 #define UNLOCK system_does_not_unlock

Definition at line 207 of file lock.h.

29.24.2 Function Documentation

29.24.2.1 int check_group_uucp ( )

Definition at line 501 of file lock.c.

29.24.2.2 int check_lock_pid (const char * *file*, int *openpid*)

Definition at line 448 of file lock.c.

29.24.2.3 int check_lock_status (const char *)

Definition at line 340 of file lock.c.

29.24.2.4 int fhs_lock (const char *, int)

Definition at line 198 of file lock.c.

29.24.2.5 void fhs_unlock (const char *, int)

Definition at line 377 of file lock.c.

29.24.2.6 int is_device_locked (const char *)

Definition at line 648 of file lock.c.

29.24.2.7 int lfs_lock (const char *, int)

29.24.2.8 int lfs_unlock (const char *, int)

29.24.2.9 int lib_lock_dev_lock (const char *, int)

29.24.2.10 int lib_lock_dev_unlock (const char *, int)

29.24.2.11 int lock_device (const char *)

29.24.2.12 void unlock_device (const char *)

29.24.2.13 int uucp_lock (const char *, int)

Definition at line 280 of file lock.c.

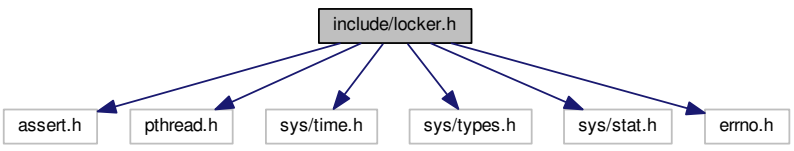
29.24.2.14 void uucp_unlock (const char *, int)

Definition at line 407 of file lock.c.

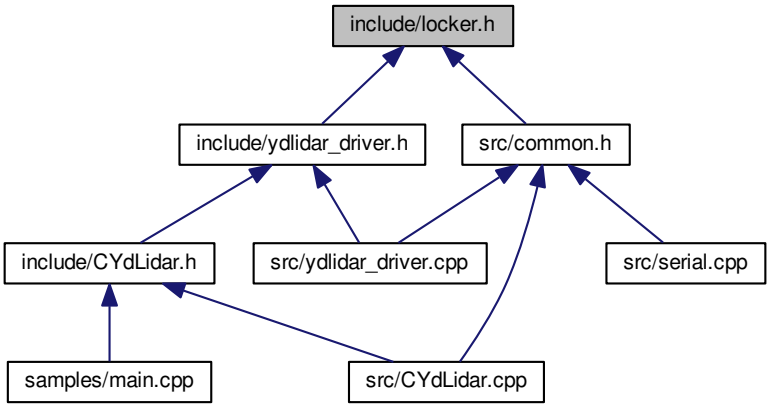
29.25 include/locker.h File Reference

```
#include <assert.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
```

Include dependency graph for locker.h:



This graph shows which files directly or indirectly include this file:



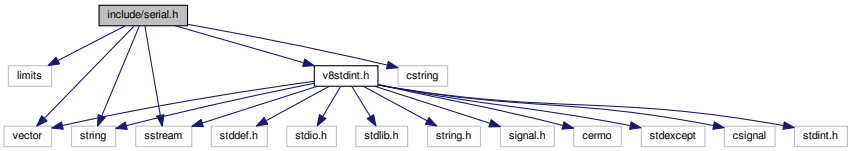
Classes

- class Locker
- class Event
- class ScopedLocker

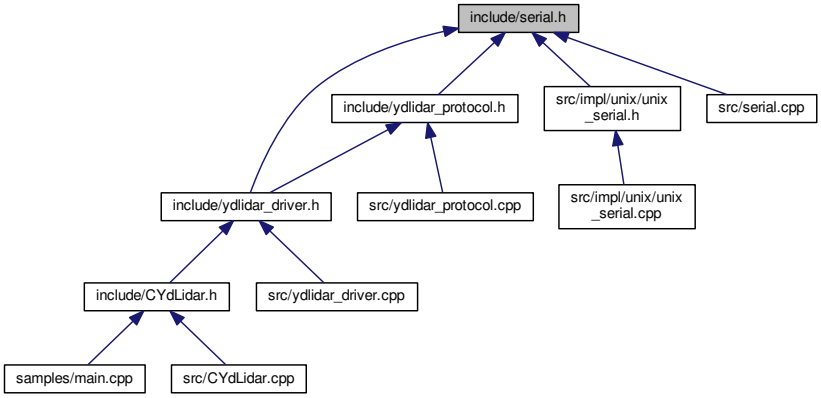
29.26 include/serial.h File Reference

```
#include <limits>
#include <vector>
#include <string>
#include <cstring>
#include <sstream>
#include "v8stdint.h"
```

Include dependency graph for serial.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct serial::Timeout
- class serial::Serial
- struct serial::PortInfo

Namespaces

- serial

Enumerations

- enum serial::bytesize_t { serial::fivebits = 5, serial::sixbits = 6, serial::sevenbits = 7, serial::eightbits = 8 }
- enum serial::parity_t {
  serial::parity_none = 0, serial::parity_odd = 1, serial::parity_even = 2, serial::parity_mark = 3,
  serial::parity_space = 4 }
- enum serial::stopbits_t { serial::stopbits_one = 1, serial::stopbits_two = 2, serial::stopbits_one_point_five }
- enum serial::flowcontrol_t { serial::flowcontrol_none = 0, serial::flowcontrol_software, serial::flowcontrol_↩
  hardware }

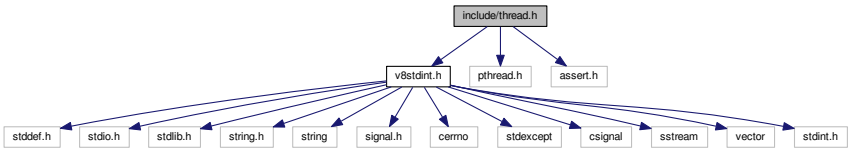
Functions

- std::vector< PortInfo > serial::list_ports ()

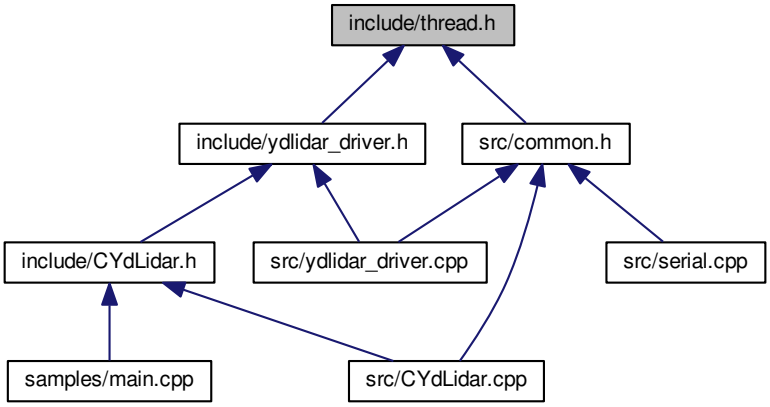
29.27 include/thread.h File Reference

```
#include "v8stdint.h"
#include <pthread.h>
#include <assert.h>
```

Include dependency graph for thread.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Thread

Macros

- #define CLASS_THREAD(c, x) Thread::ThreadCreateObjectFunctor<c, &c::x>(this)

29.27.1 Macro Definition Documentation

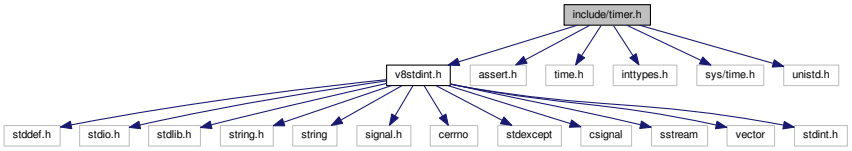
29.27.1.1 #define CLASS_THREAD(*c*, *x*) Thread::ThreadCreateObjectFunctor<c, &c::x>(this)

Definition at line 18 of file thread.h.

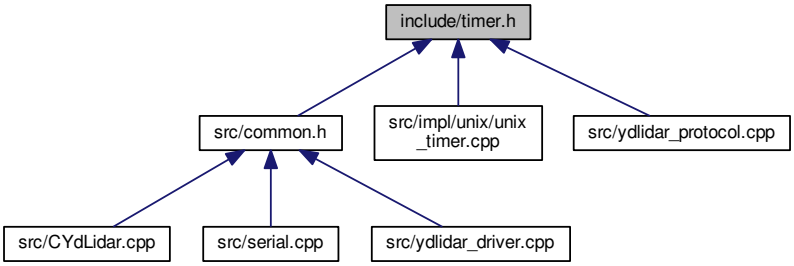
29.28 include/timer.h File Reference

```
#include "v8stdint.h"
#include <assert.h>
#include <time.h>
#include <inttypes.h>
#include <sys/time.h>
#include <unistd.h>
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- impl

Macros

- #define BEGIN_STATIC_CODE(_blockname_)
- #define END_STATIC_CODE(_blockname_) } _instance_##_blockname_;
- #define getms() impl::getHDTimer()
- #define getTime() impl::getCurrentTime()

Functions

- static void delay (uint32_t ms)
- uint32_t impl::getHDTimer ()
- uint64_t impl::getCurrentTime ()

29.28.1 Macro Definition Documentation

29.28.1.1 #define BEGIN_STATIC_CODE(_blockname_)

Value:

```
static class _static_code_##_blockname_ {    \
    public:     \
    _static_code_##_blockname_ ()
```

Definition at line 9 of file timer.h.

29.28.1.2 #define END_STATIC_CODE(_blockname_) } _instance_##_blockname_;

Definition at line 15 of file timer.h.

29.28.1.3 #define getms( ) impl::getHDTimer()

Definition at line 49 of file timer.h.

29.28.1.4 #define getTime( ) impl::getCurrentTime()

Definition at line 50 of file timer.h.

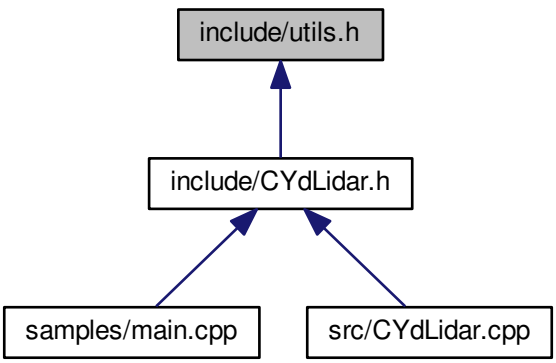
29.28.2 Function Documentation

29.28.2.1 static void delay (uint32_t *ms*) [inline], [static]

Definition at line 26 of file timer.h.

29.29 include/utils.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define YDLIDAR_API

29.29.1 Macro Definition Documentation

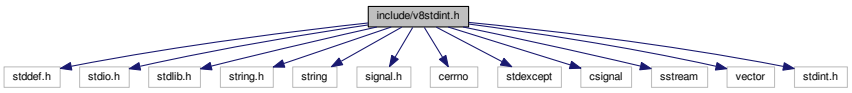
29.29.1.1 #define YDLIDAR_API

Definition at line 17 of file utils.h.

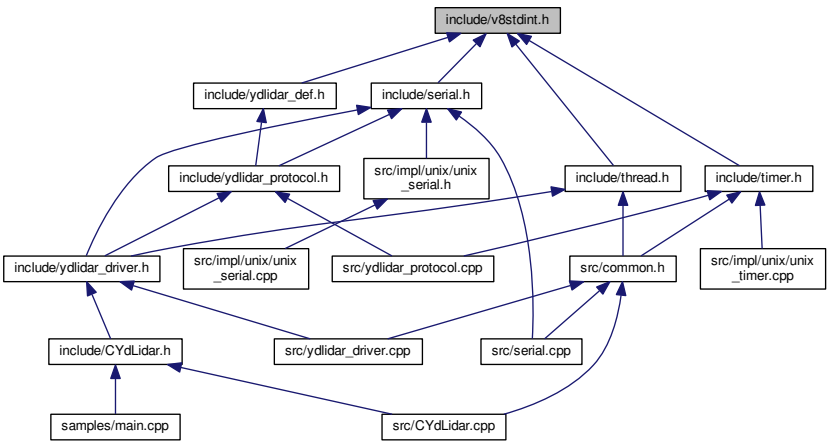
29.30 include/v8stdint.h File Reference

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <signal.h>
#include <cerrno>
#include <stdexcept>
#include <csignal>
#include <sstream>
#include <vector>
#include <stdint.h>
```

Include dependency graph for v8stdint.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- ydlidar

Macros

- #define UNUSED(x) (void)x
- #define __small_endian
- #define __attribute__(x)
- #define RESULT_OK 0
- #define RESULT_TIMEOUT -1
- #define RESULT_FAIL -2
- #define IS_OK(x) ((x) == RESULT_OK)
- #define IS_TIMEOUT(x) ((x) == RESULT_TIMEOUT)
- #define IS_FAIL(x) ((x) == RESULT_FAIL)
- #define M_PI 3.1415926

Typedefs

- typedef _size_t(THREAD_PROC * thread_proc_t) (void *)
- typedef int32_t result_t
- typedef void(* signal_handler_t) (int)

Functions

- signal_handler_t set_signal_handler (int signal_value, signal_handler_t signal_handler)
- void trigger_interrupt_guard_condition (int signal_value)
- void signal_handler (int signal_value)
- void ydlidar::init (int argc, char *argv[])
- bool ydlidar::ok ()
- void ydlidar::shutdownNow ()
- std::vector< float > ydlidar::split (const std::string &s, char delim)

  split string to vector by delim format

Variables

- static volatile sig_atomic_t g_signal_status = 0
- static signal_handler_t old_signal_handler = 0

29.30.1 Macro Definition Documentation

29.30.1.1 #define __attribute__(*x*)

Definition at line 37 of file v8stdint.h.

29.30.1.2 #define __small_endian

Definition at line 34 of file v8stdint.h.

29.30.1.3 #define IS_FAIL(x) ((x) == RESULT_FAIL)

Definition at line 72 of file v8stdint.h.

29.30.1.4 #define IS_OK(x) ((x) == RESULT_OK)

Definition at line 70 of file v8stdint.h.

29.30.1.5 #define IS_TIMEOUT(x) ((x) == RESULT_TIMEOUT)

Definition at line 71 of file v8stdint.h.

29.30.1.6 #define M_PI 3.1415926

Definition at line 76 of file v8stdint.h.

29.30.1.7 #define RESULT_FAIL -2

Definition at line 67 of file v8stdint.h.

29.30.1.8 #define RESULT_OK 0

Definition at line 65 of file v8stdint.h.

29.30.1.9 #define RESULT_TIMEOUT -1

Definition at line 66 of file v8stdint.h.

29.30.1.10 #define UNUSED(x) (void)x

Definition at line 17 of file v8stdint.h.

29.30.2 Typedef Documentation

29.30.2.1 typedef int32_t result_t

Definition at line 63 of file v8stdint.h.

29.30.2.2 typedef void(* signal_handler_t) (int)

Definition at line 91 of file v8stdint.h.

29.30.2.3 typedef _size_t(THREAD_PROC * thread_proc_t) (void *)

Definition at line 61 of file v8stdint.h.

29.30.3 Function Documentation

29.30.3.1 signal_handler_t set_signal_handler (int *signal_value,* signal_handler_t *signal_handler*) [inline]

Definition at line 100 of file v8stdint.h.

29.30.3.2 void signal_handler (int *signal_value*) [inline]

Definition at line 164 of file v8stdint.h.

29.30.3.3 void trigger_interrupt_guard_condition (int *signal_value*) [inline]

Definition at line 155 of file v8stdint.h.

29.30.4 Variable Documentation

29.30.4.1 volatile sig_atomic_t g_signal_status = 0 [static]

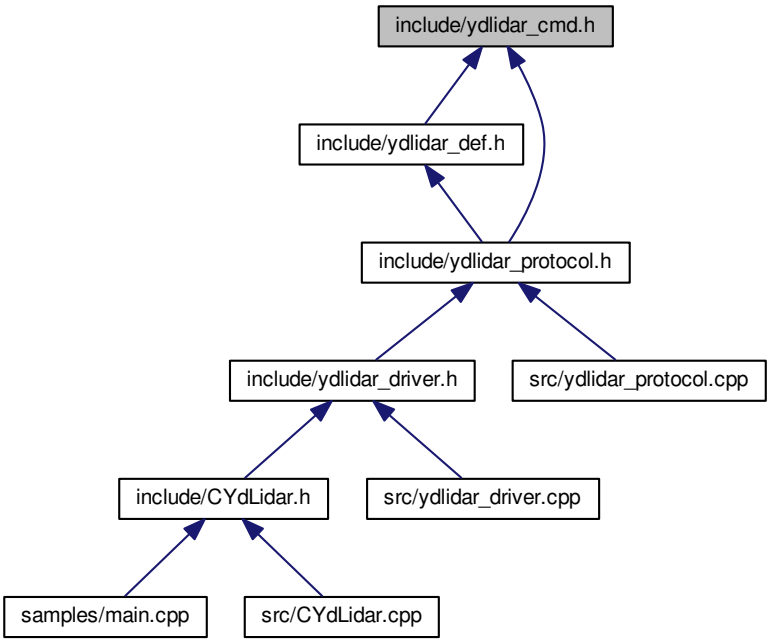
Definition at line 86 of file v8stdint.h.

29.30.4.2 signal_handler_t old_signal_handler = 0 [static]

Definition at line 92 of file v8stdint.h.

29.31 include/ydlidar_cmd.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define LIDAR_CMD_STOP 0x65
- #define LIDAR_CMD_SCAN 0x60
- #define LIDAR_CMD_FORCE_SCAN 0x61
- #define LIDAR_CMD_RESET 0x80
- #define LIDAR_CMD_FORCE_STOP 0x00
- #define LIDAR_CMD_GET_EAI 0x55
- #define LIDAR_CMD_GET_DEVICE_INFO 0x90
- #define LIDAR_CMD_GET_DEVICE_HEALTH 0x92
- #define LIDAR_ANS_TYPE_DEVINFO 0x4
- #define LIDAR_ANS_TYPE_DEVHEALTH 0x6
- #define LIDAR_CMD_SYNC_BYTE 0xA5
- #define LIDAR_CMDFLAG_HAS_PAYLOAD 0x80
- #define LIDAR_ANS_SYNC_BYTE1 0xA5
- #define LIDAR_ANS_SYNC_BYTE2 0x5A
- #define LIDAR_ANS_TYPE_MEASUREMENT 0x81
- #define LIDAR_RESP_MEASUREMENT_SYNCBIT (0x1<<0)
- #define LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT 2
- #define LIDAR_RESP_MEASUREMENT_CHECKBIT (0x1<<0)
- #define LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT 1
- #define LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT 2
- #define LIDAR_CMD_RUN_POSITIVE 0x06

- #define LIDAR_CMD_RUN_INVERSION 0x07
- #define LIDAR_CMD_SET_AIMSPEED_ADDMIC 0x09
- #define LIDAR_CMD_SET_AIMSPEED_DISMIC 0x0A
- #define LIDAR_CMD_SET_AIMSPEED_ADD 0x0B
- #define LIDAR_CMD_SET_AIMSPEED_DIS 0x0C
- #define LIDAR_CMD_GET_AIMSPEED 0x0D
- #define LIDAR_CMD_SET_SAMPLING_RATE 0xD0
- #define LIDAR_CMD_GET_SAMPLING_RATE 0xD1
- #define LIDAR_STATUS_OK 0x0
- #define LIDAR_STATUS_WARNING 0x1
- #define LIDAR_STATUS_ERROR 0x2
- #define LIDAR_CMD_GET_OFFSET_ANGLE 0x93
- #define Node_Default_Quality (10<<2)
- #define Node_Sync 0x01
- #define Node_NotSync 0x00
- #define PackagePaidBytes 10
- #define PH 0x55AA
- #define HEADER_MSB 0xAA

  Packet Header MSB.
- #define HEADER_LSB 0x55

  Packet Header LSB.
- #define FREINDEX 0
- #define HEALTHINDEX 3
- #define READ_DEFAULT_TIMEOUT 1000
- #define INFO_DEFAULT_TIMEOUT 20
- #define SCAN_DEFAULT_TIMEOUT 100

Enumerations

- enum CT { CT_Normal = 0, CT_RingStart = 1, CT_Tail }

29.31.1 Macro Definition Documentation

29.31.1.1 #define FREINDEX 0

Definition at line 79 of file ydlidar_cmd.h.

29.31.1.2 #define HEADER_LSB 0x55

Packet Header LSB.

Definition at line 77 of file ydlidar_cmd.h.

29.31.1.3 #define HEADER_MSB 0xAA

Packet Header MSB.

Definition at line 75 of file ydlidar_cmd.h.

29.31.1.4 #define HEALTHINDEX 3

Definition at line 80 of file ydlidar_cmd.h.

29.31.1.5 #define INFO_DEFAULT_TIMEOUT 20

Definition at line 82 of file ydlidar_cmd.h.

29.31.1.6 #define LIDAR_ANS_SYNC_BYTE1 0xA5

Definition at line 38 of file ydlidar_cmd.h.

29.31.1.7 #define LIDAR_ANS_SYNC_BYTE2 0x5A

Definition at line 39 of file ydlidar_cmd.h.

29.31.1.8 #define LIDAR_ANS_TYPE_DEVHEALTH 0x6

Definition at line 35 of file ydlidar_cmd.h.

29.31.1.9 #define LIDAR_ANS_TYPE_DEVINFO 0x4

Definition at line 34 of file ydlidar_cmd.h.

29.31.1.10 #define LIDAR_ANS_TYPE_MEASUREMENT 0x81

Definition at line 40 of file ydlidar_cmd.h.

29.31.1.11 #define LIDAR_CMD_FORCE_SCAN 0x61

Definition at line 28 of file ydlidar_cmd.h.

29.31.1.12 #define LIDAR_CMD_FORCE_STOP 0x00

Definition at line 30 of file ydlidar_cmd.h.

29.31.1.13 #define LIDAR_CMD_GET_AIMSPEED 0x0D

Definition at line 53 of file ydlidar_cmd.h.

29.31.1.14 #define LIDAR_CMD_GET_DEVICE_HEALTH 0x92

Definition at line 33 of file ydlidar_cmd.h.

29.31.1.15 #define LIDAR_CMD_GET_DEVICE_INFO 0x90

Definition at line 32 of file ydlidar_cmd.h.

29.31.1.16 #define LIDAR_CMD_GET_EAI 0x55

Definition at line 31 of file ydlidar_cmd.h.

29.31.1.17 #define LIDAR_CMD_GET_OFFSET_ANGLE 0x93

Definition at line 61 of file ydlidar_cmd.h.

29.31.1.18 #define LIDAR_CMD_GET_SAMPLING_RATE 0xD1

Definition at line 56 of file ydlidar_cmd.h.

29.31.1.19 #define LIDAR_CMD_RESET 0x80

Definition at line 29 of file ydlidar_cmd.h.

29.31.1.20 #define LIDAR_CMD_RUN_INVERSION 0x07

Definition at line 48 of file ydlidar_cmd.h.

29.31.1.21 #define LIDAR_CMD_RUN_POSITIVE 0x06

Definition at line 47 of file ydlidar_cmd.h.

29.31.1.22 #define LIDAR_CMD_SCAN 0x60

Definition at line 27 of file ydlidar_cmd.h.

29.31.1.23 #define LIDAR_CMD_SET_AIMSPEED_ADD 0x0B

Definition at line 51 of file ydlidar_cmd.h.

29.31.1.24 #define LIDAR_CMD_SET_AIMSPEED_ADDMIC 0x09

Definition at line 49 of file ydlidar_cmd.h.

29.31.1.25 #define LIDAR_CMD_SET_AIMSPEED_DIS 0x0C

Definition at line 52 of file ydlidar_cmd.h.

29.31.1.26 #define LIDAR_CMD_SET_AIMSPEED_DISMIC 0x0A

Definition at line 50 of file ydlidar_cmd.h.

29.31.1.27 #define LIDAR_CMD_SET_SAMPLING_RATE 0xD0

Definition at line 55 of file ydlidar_cmd.h.

29.31.1.28 #define LIDAR_CMD_STOP 0x65

Definition at line 26 of file ydlidar_cmd.h.

29.31.1.29 #define LIDAR_CMD_SYNC_BYTE 0xA5

Definition at line 36 of file ydlidar_cmd.h.

29.31.1.30 #define LIDAR_CMDFLAG_HAS_PAYLOAD 0x80

Definition at line 37 of file ydlidar_cmd.h.

29.31.1.31 #define LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT 1

Definition at line 44 of file ydlidar_cmd.h.

29.31.1.32 #define LIDAR_RESP_MEASUREMENT_CHECKBIT (0x1<<0)

Definition at line 43 of file ydlidar_cmd.h.

29.31.1.33 #define LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT 2

Definition at line 45 of file ydlidar_cmd.h.

29.31.1.34 #define LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT 2

Definition at line 42 of file ydlidar_cmd.h.

29.31.1.35 #define LIDAR_RESP_MEASUREMENT_SYNCBIT (0x1<<0)

Definition at line 41 of file ydlidar_cmd.h.

29.31.1.36 #define LIDAR_STATUS_ERROR 0x2

Definition at line 59 of file ydlidar_cmd.h.

29.31.1.37 #define LIDAR_STATUS_OK 0x0

Definition at line 57 of file ydlidar_cmd.h.

29.31.1.38 #define LIDAR_STATUS_WARNING 0x1

Definition at line 58 of file ydlidar_cmd.h.

29.31.1.39 #define Node_Default_Quality (10<<2)

Definition at line 68 of file ydlidar_cmd.h.

29.31.1.40 #define Node_NotSync 0x00

Definition at line 70 of file ydlidar_cmd.h.

29.31.1.41 #define Node_Sync 0x01

Definition at line 69 of file ydlidar_cmd.h.

29.31.1.42 #define PackagePaidBytes 10

Definition at line 71 of file ydlidar_cmd.h.

29.31.1.43 #define PH 0x55AA

Definition at line 72 of file ydlidar_cmd.h.

29.31.1.44 #define READ_DEFAULT_TIMEOUT 1000

Definition at line 81 of file ydlidar_cmd.h.

29.31.1.45 #define SCAN_DEFAULT_TIMEOUT 100

Definition at line 83 of file ydlidar_cmd.h.

29.31.2 Enumeration Type Documentation

29.31.2.1 enum CT

Enumerator

CT_Normal

CT_RingStart

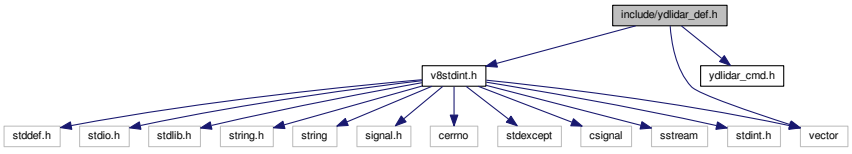
CT_Tail

Definition at line 63 of file ydlidar_cmd.h.

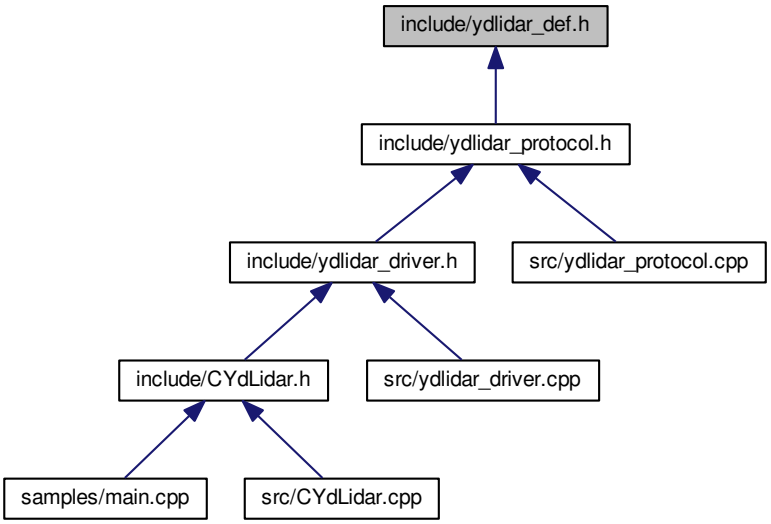
29.32 include/ydlidar_def.h File Reference

```
#include <v8stdint.h>
#include <vector>
#include <ydlidar_cmd.h>
```

Include dependency graph for ydlidar_def.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct node_package_header_t

  LiDAR Intensity Nodes Package.
- struct node_package_intensity_payload_t
- struct scan_intensity_packet_t
- struct node_package_payload_t

  package node info
- struct scan_packet_t
- struct device_info
- struct device_health
- struct sampling_rate_t
- struct scan_frequency_t
- struct offset_angle_t

  LiDAR Zero Offset Angle.
- struct cmd_packet_t
- struct lidar_ans_header_t

  LiDAR response Header.
- struct ct_packet_t
- struct LidarVersion
- struct LaserPoint
- struct LaserFan
- struct LaserConfig

  A struct for returning configuration from the YDLIDAR.
- struct LaserScan

Namespaces

- response_health_error
- response_scan_packet_sync

Macros

- #define SUNNOISEINTENSITY 0x03
- #define GLASSNOISEINTENSITY 0x02

Enumerations

- enum lidar_error_t {
  NoError, DeviceNotFoundError, PermissionError, OpenError,
  ParityError, FramingError, BreakConditionError, WriteError,
  ReadError, ResourceError, UnsupportedOperationError, TimeoutError,
  NotOpenError, HeaderError, FirstSampleAngleError, LastSampleAngleError,
  PackageNumberError, CheckSumError, SensorError, EncodeError,
  PWRError, PDError, LDError, DataError,
  TrembleError, LidarNotFoundError, UnknownError }
- enum LidarProperty {
  LidarPropSerialPort = 0, LidarPropIgnoreArray, LidarPropSerialBaudrate = 10, LidarPropLidarType,
  LidarPropDeviceType, LidarPropSampleRate, LidarPropAbnormalCheckCount, LidarPropMaxRange = 20,
  LidarPropMinRange, LidarPropMaxAngle, LidarPropMinAngle, LidarPropScanFrequency,
  LidarPropFixedResolution = 30, LidarPropReversion, LidarPropInverted, LidarPropAutoReconnect,
  LidarPropSingleChannel, LidarPropIntenstiy, LidarPropSupportMotorDtrCtrl, LidarPropSupportHeartBeat }
- enum response_health_error::bits : uint8_t {
  response_health_error::SensorError = 1 << 0, response_health_error::EncodeError = 1 << 1, response_health_error::PWRError = 1 << 2, response_health_error::PDError = 1 << 3,
  response_health_error::LDError = 1 << 4, response_health_error::DataError = 1 << 5, response_health_error::CSError = 1 << 6 }
- enum response_scan_packet_sync::bits : uint8_t {
  response_scan_packet_sync::sync = 1 << 0, response_scan_packet_sync::reserved1 = 1 << 1,
  response_scan_packet_sync::reserved2 = 1 << 2, response_scan_packet_sync::reserved3 = 1 << 3,
  response_scan_packet_sync::reserved4 = 1 << 4, response_scan_packet_sync::reserved5 = 1 << 5,
  response_scan_packet_sync::reserved6 = 1 << 6, response_scan_packet_sync::reserved7 = 1 << 7 }

29.32.1 Macro Definition Documentation

29.32.1.1 #define GLASSNOISEINTENSITY 0x02

Definition at line 31 of file ydlidar_def.h.

29.32.1.2 #define SUNNOISEINTENSITY 0x03

Definition at line 30 of file ydlidar_def.h.

29.32.2 Enumeration Type Documentation

29.32.2.1 enum lidar_error_t

Enumerator

NoError

DeviceNotFoundError

PermissionError
OpenError
ParityError
FramingError
BreakConditionError
WriteError
ReadError
ResourceError
UnsupportedOperationError
TimeoutError
NotOpenError
HeaderError
FirstSampleAngleError
LastSampleAngleError
PackageNumberError
CheckSumError
SensorError
EncodeError
PWRError
PDError
LDError
DataError
TrembleError
LidarNotFoundError
UnknownError

Definition at line 34 of file ydlidar_def.h.

29.32.2.2 enum LidarProperty

Lidar Properties,Lidar Can set and get parameter property index.
float properties must be float type, not double type.

Enumerator

LidarPropSerialPort Lidar serial port or network ipaddress
LidarPropIgnoreArray Lidar ignore angle array
LidarPropSerialBaudrate lidar serial baudrate or network port
LidarPropLidarType lidar type code
LidarPropDeviceType lidar connection type code
LidarPropSampleRate lidar sample rate
LidarPropAbnormalCheckCount abnormal maximum check times
LidarPropMaxRange lidar maximum range
LidarPropMinRange lidar minimum range
LidarPropMaxAngle lidar maximum angle
LidarPropMinAngle lidar minimum angle

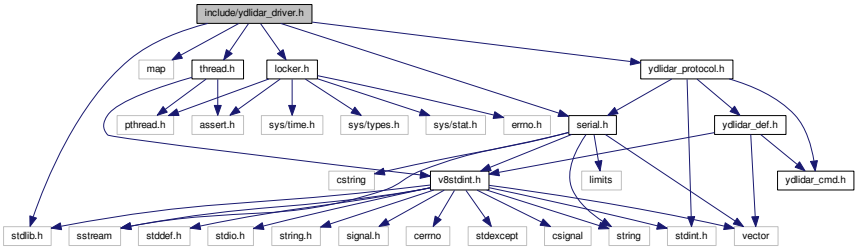
LidarPropScanFrequency lidar scanning frequency
LidarPropFixedResolution fixed angle resolution flag
LidarPropReversion lidar reversion flag
LidarPropInverted lidar inverted flag
LidarPropAutoReconnect lidar hot plug flag
LidarPropSingleChannel lidar single-channel flag
LidarPropIntenstiy lidar intensity flag
LidarPropSupportMotorDtrCtrl lidar support motor Dtr ctrl flag
LidarPropSupportHeartBeat lidar support heartbeat flag

Definition at line 67 of file ydlidar_def.h.

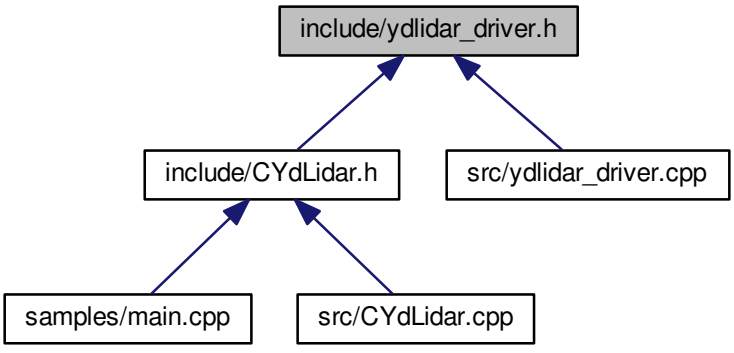
29.33 include/ydlidar_driver.h File Reference

```
#include <stdlib.h>
#include <map>
#include "serial.h"
#include "locker.h"
#include "thread.h"
#include "ydlidar_protocol.h"
```

Include dependency graph for ydlidar_driver.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ydlidar::YDlidarDriver

Namespaces

- ydlidar

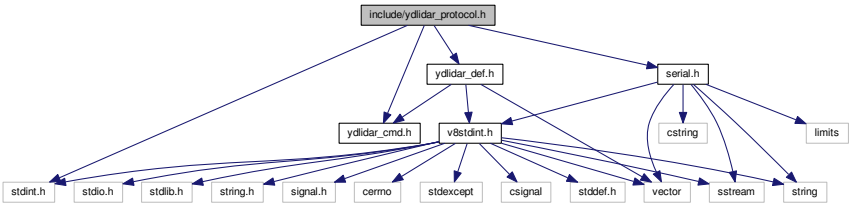
Functions

- std::string ydlidar::format (const char *fmt,...)

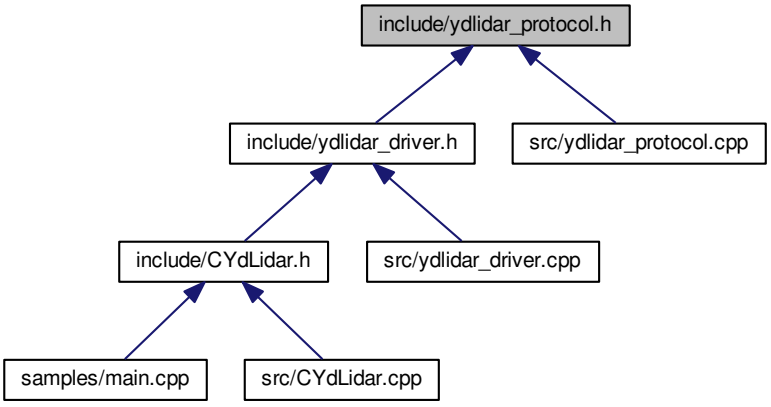
29.34 include/ydlidar_protocol.h File Reference

```
#include <stdint.h>
#include "ydlidar_def.h"
#include "ydlidar_cmd.h"
#include "serial.h"
```

Include dependency graph for ydlidar_protocol.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- ydlidar
- ydlidar::protocol

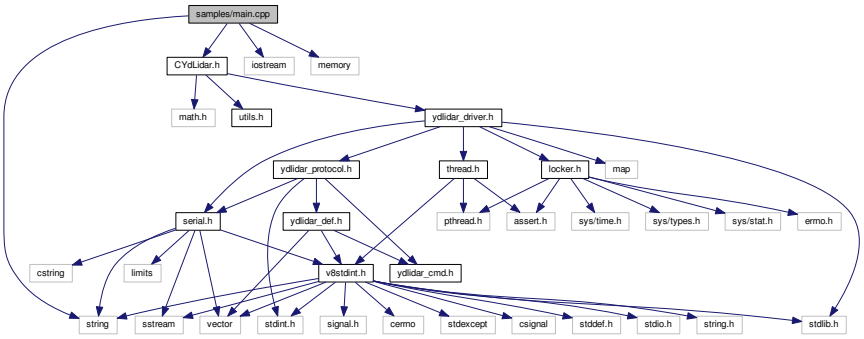
Functions

- const char * ydlidar::protocol::DescribeError (const lidar_error_t &error)
- void ydlidar::protocol::reset_ct_packet_t (ct_packet_t &ct)
- lidar_error_t ydlidar::protocol::convert_ct_packet_to_error (const ct_packet_t &ct)
- result_t ydlidar::protocol::check_ct_packet_t (const ct_packet_t &ct)
- void ydlidar::protocol::write_command (Serial *serial, uint8_t cmd)
- result_t ydlidar::protocol::wait_for_data (Serial *serial, size_t data_count, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_command (Serial *serial, uint8_t *buffer, size_t size, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_header_t (Serial *serial, lidar_ans_header_t &header, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::check_ans_header_t (const lidar_ans_header_t &header, lidar_error_t &error)
- result_t ydlidar::protocol::read_response_health_t (Serial *serial, device_health &health, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_device_info_t (Serial *serial, device_info &info, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_sample_rate_t (Serial *serial, sampling_rate_t &rate, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_frequency_t (Serial *serial, scan_frequency_t &frequency, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_offset_angle_t (Serial *serial, offset_angle_t &angle, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::parse_payload (const scan_packet_t &scan, LaserFan &data)
- result_t ydlidar::protocol::parse_intensity_payload (const scan_intensity_packet_t &scan, LaserFan &data)
- result_t ydlidar::protocol::check_package_header_t (const node_package_header_t &header, lidar_error_t &error)
- result_t ydlidar::protocol::parse_ct_packet_t (const node_package_header_t &header, unsigned short error_count, ct_packet_t &ct)
- uint8_t ydlidar::protocol::crc8_t (uint8_t *ptr, uint16_t len, uint8_t default_crc=0x00, uint8_t poly=0x8c, uint8_t inverted=1)
- uint16_t ydlidar::protocol::checksum_response_scan_packet_t (const scan_packet_t &scan)
- uint16_t ydlidar::protocol::checksum_response_scan_intensity_packet_t (const scan_intensity_packet_t &scan)
- result_t ydlidar::protocol::read_response_scan_header_t (Serial *serial, node_package_header_t &header, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_t (Serial *serial, scan_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_intensity_t (Serial *serial, scan_intensity_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)

29.35 samples/main.cpp File Reference

```
#include "CYdLidar.h"
#include <iostream>
#include <string>
#include <memory>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

29.35.1 Function Documentation

29.35.1.1 int main (int *argc,* char ∗ *argv[ ]*)

string property////////////////// lidar port

ignore array

int property////////////////// lidar baudrate

abnormal count

bool property////////////////// fixed angle resolution

rotate 180

Counterclockwise

float property////////////////// unit: °

unit: m

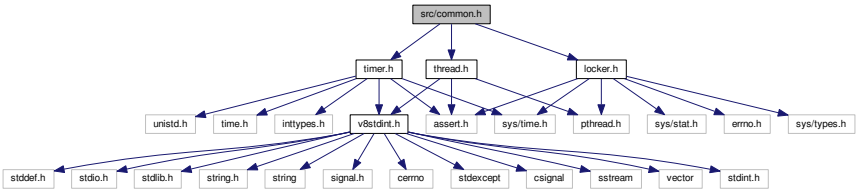
unit: Hz

Definition at line 35 of file main.cpp.

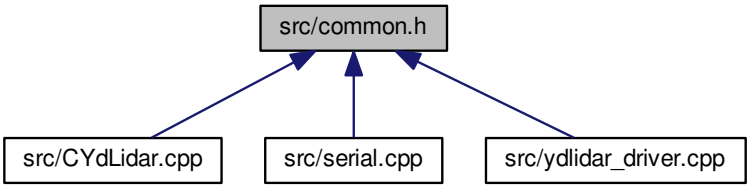
29.36 src/common.h File Reference

```
#include "locker.h"
#include "thread.h"
#include "timer.h"
```

Include dependency graph for common.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SDKVerision "1.0.0"

29.36.1 Macro Definition Documentation

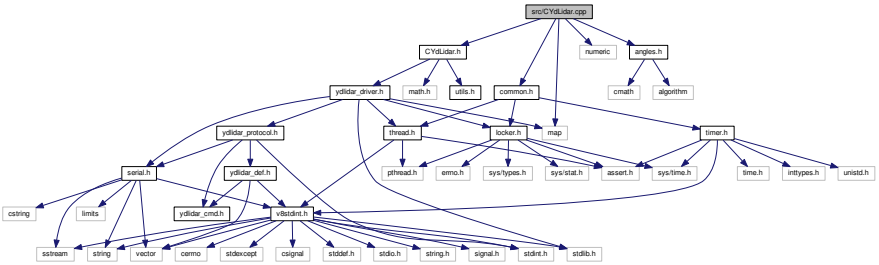
29.36.1.1 #define SDKVerision "1.0.0"

Definition at line 39 of file common.h.

29.37 src/CYdLidar.cpp File Reference

```
#include "CYdLidar.h"
#include "common.h"
#include <map>
#include <numeric>
#include "angles.h"
```

Include dependency graph for CYdLidar.cpp:



29.38 src/impl/unix/list_ports_linux.cpp File Reference

29.39 src/impl/unix/unix.h File Reference

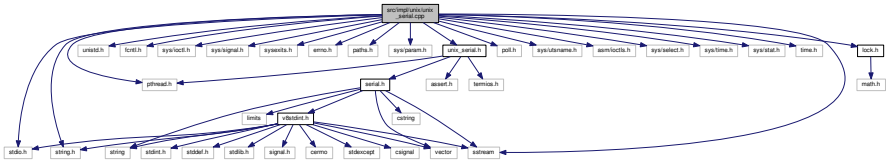
```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#include <time.h>
#include <stdarg.h>
#include <iostream>
#include <string>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/select.h>
```

Include dependency graph for unix.h:

29.40 src/impl/unix/unix_serial.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <sstream>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/signal.h>
#include <sysexits.h>
#include <errno.h>
#include <paths.h>
#include <sys/param.h>
#include <pthread.h>
#include <poll.h>
#include <sys/utsname.h>
#include <asm/ioctls.h>
#include <sys/select.h>
#include <sys/time.h>
#include <sys/stat.h>
#include <time.h>
#include "unix_serial.h"
#include <lock.h>
```

Include dependency graph for unix_serial.cpp:



Classes

- struct termios2

Macros

- #define TIOCINQ 0x541B
- #define SNCCS 19
- #define TCGETS2 _IOR('T', 0x2A, struct termios2)
- #define TCSETS2 _IOW('T', 0x2B, struct termios2)
- #define BOTHER 0010000

Functions

- timespec timespec_from_ms (const uint32_t millis)
- static void set_common_props (termios *tio)
- static void set_databits (termios *tio, serial::bytesize_t databits)
- static void set_parity (termios *tio, serial::parity_t parity)
- static void set_stopbits (termios *tio, serial::stopbits_t stopbits)
- static void set_flowcontrol (termios *tio, serial::flowcontrol_t flowcontrol)
- static bool is_standardbaudrate (unsigned long baudrate, speed_t &baud)

29.40.1 Macro Definition Documentation

29.40.1.1 #define BOTHER 0010000

Definition at line 196 of file unix_serial.cpp.

29.40.1.2 #define SNCCS 19

Definition at line 173 of file unix_serial.cpp.

29.40.1.3 #define TCGETS2 _IOR('T', 0x2A, struct termios2)

Definition at line 188 of file unix_serial.cpp.

29.40.1.4 #define TCSETS2 _IOW('T', 0x2B, struct termios2)

Definition at line 192 of file unix_serial.cpp.

29.40.1.5 #define TIOCINQ 0x541B

Definition at line 47 of file unix_serial.cpp.

29.40.2 Function Documentation

29.40.2.1 static bool is_standardbaudrate (unsigned long *baudrate,* speed_t & *baud*) [inline], [static]

Definition at line 399 of file unix_serial.cpp.

29.40.2.2 static void set_common_props (termios * *tio*) [inline], [static]

Definition at line 272 of file unix_serial.cpp.

29.40.2.3 static void set_databits (termios * *tio,* serial::bytesize_t *databits*) [inline], [static]

Definition at line 289 of file unix_serial.cpp.

29.40.2.4 static void set_flowcontrol (termios * *tio,* serial::flowcontrol_t *flowcontrol*) [inline], [static]

Definition at line 373 of file unix_serial.cpp.

29.40.2.5 static void set_parity (termios * *tio,* serial::parity_t *parity*) [inline], [static]

Definition at line 316 of file unix_serial.cpp.

29.40.2.6 static void set_stopbits (termios * *tio,* serial::stopbits_t *stopbits*) [inline], [static]

Definition at line 357 of file unix_serial.cpp.

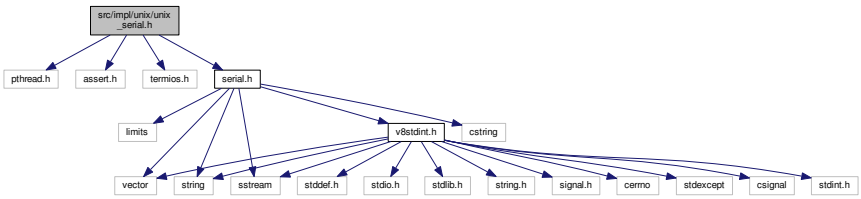
29.40.2.7 timespec timespec_from_ms (const uint32_t *millis*)

Definition at line 264 of file unix_serial.cpp.

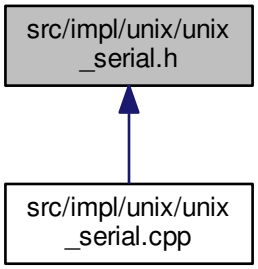
29.41 src/impl/unix/unix_serial.h File Reference

```
#include <pthread.h>
#include <assert.h>
#include <termios.h>
#include "serial.h"
```

Include dependency graph for unix_serial.h:



This graph shows which files directly or indirectly include this file:



Classes

- class serial::MillisecondTimer
- class serial::Serial::SerialImpl

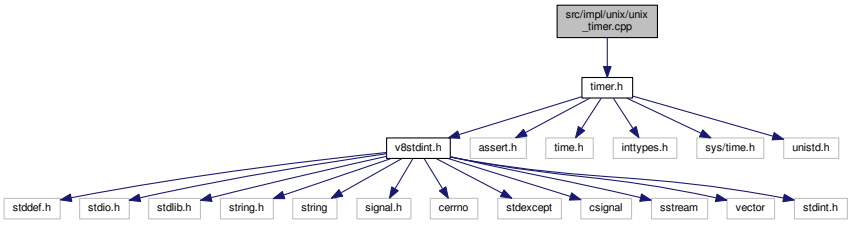
Namespaces

- serial

29.42 src/impl/unix/unix_timer.cpp File Reference

```
#include "timer.h"
```

Include dependency graph for unix_timer.cpp:



Namespaces

- impl

Functions

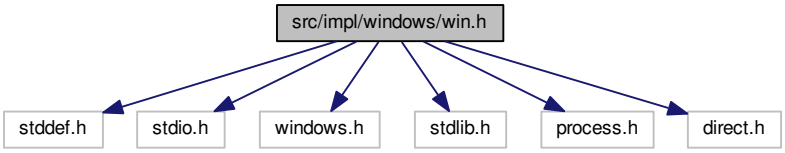
- uint32_t impl::getHDTimer ()
- uint64_t impl::getCurrentTime ()

29.43 src/impl/windows/list_ports_win.cpp File Reference

29.44 src/impl/windows/win.h File Reference

```
#include <stddef.h>
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <process.h>
#include <direct.h>
```

Include dependency graph for win.h:



29.45 src/impl/windows/win_serial.cpp File Reference

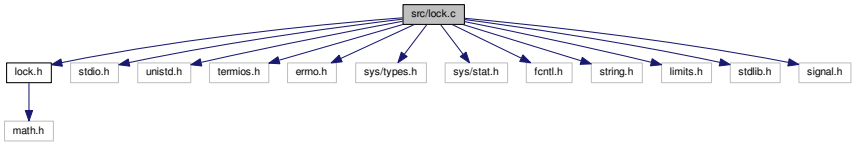
29.46 src/impl/windows/win_serial.h File Reference

29.47 src/impl/windows/win_timer.cpp File Reference

29.48 src/lock.c File Reference

```
#include "lock.h"
#include <stdio.h>
#include <unistd.h>
#include <termios.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <signal.h>
```

Include dependency graph for lock.c:



Functions

- int fhs_lock (const char *filename, int pid)
- int uucp_lock (const char *filename, int pid)
- int check_lock_status (const char *filename)
- void fhs_unlock (const char *filename, int openpid)
- void uucp_unlock (const char *filename, int openpid)
- int check_lock_pid (const char *file, int openpid)
- int check_group_uucp ()
- int is_device_locked (const char *port_filename)

29.48.1 Function Documentation

29.48.1.1 int check_group_uucp ( )

Definition at line 501 of file lock.c.

29.48.1.2 int check_lock_pid (const char * *file,* int *openpid*)

Definition at line 448 of file lock.c.

29.48.1.3 int check_lock_status (const char * *filename*)

Definition at line 340 of file lock.c.

29.48.1.4 int fhs_lock (const char * *filename,* int *pid*)

Definition at line 198 of file lock.c.

29.48.1.5 void fhs_unlock (const char * *filename,* int *openpid*)

Definition at line 377 of file lock.c.

29.48.1.6 int is_device_locked (const char * *port_filename*)

Definition at line 648 of file lock.c.

29.48.1.7 int uucp_lock (const char * *filename,* int *pid*)

Definition at line 280 of file lock.c.

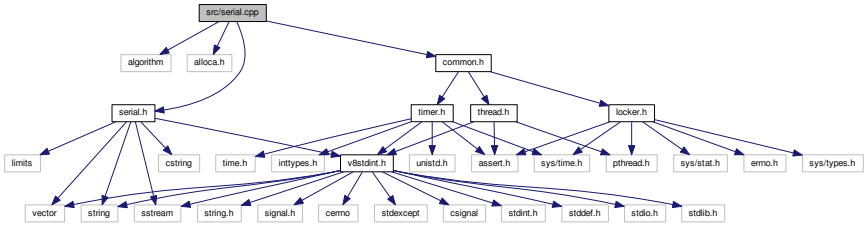
29.48.1.8 void uucp_unlock (const char * *filename,* int *openpid*)

Definition at line 407 of file lock.c.

29.49 src/serial.cpp File Reference

```
#include <algorithm>
#include <alloca.h>
#include "serial.h"
#include "common.h"
```

Include dependency graph for serial.cpp:



Classes

- class serial::Serial::ScopedReadLock
- class serial::Serial::ScopedWriteLock

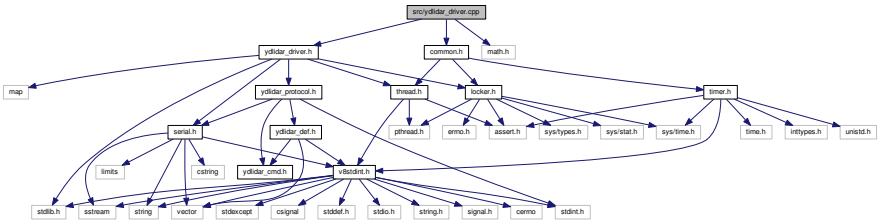
Namespaces

- serial

29.50 src/ydlidar_driver.cpp File Reference

```
#include "ydlidar_driver.h"
#include "common.h"
#include <math.h>
```

Include dependency graph for ydlidar_driver.cpp:



Namespaces

- ydlidar

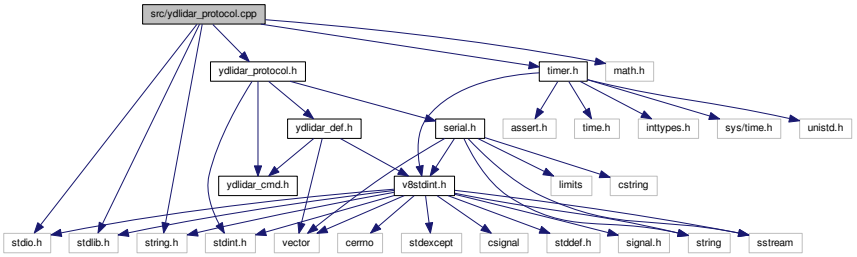
Functions

- std::string ydlidar::format (const char *fmt,...)

29.51 src/ydlidar_protocol.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "ydlidar_protocol.h"
#include "timer.h"
```

Include dependency graph for ydlidar_protocol.cpp:



Namespaces

- ydlidar
- ydlidar::protocol

Functions

- const char * ydlidar::protocol::DescribeError (const lidar_error_t &error)
- lidar_error_t ydlidar::protocol::convert_ct_packet_to_error (const ct_packet_t &ct)
- void ydlidar::protocol::reset_ct_packet_t (ct_packet_t &ct)
- result_t ydlidar::protocol::check_ct_packet_t (const ct_packet_t &ct)
- void ydlidar::protocol::write_command (Serial *serial, uint8_t cmd)
- result_t ydlidar::protocol::wait_for_data (Serial *serial, size_t data_count, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_command (Serial *serial, uint8_t *buffer, size_t size, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_header_t (Serial *serial, lidar_ans_header_t &header, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::check_ans_header_t (const lidar_ans_header_t &header, lidar_error_t &error)
- result_t ydlidar::protocol::read_response_health_t (Serial *serial, device_health &health, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_device_info_t (Serial *serial, device_info &info, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_sample_rate_t (Serial *serial, sampling_rate_t &rate, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_frequency_t (Serial *serial, scan_frequency_t &frequency, lidar_error_t &error, uint32_t timeout=1000)

- result_t ydlidar::protocol::read_response_offset_angle_t (Serial *serial, offset_angle_t &angle, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::parse_payload (const scan_packet_t &scan, LaserFan &data)
- result_t ydlidar::protocol::parse_intensity_payload (const scan_intensity_packet_t &scan, LaserFan &data)
- result_t ydlidar::protocol::check_package_header_t (const node_package_header_t &header, lidar_error_t &error)
- uint8_t ydlidar::protocol::crc8_t (uint8_t *ptr, uint16_t len, uint8_t default_crc=0x00, uint8_t poly=0x8c, uint8_t inverted=1)
- uint16_t ydlidar::protocol::checksum_response_scan_packet_t (const scan_packet_t &scan)
- uint16_t ydlidar::protocol::checksum_response_scan_intensity_packet_t (const scan_intensity_packet_t &scan)
- result_t ydlidar::protocol::parse_ct_packet_t (const node_package_header_t &header, unsigned short error_count, ct_packet_t &ct)
- result_t ydlidar::protocol::read_response_scan_header_t (Serial *serial, node_package_header_t &header, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_t (Serial *serial, scan_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)
- result_t ydlidar::protocol::read_response_scan_intensity_t (Serial *serial, scan_intensity_packet_t &scan, ct_packet_t &ct, lidar_error_t &error, uint32_t timeout=1000)

Index