Documentation of RiVLib, Riegl's V-Line Scanner Library

Documentation of RiVLib, Riegl's V-Line Scanner Library

Table of Contents

Overview

This document is about the RiVLib™, a C++ library for interfacing to Riegl's V-Line scanners.

Scanners of the V-Line employ a binary data protocol for measurement and status data that is different from previous riegl scanner devices. The protocol allows for variable sized data packets and bitsizes of data items that are not directly available as datatypes on the host cpu.

A data packet consists of a number of datafields. A data packet has an unique identifier and its layout is fixed, but it does not necessarily have fixed total size. Some packets are allowed to have a variable sized array at its tail.

Receiving data from a V-Line scanner means receiving a stream of packets. The packets are inserted into the stream by the various subsystems of the scanning device. Since the subsystems are not required to operate in synchronity the exact sequence of packet types cannot be predicted beforehand. The receiving side therefore will need to employ some kind of event driven processing.

The information present in a stream consists of calibration data which needs to be transfered only once followed by the (uncalibrated) measurements.

The objectives of RiVLib™ thus are:

- recognizing packets, i.e. determine their type and actual length,
- converting datatypes to native host types,
- dispatching the basic packets to user code or
- applying calibration and dispatching calibrated data.

User application code can interface to the library in one of two ways. For one a set of C++ header files and statically linkable libraries is available. Then a dynamically bindable library is available. Counter to what one would expect the both have different API's. The first one offers tight integration into C++ which is required to make use of C++'s templates. The second offers a plain C, i.e. function level API. Each of the both methods have its advantages and serve different objectives.

Part I. Getting started with the library

Table of Contents

Installation

Obtaining the library

You typically obtain the library files as part of the deliverables of your scanner device, or download them from riegls website at: http://www.riegl.com. For reference and troubleshooting you will also find older versions of the library in the download area. Normally you will want to pick the newest, i.e. with the highest version number. There are various packages of the library available. Finally you will need a variant that closely matches your compiler. If you obtained a package that has the term *-all-* in it you will need to further unpack the file until you find something related to your environemnt. Generally the name(s) of the packages are patterned after: `rivlib-`*`version-processor-os-compiler`*`.zip`

Prerequisites

Depending on how the library will be used there are different requirements:

- When used as a C++ library:
  - A supported C++ compiler: Currently one of Visual C++™ (9.0), Visual C++™ (10.0), Visual C++™ (12.0), gcc (4.4), mingw (4.6).
  - Compiler support for the C++ 2011 extensions.
- When used as a dynamically bound library interfacing to another language than C++:
  - Support to dynamically bind to a shared object (also known as DLL).
  - Support to convert the interfcae description from a C header file.

Setting up your environment

The library should be copied to a convenient place on the harddrive, where it can be found by the compiler. Whether you make the library part of your project or use it as a system wide library is up to you.

After unpacking the downloaded files you should have a directory structure similar to:

```
/rivlib_rootdir/   ..................... The RiVLib root directory.
    doc/   ............................. The documentation directory.
        rivlib.pdf   ................... Documentation in PDF format.
    bin/   ............................. The directory for executable programs.
    include/   ......................... The directory for compiler includes.
    lib/   ............................. The library subdirectory.
    examples/   ........................ Example files.
    rivlib-config.cmake   .............. CMake support file.
```

The name `/rivlib_rootdir/` is of course only an example. You may choose any directory name your compiler is able to use.

The binary subdirectory contains a couple of utility programs that may be useful during development but also as part of the final application software.

For using RiVLib with C++ you will need to configure your compile environment to

- find the `include/` directory and
- the `lib` subdirectory.

Because of the so called "name-mangling" of binary builds of C++ libraries you will need a library matching the compiler. RiVLib provides a set of variants for various compilers. If your compiler cannot be found in the list, please contact <support@riegl.com> to find out if we can add support for your compiler.

For using RiVLib as a DLL (or shared object) at minimum you will need the respective DLL. An import library and a corresponding C include header are provided in the set of files.

Compiling

Using RiVLib with C++

When using RiVLib with C++ you will need to statically link to it. Since RiVLib itself is making use of the boost™ library internally, you will also need to link to a couple of boost libraries. Please note that the boost libraries supplied as part of the RiVLib are different to the libraries you can download from the boost web site. The libraries that come with RiVLib are built within a dedicated namespace so they can be used without interference of the regular boost libraries, you possibly will be using with your project.

RiVLib comes with support for CMake. So if you are already using CMake to configure your project all you need is to follow the information supplied near the top in the `rivlib-config.cmake`.

If you are using another tool to configure your project, you likely will need to specify the include directories and libraries manually so your build will take them up. In general you need to find the correct name of the library according to the following general scheme.

Library naming

lib *Prefix:* except on Microsoft Windows, every library name begins with this string. On Windows, only ordinary static libraries use the lib prefix; import libraries and DLLs do not.

name *Library name:* for boost libraries the name begins with riboost_, other names are rivlib libraries.

-mt *Threading tag:* indicates that the library was built with multithreading support enabled. Libraries built without multithreading support can be identified by absence of this tag. (All RiVLib library components should have this tag.) Please note, that the presence of this tag does not necessarily mean that the library itself is multithread safe. You will need to refer to the reference of the library API to find this out. (As of this writing all libraries include this tag.)

-s *ABI Tag:* encodes details that affect the library's interoperability with other compiled code. For each such feature, a single letter is added to the tag:

Key	**Use this library when:**
s	linking statically to the C++ standard library and compiler runtime support libraries.
g	using debug versions of the standard and runtime support libraries.
d	using a debug build of a library.

.lib *Extension:* determined according to the operating system's usual convention. On most unix-style platforms the extensions are .a and .so for static libraries (archives) and shared libraries, respectively. On Windows, .dll indicates a shared library and .lib indicates a static or import library.

Library components

The C++ part of the library consists of the following components:

ctrllib Configuration and control of the scanner device.

scanlib Data retrieval and decoding of scanner data streams.

riboost Boost library components:`chrono`, `date_time`, `filesystem`, `regex`, `system`, `thread`.

In general you should link against all library components to resolve all symbols.

Using RiVLib as a DLL

When using RiVLib with another language than C++ or using an unsupported C++ compiler you should dynamically link to it and use the plain functional interface. No other library (except the compiler runtime lib) is needed in this case (in particular no boost libraries). Library naming follows the same scheme as in the static case, altough fewer variants will be available.

Part II. C++ Interface

Table of Contents

Scanlib

Scanlib Usage

*Riegl*s V-Line™ scanner deliver measurement data in a stream of variable sized binary coded packets. The start of a packet is marked with an unique bit combination that must not appear within the payload (the data) section of a packet. Consequently the payload section has to be encoded to meet this requirement. The method used, is having the transmitting side insert a special escape sequence whenever a (forbidden) packet start marker would appear as part of the real data. On the receiving side this procedure must be reverted before the data can be used as a binary packet. The RiVLib™ provides a special class `decoder_rxpmarker` that delivers a sequenece of binary packets without the escapes. The binary data then needs to be separated into the various information fields and dispatched to some program code that will act upon them.

The library provides some base classes that disassemble the binary raw data into data types that are native to the compiler and calls into a (virtual) function with the packet information supplied as an argument. Using the library typically means to derive your class from one of these base classes and override the virtual function(s) to perform the desired action. This might be a real time visual display, a calculation or conversion and storage into a database.

The part of the library that is directly visible to the user consists of several header files and binary, compiled libraries. The header files that can be found in the directory `riegl/` are considered to be part of the offcial and documented user interface. Header files that can be found in subdirectory `detail/` are considered implementation details, and you should avoid refering to them or their contents directly. Typically, instead of including exactly all of the header(s) really needed, you likely will want to include the convenience header `riegl/scanlib.hpp` instead.

RiVLib depends on the boost library. Therefore you not only need to link against `scanlib` but also against:

- `boost_datetime`,
- `boost_regex`,
- `boost_system` and
- `boost_thread`.

You need to link against the special boost libraries supplied with RiVLib, but at the same time you can link against another official version from the boost website, since the RiVLib version uses boost internally within a distinct (versioned) namespace.

RiVLib also makes use of some of the `std::tr1` extensions for C++. So do not forget to check your compiler if it already supports this functionality, or set up boost or miniboost (part of RiVLib).

All classes and functions of RiVLib are in namespace `scanlib`scanlib.

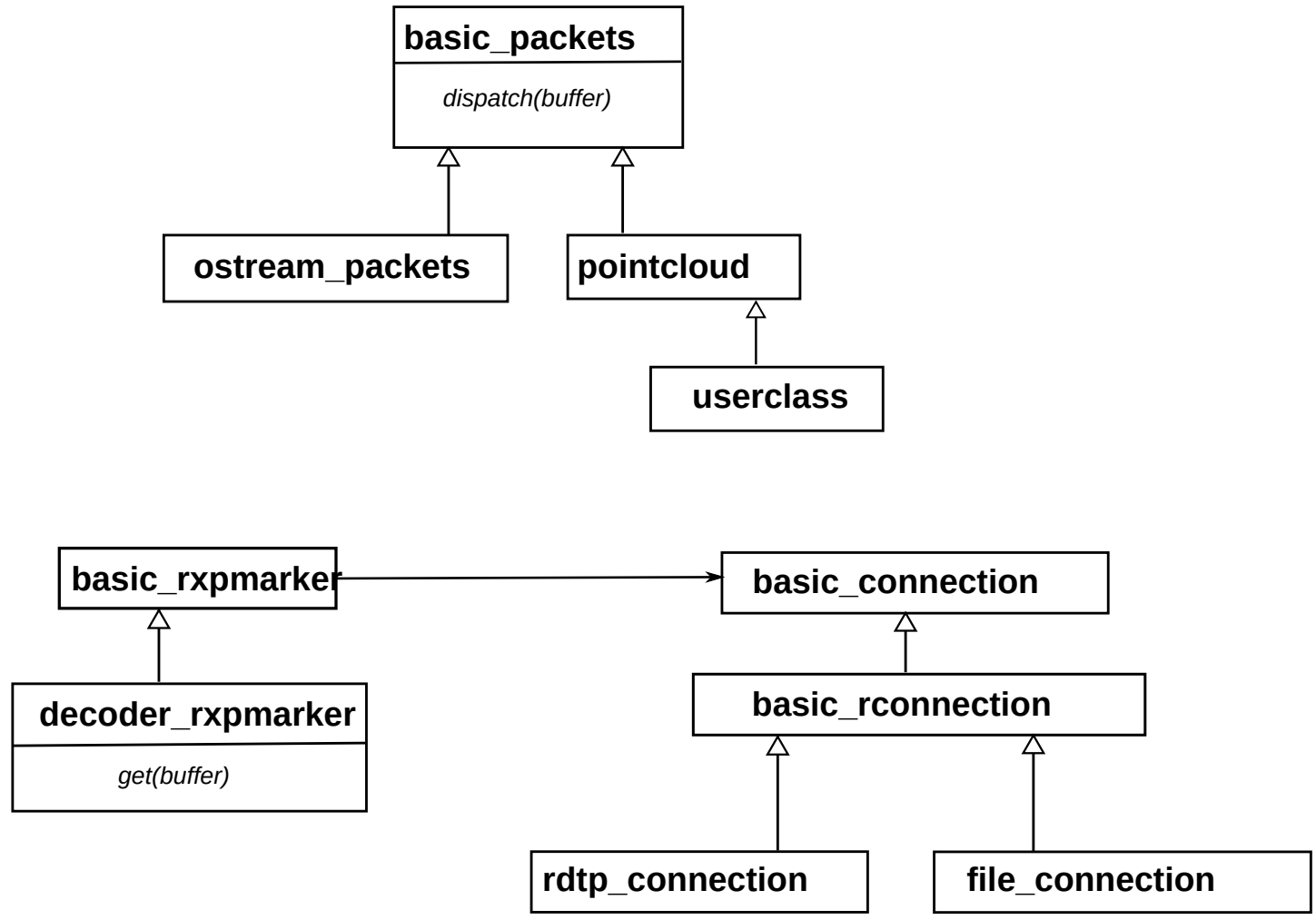
Error handling in general is making use of C++'s `throw/catch` mechanism. RiVLib's exception classes inherit from `std::exception`, so you can always get the error description from evaluating the `what()` member of the caught exception. The following example shows how you might wrap the main entry point of your application. This is particulary import if you are using a DLL, since the ability to throw exceptions across DLL boundaries is compiler and operating system dependant. Generally you would not want to let an exception cross a DLL boundary.

```
#include <riegl/scanlib.hpp>
#include <stdexcept>
#include <iostream>

using namespace std;
using namespace scanlib;

int main(int argc, char* argv[])
{
    try {
    // code, calling RiVLib here
    }
    catch(exception& e) {
```

```
        cerr << e.what() << endl;
        return 1;
    }
    catch(...) {
        cerr << "unknown exception" << endl;
        return 1;
    }
    return 0;
}
```



RiVLib's three main classes

1. `basic_connection` the base class for receiving data,
2. `decoder_rxpmarker` the decoder of escaped packets and
3. `basic_packets` the dispatching process class

are designed to implement the processing chain efficiently.

The processing chain for measurement packets has the following three main steps:

1. receive the data from the scanning device,
2. unescape the data, i.e. revert the encoding and
3. process each data packet.

Receiving

The abstract class `basic_connection` and its derived classes `file_rconnection` and `rdtp_rconnection` implement the data access layer of the RiVLib. The resource to open, e.g. a scanner device is specified using the URI (unified resource identifier) syntax: (Also look at RFC 3986.) The used uri parts are `scheme://authority/path ?query`.

scheme — *scheme* can be

- `rdtp` to establish a connection over the **R**iegl **D**ata **T**ransfer **P**rotocol or
- `file` to read data directly from a file on the local file system.

authority — *authority* depends on the used scheme. For `rdtp` it can be everything the operating systems TCP resolver service is able to resolve. This might be a simple hostname or an IP address. You can also specify a port number, but if you do so it must match the scaners port for the rdtp protocol. If you do not specify a port the default (port 20001) is used. If used together with the `file:` scheme this part typically is left empty.

path — *path* is a file system path when used with `file:` scheme. It designates the source of the scan data when used with `rdtp:`. The path element is mapped to the request command of the rdtp protocol. E.g. if you want to connect to the current data stream, this element would be `CURRENT`. For other available commands please refer to the scanner manaual.

query — *query* is used with `rdtp:` to supply options to the transfer protocol and to the remote end. The arguments are one ore more `key=value` pairs separated by `;` and are mapped to the options of the rdtp protocol as specified in the scanner manual.

If you wanted to connect to the `current` datastream on a scanner acccessible with IP address `192.168.0.33` and select the monitoring substream you would supply

```
rdtp://192.168.0.33/current?type=mon
```

for the URI. If you wanted to access a file with pathname `C:/scandata/testfile.rxp` on the local drive, you would supply

```
file:S:/scandata/testfile.rxp
```

for the URI.

The base class `basic_rconnection` has a static member function `create` which sometimes is termed a virtual constructor. It's purpose is to create a class that is derived from `basic_rconnection` based on the **scheme** of the supplied uri. This makes it possible to open files and live connections with the same syntax.

If the only thing you want to do is to store the data from the scanning device to a local disc, the `basic_rconnection` class is all you need. Simply within a loop call into the `readsome` and store the received bits to disk. This will result in a file that can be opened later by means of the `file:` **scheme**.

Decoding

Once the connection to the scanner or a file has been sucessfully established, the packet original boundaries have to be found and the original binary data has to be recovered from the data stream. This is done by the class `decoder_rxpmarker`. The class needs a connection as a parameter to the constructor and delivers a binary packet on each call to the `get` function.

Dispatching

The binary packets coming from the `decoder_rxpmarker` class are fed into a class that is derived from `basic_packets`. The base class has a virtual function corresponding to each possibly packet that can be generated by the scanning device. The derived class (the users class) need override only the functions for packets the user is interested in.

Caution

Always first call the base version of the overriden function, if not stated explicitely otherwise. The base version often needs to perform some operations that affect the state of the decoding. If you omit these calls, failures may result.

The dispatcher classes do not alter the contents of the binary data buffer fed to their `dispatch` functions. This makes it possible to call into more than one dispatcher class with the same data in succession. You might want to do this to separate tasks into different classes. E.g. one class could be dedicated to read pointcloud data while another monitors scanner alerts.

The library comes with two classes that accomplish common tasks:

- Converting a package to readable ASCII format.
- Converting the raw data to pointcloud data.

The `ostream_packets` class accepts a C++ standard library `ostream` into which it emits the packet data in plain ASCII format. This format is expected mainly useful for debugging purposes since it enlarges the amount of data to a large extent.

The second class `pointcloud` reads off the calibration data, raw range and angle data and the GPS timing information if available. The class assembles this information and calls into an additional virtual function whenever a calibrated point is ready for delivery. A user of the library most likely will derive his class from the `pointcloud` class since this is the most important data available from a scanner device.

The following skeleton example shows a custom class `my_class` derived from `pointcloud`. The class overrides the `on_echo_transformed` function an writes the data to an output stream that has been supplied by means of the constructor.

```
class my_class
: public pointcloud
{
    ostream& o_;

    public:
    my_class(ostream& o)
    : pointcloud(false) // set this to true if you need gps aligned timing
    , o_(o)
    {}

    protected:

    void on_echo_transformed(echo_type echo)
    {
        pointcloud::on_echo_transformed(echo);
        target& t(targets[target_count-1]);
        o_ << t.vertex[0] << ", " << t.vertex[1] << ", " << t.vertex[2]
           << ", " << t.time << endl;
    }
};
```

The next code snippet demonstates the basic interconnection of the main classes.

```
shared_ptr<basic_rconnection> rc = basic_rconnection::create("rdtp://192.168.0.42/CURRENT");
```

```
decoder_rxpmarker   dec(rc);
my_class            my(cout);
buffer              buf;

rc->open();

for ( dec.get(buf); !dec.eoi(); dec.get(buf) ) {
    my.dispatch(buf.begin(), buf.end());
}

rc->close();
```

Examples

The library comes with heavily documented examples that are considered part of the documentaion. Please study them or use them as starting points for your own development.

Example	**Topics covered**
pointcloudcpp	Deriving a custom class from pointcloud, setting up a basic dispatcher loop to read from a data source.

Scanlib Reference

Header <riegl/ridataspec.cpp>

```
namespace scanlib {
  const selector_type select_none(std:: string);
  const selector_type select_all(std:: string);
  const selector_type select_protocol(std:: string);
  const selector_type select_attribute(std:: string);
  const selector_type select_instrument(std:: string);
  const selector_type select_scan(std:: string);
  const selector_type select_debug(std:: string);
  const selector_type select_data(std:: string);
  const selector_type select_wave(std:: string);
  const selector_type select_deprecated(std:: string);
  const selector_type select_internal(std:: string);
  const selector_type select_status(std:: string);
  const selector_type select_packed(std:: string);
  const selector_type select_notify(std:: string);
  const selector_type select_legacy(std:: string);
  selector_type get_class_selector(const std::string &);
  selector_type get_class_selector(const char *);
}
```

Function get_class_selector

scanlib::get_class_selector — Get class selector by string name.

Synopsis

```
// In header: <riegl/ridataspec.cpp>


selector_type get_class_selector(const std::string & name);
```

Description

Returns a selector by giving a string representaion of its name. If the name is not a valid class name, the select_none selector will be returned.

Parameters: `name` any valid class name string

Function get_class_selector

scanlib::get_class_selector — Get class selector by string name.

Synopsis

```
// In header: <riegl/ridataspec.cpp>


selector_type get_class_selector(const char * name);
```

Description

Returns a selector by giving a string representaion of its name. If the name is not a valid class name, the select_none selector will be returned.

Parameters: `name` any valid class name string

Header <riegl/ridataspec.hpp>

```
RIDATASPEC_VERSION
```

```
namespace scanlib {
  template<typename it> struct atmosphere;
  template<typename it> struct atmosphere_1;
  template<typename it> struct atmosphere_2;
  template<typename it> struct atmosphere_3;
  template<typename it> struct atmosphere_4;

  class basic_packets;

  template<typename it> struct beam_geometry;
  template<typename it> struct biaxial_geometry;
  template<typename it> struct counter_sync;
  template<typename it> struct device_mounting;
  template<typename it> struct extents;
  template<typename it> struct frame_start_dn;
  template<typename it> struct frame_start_up;
  template<typename it> struct frame_stop;
  template<typename it> struct header;
  template<typename it> struct header_ext;
  template<typename it> struct hk_bat;
  template<typename it> struct hk_bat_1;
  template<typename it> struct hk_bat_2;
  template<typename it> struct hk_ctr;
  template<typename it> struct hk_ctr_1;
  template<typename it> struct hk_gps;
  template<typename it> struct hk_gps_hr;
  template<typename it> struct hk_gps_ts;
  template<typename it> struct hk_gps_ts_status;
  template<typename it> struct hk_gps_ts_status_dop;
  template<typename it> struct hk_gps_ts_status_dop_ucs;
  template<typename it> struct hk_incl;
  template<typename it> struct hk_incl_4axes;
  template<typename it> struct hk_ph_data;
  template<typename it> struct hk_ph_data_1;
  template<typename it> struct hk_ph_units;
  template<typename it> struct hk_ph_units_1;
  template<typename it> struct hk_pwr;
  template<typename it> struct hk_pwr_1;
  template<typename it> struct hk_rtc;
  template<typename it> struct hk_rtc_sys;
  template<typename it> struct IMU_data;
  template<typename it> struct inclination_wyler;
  template<typename it> struct line_start_dn;
  template<typename it> struct line_start_segment_1;
  template<typename it> struct line_start_segment_2;
  template<typename it> struct line_start_segment_3;
  template<typename it> struct line_start_up;
  template<typename it> struct line_stop;
  template<typename it> struct meas_start;
```

```
  template<typename it> struct meas_stop;
  template<typename it> struct monitoring_info;
  template<typename it> struct mta_settings;
  template<typename it> struct mta_settings_1;
  template<typename it> struct mta_settings_2;

  class ostream_packets;

  struct package_id;
  template<typename it> struct pps_sync;
  template<typename it> struct pps_sync_ext;
  template<typename it> struct pps_sync_hr;
  template<typename it> struct pps_sync_hr_ext;
  template<typename it> struct pwm_sync;
  template<typename it> struct scan_rect_fov;
  template<typename it> struct scan_segments_fov;
  template<typename it> struct scanner_pose;
  template<typename it> struct scanner_pose_hr;
  template<typename it> struct scanner_pose_hr_1;
  template<typename it> struct scanner_pose_ucs;
  template<typename it> struct scanner_pose_ucs_1;
  template<typename it> struct units;
  template<typename it> struct units_1;
  template<typename it> struct units_2;
  template<typename it> struct units_3;
  template<typename it> struct units_IMU;
  template<typename it> struct unsolicited_message;
  template<typename it> struct unsolicited_message_1;
  template<typename it> struct void_data;
  typedef std::bitset< 202 > selector_type;

  const selector_type select_all;  // select all packets
  const selector_type select_none;  // select no packets
  const selector_type select_protocol;  // select packets of protocol class
  const selector_type select_attribute;  // select packets of attribute class
  const selector_type select_instrument;  // select packets of instrument class
  const selector_type select_scan;  // select packets of scan class
  const selector_type select_data;  // select packets of data class
  const selector_type select_deprecated;  // select packets of deprecated class
  const selector_type select_status;  // select packets of status class
  const selector_type select_notify;  // select packets of notify class
  const selector_type select_legacy;  // select packets of legacy class
}
```

Struct template atmosphere

scanlib::atmosphere — environmental information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct atmosphere {

  enum @1 { id_main = = 27, id_sub = = 0 };

  // public data members
  float temperature;  // user specified average temperature along measurement path [C]
  float pressure;  // user specified average pressure along measurment path [mbar]
  float rel_humidity;  // user specified relative humidity along measurement path [%]
```

```
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template atmosphere_1

scanlib::atmosphere_1 — extended environmental information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct atmosphere_1 {

  enum @2 { id_main = = 27, id_sub = = 1 };

  // public data members
  float temperature;  // user specified average temperature along measurement path [C]
  float pressure;  // user specified average pressure along measurment path [mbar]
  float rel_humidity;  // user specified relative humidity along measurement path [%]
  float pressure_sl;  // user specified atmospheric pressure at sea level [mbar]
  float amsl;  // user specified height above mean sea level (AMSL) [m]
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template atmosphere_2

scanlib::atmosphere_2 — extended environmental information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct atmosphere_2 {

  enum @3 { id_main = = 27, id_sub = = 2 };

  // public data members
  float temperature;  // user specified average temperature along measurement path [C]
  float pressure;  // user specified average pressure along measurment path [mbar]
  float rel_humidity;  // user specified relative humidity along measurement path [%]
  float pressure_sl;  // user specified atmospheric pressure at sea level [mbar]
  float amsl;  // user specified height above mean sea level (AMSL) [m]
  float group_velocity;  // effective group velocity of laser beam
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template atmosphere_3

scanlib::atmosphere_3 — extended environmental information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct atmosphere_3 {

  enum @4 { id_main = = 27, id_sub = = 3 };

  // public data members
  float temperature;  // user specified average temperature along measurement path [C]
  float pressure;  // user specified average pressure along measurment path [mbar]
  float rel_humidity;  // user specified relative humidity along measurement path [%]
  float pressure_sl;  // user specified atmospheric pressure at sea level [mbar]
  float amsl;  // user specified height above mean sea level (AMSL) [m]
  float group_velocity;  // effective group velocity of laser beam
  float attenuation;  // atmospheric attenuation, used for correction of reflectance [1/km]
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template atmosphere_4

scanlib::atmosphere_4 — extended environmental information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct atmosphere_4 {

  enum @5 { id_main = = 27, id_sub = = 4 };

  // public data members
  float temperature;  // user specified average temperature along measurement path [C]
  float pressure;  // user specified average pressure along measurment path [mbar]
  float rel_humidity;  // user specified relative humidity along measurement path [%]
  float pressure_sl;  // user specified atmospheric pressure at sea level [mbar]
  float amsl;  // user specified height above mean sea level (AMSL) [m]
  float group_velocity;  // effective group velocity of laser beam
  float attenuation;  // atmospheric attenuation, used for correction of reflectance [1/km]
  float wavelength;  // wavelength of the laser in units of nm
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Class basic_packets

scanlib::basic_packets — The base for packet dispatcher classes.

Synopsis

```
// In header: <riegl/ridataspec.hpp>


class basic_packets {
public:
  // types
  typedef decoder_rxpmarker::const_iterator iterator_type;  // embedded iterator type defini-
tion

  // construct/copy/destruct
  basic_packets();
  ~basic_packets();

  // public member functions
  bool dispatch(const iterator_type &, const iterator_type &);

  // protected member functions
  virtual void
  on_id(const package_id &, const basic_package< iterator_type > &);
  virtual void on_IMU_data(const IMU_data< iterator_type > &);
  virtual void on_atmosphere(const atmosphere< iterator_type > &);
  virtual void on_atmosphere_1(const atmosphere_1< iterator_type > &);
  virtual void on_atmosphere_2(const atmosphere_2< iterator_type > &);
  virtual void on_atmosphere_3(const atmosphere_3< iterator_type > &);
  virtual void on_atmosphere_4(const atmosphere_4< iterator_type > &);
  virtual void on_beam_geometry(const beam_geometry< iterator_type > &);
  virtual void on_biaxial_geometry(const biaxial_geometry< iterator_type > &);
  virtual void on_counter_sync(const counter_sync< iterator_type > &);
  virtual void on_device_mounting(const device_mounting< iterator_type > &);
  virtual void on_extents(const extents< iterator_type > &);
  virtual void on_frame_start_dn(const frame_start_dn< iterator_type > &);
  virtual void on_frame_start_up(const frame_start_up< iterator_type > &);
  virtual void on_frame_stop(const frame_stop< iterator_type > &);
  virtual void on_header(const header< iterator_type > &);
  virtual void on_header_ext(const header_ext< iterator_type > &);
  virtual void on_hk_bat(const hk_bat< iterator_type > &);
  virtual void on_hk_bat_1(const hk_bat_1< iterator_type > &);
  virtual void on_hk_bat_2(const hk_bat_2< iterator_type > &);
  virtual void on_hk_ctr(const hk_ctr< iterator_type > &);
  virtual void on_hk_ctr_1(const hk_ctr_1< iterator_type > &);
  virtual void on_hk_gps(const hk_gps< iterator_type > &);
  virtual void on_hk_gps_hr(const hk_gps_hr< iterator_type > &);
  virtual void on_hk_gps_ts(const hk_gps_ts< iterator_type > &);
  virtual void on_hk_gps_ts_status(const hk_gps_ts_status< iterator_type > &);
  virtual void
  on_hk_gps_ts_status_dop(const hk_gps_ts_status_dop< iterator_type > &);
  virtual void
  on_hk_gps_ts_status_dop_ucs(const hk_gps_ts_status_dop_ucs< iterator_type > &);
  virtual void on_hk_incl(const hk_incl< iterator_type > &);
  virtual void on_hk_incl_4axes(const hk_incl_4axes< iterator_type > &);
```

```
  virtual void on_hk_ph_data(const hk_ph_data< iterator_type > &);
  virtual void on_hk_ph_data_1(const hk_ph_data_1< iterator_type > &);
  virtual void on_hk_ph_units(const hk_ph_units< iterator_type > &);
  virtual void on_hk_ph_units_1(const hk_ph_units_1< iterator_type > &);
  virtual void on_hk_pwr(const hk_pwr< iterator_type > &);
  virtual void on_hk_pwr_1(const hk_pwr_1< iterator_type > &);
  virtual void on_hk_rtc(const hk_rtc< iterator_type > &);
  virtual void on_hk_rtc_sys(const hk_rtc_sys< iterator_type > &);
  virtual void
  on_inclination_wyler(const inclination_wyler< iterator_type > &);
  virtual void on_line_start_dn(const line_start_dn< iterator_type > &);
  virtual void
  on_line_start_segment_1(const line_start_segment_1< iterator_type > &);
  virtual void
  on_line_start_segment_2(const line_start_segment_2< iterator_type > &);
  virtual void
  on_line_start_segment_3(const line_start_segment_3< iterator_type > &);
  virtual void on_line_start_up(const line_start_up< iterator_type > &);
  virtual void on_line_stop(const line_stop< iterator_type > &);
  virtual void on_meas_start(const meas_start< iterator_type > &);
  virtual void on_meas_stop(const meas_stop< iterator_type > &);
  virtual void on_monitoring_info(const monitoring_info< iterator_type > &);
  virtual void on_mta_settings(const mta_settings< iterator_type > &);
  virtual void on_mta_settings_1(const mta_settings_1< iterator_type > &);
  virtual void on_mta_settings_2(const mta_settings_2< iterator_type > &);
  virtual void on_pps_sync(const pps_sync< iterator_type > &);
  virtual void on_pps_sync_ext(const pps_sync_ext< iterator_type > &);
  virtual void on_pps_sync_hr(const pps_sync_hr< iterator_type > &);
  virtual void on_pps_sync_hr_ext(const pps_sync_hr_ext< iterator_type > &);
  virtual void on_pwm_sync(const pwm_sync< iterator_type > &);
  virtual void on_scan_rect_fov(const scan_rect_fov< iterator_type > &);
  virtual void
  on_scan_segments_fov(const scan_segments_fov< iterator_type > &);
  virtual void on_scanner_pose(const scanner_pose< iterator_type > &);
  virtual void on_scanner_pose_hr(const scanner_pose_hr< iterator_type > &);
  virtual void
  on_scanner_pose_hr_1(const scanner_pose_hr_1< iterator_type > &);
  virtual void on_scanner_pose_ucs(const scanner_pose_ucs< iterator_type > &);
  virtual void
  on_scanner_pose_ucs_1(const scanner_pose_ucs_1< iterator_type > &);
  virtual void on_units(const units< iterator_type > &);
  virtual void on_units_1(const units_1< iterator_type > &);
  virtual void on_units_2(const units_2< iterator_type > &);
  virtual void on_units_3(const units_3< iterator_type > &);
  virtual void on_units_IMU(const units_IMU< iterator_type > &);
  virtual void
  on_unsolicited_message(const unsolicited_message< iterator_type > &);
  virtual void
  on_unsolicited_message_1(const unsolicited_message_1< iterator_type > &);
  virtual void on_void_data(const void_data< iterator_type > &);
  virtual void
  on_unknown(unsigned, unsigned, const iterator_type &, const iterator_type &);
  void dont_handle_parent();
  void request_dispatch_end();

  // public data members
  selector_type selector;  // package dispatch selection
  package_id id;
};
```

Description

An instance of this class implements package dispatching. Raw packages are fed to the class via the dispatch member function. The raw package will be decoded and a virtual function corresponding to its type will be called. The overloaded function will receive the decoded package as a paramter. Each packet has several fields which are separately documented. The actual implementation of packets differs from how they appear in the documentation. Instead of the shown packets the handler funtions receive a proxy of the packet, that acts as if it was the real packet. This means that one can use the various fields in read (and write, where apropriate) expressions, but one cannot e.g. take the address of a field or make a reference to it. If you need to know the real implementation, please look at the source code.

The "it" template paramater specifies an iterator to a raw package, which in principle can be a random access pointer to unsigned of various bitsizes. In practice the iterator always shall be taken from the one defined in the buffer class.

Package dispatching can be fine controlled by setting the selector, which is a bitset, indexed by the package id. The package id to use for indexing is one of the enum values defined in struct package_id. A couple of predefined sets are available. Please note, that while it is possible to create arbitrary selector combinations, care should be taken to always include the mandatory header packet.

`basic_packets` public construct/copy/destruct

1.
```
basic_packets();
```

2.
```
~basic_packets();
```

`basic_packets` public member functions

1.
```
bool dispatch(const iterator_type & begin, const iterator_type & end);
```

dispatch a raw packet

This function invokes the dispatch process. It is fed with a buffer taken from the `decoder_rxpmarker`. The dispatcher first infers the type of the raw packet from its embeded id and a lookup table which has been provided from the instrument in the header packet. Then the dispatcher uses the major and minor packet id numbers to map the packet to a handler function. A received packet with a higher minor number is compatible to any packet of the same major but smaller minor number. The dispatcher will use such packets in place of these smaller numbered packets.

The dispatching process is subject to a selector filter which has a flag for each packet.

`basic_packets` protected member functions

1.
```
virtual void
on_id(const package_id & arg, const basic_package< iterator_type > & pkg);
```

2.
```
virtual void on_IMU_data(const IMU_data< iterator_type > & arg);
```

Parameters: `arg` IMU raw data

3.
```
virtual void on_atmosphere(const atmosphere< iterator_type > & arg);
```

Parameters: `arg` environmental information

4.
```
virtual void on_atmosphere_1(const atmosphere_1< iterator_type > & arg);
```

Parameters: `arg` extended environmental information

5.
```
virtual void on_atmosphere_2(const atmosphere_2< iterator_type > & arg);
```

Parameters: `arg` extended environmental information

6.
```
virtual void on_atmosphere_3(const atmosphere_3< iterator_type > & arg);
```

Parameters: `arg` extended environmental information

7.
```
virtual void on_atmosphere_4(const atmosphere_4< iterator_type > & arg);
```

Parameters: `arg` extended environmental information

8.
```
virtual void on_beam_geometry(const beam_geometry< iterator_type > & arg);
```

Parameters: `arg` laser beam description

9.
```
virtual void
on_biaxial_geometry(const biaxial_geometry< iterator_type > & arg);
```

Parameters: `arg` modelling biaxial shift of cog of range measurement in the near range

10.
```
virtual void on_counter_sync(const counter_sync< iterator_type > & arg);
```

Parameters: `arg` external synchronization input

11.
```
virtual void on_device_mounting(const device_mounting< iterator_type > & arg);
```

Parameters: `arg` geometrical parameters of external devices

12.
```
virtual void on_extents(const extents< iterator_type > & arg);
```

Parameters: `arg` extents of various data fields

13.
```
virtual void on_frame_start_dn(const frame_start_dn< iterator_type > & arg);
```

Parameters: `arg` start of a scan frame in down direction.

14.
```
virtual void on_frame_start_up(const frame_start_up< iterator_type > & arg);
```

Parameters: `arg` Start of a scan frame in up direction.

15.
```
virtual void on_frame_stop(const frame_stop< iterator_type > & arg);
```

Parameters: `arg` end of a scan frame

16.
```
virtual void on_header(const header< iterator_type > & arg);
```

Parameters: `arg` The mandatory header package.

17.
```
virtual void on_header_ext(const header_ext< iterator_type > & arg);
```

Parameters: `arg` Extension header

18.
```
virtual void on_hk_bat(const hk_bat< iterator_type > & arg);
```

Parameters: `arg` power supply unit (24 Bytes)

19.
```
virtual void on_hk_bat_1(const hk_bat_1< iterator_type > & arg);
```

Parameters: `arg` power supply unit (24 Bytes)

20.
```
virtual void on_hk_bat_2(const hk_bat_2< iterator_type > & arg);
```

Parameters: `arg` power supply unit (24 Bytes)

21.
```
virtual void on_hk_ctr(const hk_ctr< iterator_type > & arg);
```

Parameters: `arg` control unit, i.e., mother board, including storage levels, info on attached equipment (60 Bytes)

22.
```
virtual void on_hk_ctr_1(const hk_ctr_1< iterator_type > & arg);
```

Parameters: `arg` control unit, i.e., mother board, including storage levels, info on attached equipment (60 Bytes)

23.
```
virtual void on_hk_gps(const hk_gps< iterator_type > & arg);
```

Parameters: `arg` GPS data

24.
```
virtual void on_hk_gps_hr(const hk_gps_hr< iterator_type > & arg);
```

Parameters: `arg` GPS data

25.
```
virtual void on_hk_gps_ts(const hk_gps_ts< iterator_type > & arg);
```

Parameters: `arg` GPS data

26.
```
virtual void
on_hk_gps_ts_status(const hk_gps_ts_status< iterator_type > & arg);
```

Parameters: `arg` GPS data

27.
```
virtual void
on_hk_gps_ts_status_dop(const hk_gps_ts_status_dop< iterator_type > & arg);
```

Parameters: `arg` GPS data

28.
```
virtual void
on_hk_gps_ts_status_dop_ucs(const hk_gps_ts_status_dop_ucs< iterator_type > & arg);
```

Parameters: `arg` GPS data

29.
```
virtual void on_hk_incl(const hk_incl< iterator_type > & arg);
```

Parameters: `arg` inclination sensor

30.
```
virtual void on_hk_incl_4axes(const hk_incl_4axes< iterator_type > & arg);
```

Parameters: `arg` inclination sensor

31.
```
virtual void on_hk_ph_data(const hk_ph_data< iterator_type > & arg);
```

Parameters: `arg` protective housing data

32.
```
virtual void on_hk_ph_data_1(const hk_ph_data_1< iterator_type > & arg);
```

Parameters: `arg` protective housing data

33.
```
virtual void on_hk_ph_units(const hk_ph_units< iterator_type > & arg);
```

Parameters: `arg` protective housing units

34.
```
virtual void on_hk_ph_units_1(const hk_ph_units_1< iterator_type > & arg);
```

Parameters: `arg` protective housing units

35.
```
virtual void on_hk_pwr(const hk_pwr< iterator_type > & arg);
```

Parameters: `arg` power supply unit (24 Bytes)

36.
```
virtual void on_hk_pwr_1(const hk_pwr_1< iterator_type > & arg);
```

Parameters: `arg` power supply unit (24 Bytes)

37.
```
virtual void on_hk_rtc(const hk_rtc< iterator_type > & arg);
```

Parameters: `arg` built in real time clock of scanning device, local time

38.
```
virtual void on_hk_rtc_sys(const hk_rtc_sys< iterator_type > & arg);
```

Parameters: `arg` real time clock, local time (10 Bytes)

39.
```
virtual void
on_inclination_wyler(const inclination_wyler< iterator_type > & arg);
```

Parameters: `arg` wyler inclination sensor

40.
```
virtual void on_line_start_dn(const line_start_dn< iterator_type > & arg);
```

Parameters: `arg` start of a line scan in down direction

41.
```
virtual void
on_line_start_segment_1(const line_start_segment_1< iterator_type > & arg);
```

Parameters: `arg` start of the 1 segment of a scan line

42.
```
virtual void
on_line_start_segment_2(const line_start_segment_2< iterator_type > & arg);
```

Parameters: `arg` start of the 2 segment of a scan line

43.
```
virtual void
on_line_start_segment_3(const line_start_segment_3< iterator_type > & arg);
```

Parameters: `arg` start of the 3 segment of a scan line

44.
```
virtual void on_line_start_up(const line_start_up< iterator_type > & arg);
```

Parameters: `arg` start of a line scan in up direction

45.
```
virtual void on_line_stop(const line_stop< iterator_type > & arg);
```

Parameters: `arg` end of line scan

46.
```
virtual void on_meas_start(const meas_start< iterator_type > & arg);
```

Parameters: `arg` measurement has started

47.
```
virtual void on_meas_stop(const meas_stop< iterator_type > & arg);
```

Parameters: `arg` measurement has stopped

48.
```
virtual void on_monitoring_info(const monitoring_info< iterator_type > & arg);
```

Parameters: `arg` current setting of subdivider for monitoring data stream

49.
```
virtual void on_mta_settings(const mta_settings< iterator_type > & arg);
```

Parameters: `arg` default parameters for MTA processing

50.
```
virtual void on_mta_settings_1(const mta_settings_1< iterator_type > & arg);
```

Parameters: `arg` default parameters for MTA processing

51.
```
virtual void on_mta_settings_2(const mta_settings_2< iterator_type > & arg);
```

Parameters: `arg` default parameters for MTA processing

52.
```
virtual void on_pps_sync(const pps_sync< iterator_type > & arg);
```

Parameters: `arg` pulse per second, external time synchronisation

53.
```
virtual void on_pps_sync_ext(const pps_sync_ext< iterator_type > & arg);
```

Parameters: `arg` pulse per second, external time synchronisation

54.
```
virtual void on_pps_sync_hr(const pps_sync_hr< iterator_type > & arg);
```

Parameters: `arg` pulse per second, external time synchronisation

55.
```
virtual void on_pps_sync_hr_ext(const pps_sync_hr_ext< iterator_type > & arg);
```

Parameters: `arg` pulse per second, external time synchronisation

56.
```
virtual void on_pwm_sync(const pwm_sync< iterator_type > & arg);
```

Parameters: `arg` external pwm signal input

57.
```
virtual void on_scan_rect_fov(const scan_rect_fov< iterator_type > & arg);
```

Parameters: `arg` scan pattern description

58.
```
virtual void
on_scan_segments_fov(const scan_segments_fov< iterator_type > & arg);
```

Parameters: `arg` scan pattern description

59.
```
virtual void on_scanner_pose(const scanner_pose< iterator_type > & arg);
```

Parameters: `arg` scanner pose (position and orientation

60.
```
virtual void on_scanner_pose_hr(const scanner_pose_hr< iterator_type > & arg);
```

Parameters: `arg` scanner pose (position and orientation

61.
```
virtual void
on_scanner_pose_hr_1(const scanner_pose_hr_1< iterator_type > & arg);
```

Parameters: `arg` scanner pose (position and orientation

62.
```
virtual void
on_scanner_pose_ucs(const scanner_pose_ucs< iterator_type > & arg);
```

Parameters: `arg` scanner pose in user coordinate system (ucs);

63.
```
virtual void
on_scanner_pose_ucs_1(const scanner_pose_ucs_1< iterator_type > & arg);
```

Parameters: `arg` scanner pose in user coordinate system (ucs);

64.
```
virtual void on_units(const units< iterator_type > & arg);
```

Parameters: `arg` units information

65.
```
virtual void on_units_1(const units_1< iterator_type > & arg);
```

Parameters: `arg` units information

66.
```
virtual void on_units_2(const units_2< iterator_type > & arg);
```

Parameters: `arg` units information

67.
```
virtual void on_units_3(const units_3< iterator_type > & arg);
```

Parameters: `arg` units information

68.
```
virtual void on_units_IMU(const units_IMU< iterator_type > & arg);
```

Parameters: `arg` units information

69.
```
virtual void
on_unsolicited_message(const unsolicited_message< iterator_type > & arg);
```

Parameters: `arg` spontaneous information and error messages

70.
```
virtual void
on_unsolicited_message_1(const unsolicited_message_1< iterator_type > & arg);
```

Parameters: `arg` spontaneous information and error messages

71.
```
virtual void on_void_data(const void_data< iterator_type > & arg);
```

Parameters: `arg` empty helper package

72.
```
virtual void
on_unknown(unsigned id_main, unsigned id_sub, const iterator_type & begin,
           const iterator_type & end);
```

This function will be called for packets unknown to this version of the library.

Parameters: `id_main` the main id
`id_sub` the sub id

73.
```
void dont_handle_parent();
```

By default packets are dispatched to the most derived packet version known to the library and to all ancestors. If you dont want this behaviour you need to call this function.

74.
```
void request_dispatch_end();
```

This function will cause the dispatch function to return with a value of true.

Struct template beam_geometry

scanlib::beam_geometry — laser beam description

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct beam_geometry {

  enum @6 { id_main = = 3, id_sub = = 0 };

  // public data members
  float beam_exit_diameter;  // beam width at exit aperture [m]
  float beam_divergence;  // beam divergence in far field [rad]
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template biaxial_geometry

scanlib::biaxial_geometry — modelling biaxial shift of cog of range measurement in the near range

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct biaxial_geometry {
  // member classes/structs/unions

  // shift towards receiver aperture

  struct sequence_definition {

    // public data members
    float pivot;    // pivot value
    float value;    // shift in range of 0..1
  };

  enum @7 { id_main = = 76, id_sub = = 0 };

  enum @8 { biax_shift_max_size = = 16 };

  // public data members
  float aperture_direction;
  std::size_t biax_shift_size;
  sequence_definition biax_shift;
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct sequence_definition

scanlib::biaxial_geometry::sequence_definition — shift towards receiver aperture

Synopsis

```
// In header: <riegl/ridataspec.hpp>



// shift towards receiver aperture

struct sequence_definition {

  // public data members
  float pivot;  // pivot value
  float value;  // shift in range of 0..1
};
```

Struct template counter_sync

scanlib::counter_sync — external synchronization input

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct counter_sync {

  enum @9 { id_main = = 13, id_sub = = 0 };

  // public data members
  uint32_t systime;  // internal time stamp of external event in units of units.time_unit
  uint32_t count;  // number of events of the external event input
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template device_mounting

scanlib::device_mounting — geometrical parameters of external devices

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct device_mounting {
```

```
  enum @10 { id_main = = 30, id_sub = = 0 };

  // public data members
  float base_ofs;  // height of tripod base plate over soil or pavement [m]
  float gps_ofs;  // position of GPS antenna in z-axis direction in scanners coordinate sys-
tem [m]
  float soc_ofs;  // height of scanners coordinate system origin over tripod base plate [m]
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template extents

scanlib::extents — extents of various data fields

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct extents {

  enum @11 { id_main = = 57, id_sub = = 0 };

  // public data members
  float range_min;  // minimum and maximum possible
  float range_max;  // range in meter
  float amplitude_min;  // minimum and maximum possible
  float amplitude_max;  // amplitudes in dB
  float reflectance_min;  // minimum and maximum possible
  float reflectance_max;  // reflectance in dB
  uint16_t systime_bits;  // significant bits of systime stamp
  uint16_t backgnd_rad_min;  // minimum and maximum possible
  uint16_t backgnd_rad_max;  // background radiation
  int16_t dev_min;  // minimum and maximum possible
  int16_t dev_max;  // deviation from echo shape
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template frame_start_dn

scanlib::frame_start_dn — start of a scan frame in down direction.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct frame_start_dn {
```

```
  enum @12 { id_main = = 23, id_sub = = 0 };
};
```

Description

Sent, when a scan with changing frame angle enters the specified frame range in negative (decreasing angle) direction before data packets belonging to this frame

This package belongs to the predefined selectors:

protocol, scan

Struct template frame_start_up

scanlib::frame_start_up — Start of a scan frame in up direction.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct frame_start_up {

  enum @13 { id_main = = 22, id_sub = = 0 };
};
```

Description

Sent, when a scan with changing frame angle enters the specified frame range in positive (increasing angle) direction before data packets belonging to this frame

This package belongs to the predefined selectors:

protocol, scan

Struct template frame_stop

scanlib::frame_stop — end of a scan frame

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct frame_stop {

  enum @14 { id_main = = 15, id_sub = = 0 };
};
```

Description

after data packets belonging to this frame

This package belongs to the predefined selectors:

protocol, scan

Struct template header

scanlib::header — The mandatory header package.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct header {
  // member classes/structs/unions

  // id lookup table for short-id's

  struct sequence_definition {

    // public data members
    uint16_t sub;    // sub id
    uint16_t main;    // main id
  };

  enum @15 { id_main = = 1, id_sub = = 0 };

  enum @16 { id_lookup_max_size = = 254 };

  // public data members
  char type_id;  // instrument type
  char build;  // build variant
  char serial;  // serial number
  std::size_t id_lookup_size;
  sequence_definition id_lookup;
};
```

Description

Basic information to identify the scanning device and short id definition. Handling of the short IDs is mostly automatic, i.e. you do not normally need to care about the short ID mechanism.

This package belongs to the predefined selectors:

protocol

Struct sequence_definition

scanlib::header::sequence_definition — id lookup table for short-id's

Synopsis

```
// In header: <riegl/ridataspec.hpp>



// id lookup table for short-id's

struct sequence_definition {

  // public data members
  uint16_t sub;  // sub id
  uint16_t main;  // main id
```

```
};
```

Struct template header_ext

scanlib::header_ext — Extension header.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct header_ext {
  // member classes/structs/unions

  struct sequence_definition {

    // public data members
    uint16_t sub;    // sub id
    uint16_t main;    // main id
  };

  enum @17 { id_main = = 43, id_sub = = 0 };

  enum @18 { id_max_size = = 254 };

  // public data members
  std::size_t id_size;
  sequence_definition id;
};
```

Description

The extension header lists all major and minor numbers of packages which might be actual present in the current data stream

This package belongs to the predefined selectors:

protocol

Struct sequence_definition

scanlib::header_ext::sequence_definition

Synopsis

```
// In header: <riegl/ridataspec.hpp>



struct sequence_definition {

  // public data members
  uint16_t sub;  // sub id
  uint16_t main;  // main id
};
```

Struct template hk_bat

scanlib::hk_bat — power supply unit (24 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_bat {

  enum @19 { id_main = = 10003, id_sub = = 1 };

  // public data members
  int16_t V_IN_1;  // power supply unit, input voltage port 1 [10 mV]
  int16_t V_IN_2;  // power supply unit, input voltage port 2 [10 mV]
  int16_t V_IN_3;  // power supply unit, input voltage battery port [10 mV]
  uint16_t V_IN_SEL;  // power supply unit, selected input voltage [10 mV]
  uint16_t A_IN_SEL;  // power supply unit, input current [ 1 mA]
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  int16_t RESERVED_09;
  uint8_t RESERVED_10;
  uint8_t RESERVED_11;
  uint16_t RESERVED_20;
  uint16_t RESERVED_21;
  uint16_t RESERVED_22;
  uint16_t RESERVED_23;
  uint16_t RESERVED_24;
  uint16_t RESERVED_25;
  uint16_t RESERVED_26;
  uint16_t RESERVED_27;
  int16_t RESERVED_28;
  int16_t RESERVED_29;
  int16_t RESERVED_30;
  uint8_t level;  // battery unit, charge level [ 1 %]
  uint8_t RESERVED_32;
};
```

Description

This package belongs to the predefined selectors:

status, deprecated

Struct template hk_bat_1

scanlib::hk_bat_1 — power supply unit (24 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_bat_1 {

  enum @20 { id_main = = 10007, id_sub = = 1 };

  // public data members
  int16_t V_IN_1;  // power supply unit, input voltage port 1 [10 mV]
  int16_t V_IN_2;  // power supply unit, input voltage port 2 [10 mV]
  int16_t V_IN_3;  // power supply unit, input voltage battery port [10 mV]
```

```
  uint16_t V_IN_SEL;  // power supply unit, selected input voltage [10 mV]
  uint16_t A_IN_SEL;  // power supply unit, input current [ 1 mA]
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  int16_t RESERVED_09;
  uint16_t RESERVED_10;
  uint8_t RESERVED_11;  // battery unit (optionally attached) (22 Bytes)
  uint16_t RESERVED_20;
  uint16_t RESERVED_21;
  uint16_t RESERVED_22;
  uint16_t RESERVED_23;
  uint16_t RESERVED_24;
  uint16_t RESERVED_25;
  uint16_t RESERVED_26;
  uint16_t RESERVED_27;
  int16_t RESERVED_28;
  int16_t RESERVED_29;
  int16_t RESERVED_30;
  uint8_t level;  // battery unit, charge level [ 1 %]
  uint8_t RESERVED_32;
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_bat_2

scanlib::hk_bat_2 — power supply unit (24 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_bat_2 {

  enum @21 { id_main = = 10007, id_sub = = 2 };

  // public data members
  int16_t V_IN_1;  // power supply unit, input voltage port 1 [10 mV]
  int16_t V_IN_2;  // power supply unit, input voltage port 2 [10 mV]
  int16_t V_IN_3;  // power supply unit, input voltage battery port [10 mV]
  uint16_t V_IN_SEL;  // power supply unit, selected input voltage [10 mV]
  uint16_t A_IN_SEL;  // power supply unit, input current [ 1 mA]
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  int16_t RESERVED_09;
  uint16_t RESERVED_10;
  uint8_t RESERVED_11;  // battery unit (optionally attached) (22 Bytes)
  uint16_t RESERVED_20;
  uint16_t RESERVED_21;
  uint16_t RESERVED_22;
  uint16_t RESERVED_23;
  uint16_t RESERVED_24;
```

```
  uint16_t RESERVED_25;
  uint16_t RESERVED_26;
  uint16_t RESERVED_27;
  int16_t RESERVED_28;
  int16_t RESERVED_29;
  int16_t RESERVED_30;
  uint8_t level;  // battery unit, charge level [ 1 %]
  uint8_t RESERVED_32;
  uint8_t RESERVED_33;
  uint16_t RESERVED_34;
  uint32_t RESERVED_35;
  uint8_t RESERVED_36;
  uint8_t RESERVED_37;
  uint16_t RESERVED_38;
  uint16_t RESERVED_39;
  uint16_t RESERVED_40;
  uint16_t RESERVED_41;
  uint16_t RESERVED_42;
  uint16_t RESERVED_43;
  uint16_t RESERVED_44;
  uint16_t RESERVED_45;
  uint32_t RESERVED_46;
  uint32_t RESERVED_47;
  uint16_t RESERVED_48;
  uint16_t RESERVED_49;
  uint8_t RESERVED_50;
  uint16_t RESERVED_51;
  uint16_t RESERVED_52;
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ctr

scanlib::hk_ctr — control unit, i.e., mother board, including storage levels, info on attached equipment (60 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ctr {

  enum @22 { id_main = = 10004, id_sub = = 0 };

  // public data members
  int16_t RESERVED_00;
  uint16_t RESERVED_01;
  uint16_t RESERVED_02;
  uint16_t RESERVED_03;
  uint16_t RESERVED_04;
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  uint16_t RESERVED_09;
  uint16_t RESERVED_10;
```

```
  uint16_t RESERVED_11;
  uint8_t RESERVED_12;
  uint8_t LINK_STAT;  // control unit, link
  uint8_t WLAN_QUAL;  // control unit, wlan channel quality [ 1 %]
  uint8_t SCN_PROG;  // control unit, scan progress [ 1 %]
  float USED_FLASH;  // control unit, memory usage of int. compact flash [ 1 %]
  uint32_t TOTAL_FLASH;  // control unit, capacity of int. compact flash [ 1MB]
  float USED_USBM;  // control unit, memory usage of external USB device [ 1 %]
  uint32_t TOTAL_USBM;  // control unit, capacity of external USB device [ 1MB]
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ctr_1

scanlib::hk_ctr_1 — control unit, i.e., mother board, including storage levels, info on attached equipment (60 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ctr_1 {

  enum @23 { id_main = = 10004, id_sub = = 1 };

  // public data members
  int16_t RESERVED_00;
  uint16_t RESERVED_01;
  uint16_t RESERVED_02;
  uint16_t RESERVED_03;
  uint16_t RESERVED_04;
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  uint16_t RESERVED_09;
  uint16_t RESERVED_10;
  uint16_t RESERVED_11;
  uint8_t RESERVED_12;
  uint8_t LINK_STAT;  // control unit, link B0: usb memory detect (mass storage device) B1: -
usb camera detect B2: lan0 link B3: lan1 link
  uint8_t WLAN_QUAL;  // control unit, wlan channel quality [ 1 %]
  uint8_t SCN_PROG;  // control unit, scan progress [ 1 %]
  float USED_FLASH;  // control unit, memory usage of int. compact flash [ 1 %]
  uint32_t TOTAL_FLASH;  // control unit, capacity of int. compact flash [ 1MB]
  float USED_USBM;  // control unit, memory usage of external USB device [ 1 %]
  uint32_t TOTAL_USBM;  // control unit, capacity of external USB device [ 1MB]
  float MEAS_BUF_UTIL;  // control unit, measurement bufferutilization [ 1 %]
  float MON_BUF_UTIL;  // control unit, monitor measurement buffer utilization [ 1 %]
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps

scanlib::hk_gps — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps {

  enum @24 { id_main = = 10005, id_sub = = 0 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int32_t ECEFX;  // GPS data, ECEF X coordinate [ 1 cm].
  int32_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 cm].
  int32_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 cm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 cm].
  int32_t LONG;  // GPS data, longitude [1e-7deg].
  int32_t LAT;  // GPS data, latitude [1e-7deg].
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps_hr

scanlib::hk_gps_hr — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps_hr {

  enum @25 { id_main = = 10020, id_sub = = 0 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int64_t ECEFX;  // GPS data, ECEF X coordinate [ 1 mm].
  int64_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 mm].
  int64_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 mm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 mm].
  int64_t LONG;  // GPS data, longitude [1e-9deg].
  int64_t LAT;  // GPS data, latitude [1e-9deg].
```

```
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
  uint32_t systime;  // internal time in units of units.time_unit
  uint8_t SYNC_STATUS;  // PPS pulse and GPS data receive indicator bit 0: at least one -
valid gps information seen bit 1: valid gps information seen during last second bit 7: gps de-
tected, gps data received within last second.
  uint8_t SAT_NUM;  // GPS data, number of satelites used.
  uint16_t GEOM_DILUT;  // GPS data, geometric dilution of position [0.01].
  uint16_t TIME_DILUT;  // GPS data, time dilution [0.01].
  uint16_t VERT_DILUT;  // GPS data, vertical dilution (1D) [0.01].
  uint16_t HOR_DILUT;  // GPS data, horizontal dilution (2D) [0.01].
  uint16_t POS_DILUT;  // GPS data, position dilution (3D) [0.01].
  int64_t NORTHING;  // GPS data, northing [ 1 mm].
  int64_t EASTING;  // GPS data, easting [ 1 mm].
  int64_t HEIGHT;  // GPS data, height [ 1 mm].
  uint32_t FRAME_ANGLE;  // GPS data, frame angle, unit see units.frame_circle_count.
  uint32_t PPS_TIME;  // GPS data, internal time of PPS trigger edge in units of -
units.time_unit.
  uint32_t PPS_CNT;  // GPS data, counter of PPS, increments with each detected PPS trigger -
edge.
  uint32_t RCV_TIME;  // GPS data, internal time of receiving the time information in units -
of units.time_unit.
  uint32_t VALID_MASK;  // GPS data, valid mask.
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps_ts

scanlib::hk_gps_ts — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps_ts {

  enum @26 { id_main = = 10005, id_sub = = 1 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int32_t ECEFX;  // GPS data, ECEF X coordinate [ 1 cm].
  int32_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 cm].
  int32_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 cm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 cm].
  int32_t LONG;  // GPS data, longitude [1e-7deg].
  int32_t LAT;  // GPS data, latitude [1e-7deg].
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
```

```
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
  uint32_t systime;  // time stamp of PPS signal measured with internal clock in units of -
units.time_unit
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps_ts_status

scanlib::hk_gps_ts_status — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps_ts_status {

  enum @27 { id_main = = 10005, id_sub = = 2 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int32_t ECEFX;  // GPS data, ECEF X coordinate [ 1 cm].
  int32_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 cm].
  int32_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 cm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 cm].
  int32_t LONG;  // GPS data, longitude [1e-7deg].
  int32_t LAT;  // GPS data, latitude [1e-7deg].
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
  uint32_t systime;  // internal time in units of units.time_unit
  uint8_t SYNC_STATUS;  // PPS pulse and GPS data receive indicator bit 0: at least one valid
gps information seen bit 1: valid gps information seen during last second.
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps_ts_status_dop

scanlib::hk_gps_ts_status_dop — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps_ts_status_dop {

  enum @28 { id_main = = 10005, id_sub = = 3 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int32_t ECEFX;  // GPS data, ECEF X coordinate [ 1 cm].
  int32_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 cm].
  int32_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 cm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 cm].
  int32_t LONG;  // GPS data, longitude [1e-7deg].
  int32_t LAT;  // GPS data, latitude [1e-7deg].
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
  uint32_t systime;  // internal time in units of units.time_unit
  uint8_t SYNC_STATUS;  // PPS pulse and GPS data receive indicator bit 0: at least one -
valid gps information seen bit 1: valid gps information seen during last second bit 7: gps de-
tected, gps data received within last second.
  uint8_t SAT_NUM;  // GPS data, number of satelites used.
  uint16_t GEOM_DILUT;  // GPS data, geometric dilution of position [0.01].
  uint16_t TIME_DILUT;  // GPS data, time dilution [0.01].
  uint16_t VERT_DILUT;  // GPS data, vertical dilution (1D) [0.01].
  uint16_t HOR_DILUT;  // GPS data, horizontal dilution (2D) [0.01].
  uint16_t POS_DILUT;  // GPS data, position dilution (3D) [0.01].
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_gps_ts_status_dop_ucs

scanlib::hk_gps_ts_status_dop_ucs — GPS data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_gps_ts_status_dop_ucs {
```

```
  enum @29 { id_main = = 10005, id_sub = = 4 };

  // public data members
  uint32_t TOWms;  // GPS data, GPS Time of Week [ 1 ms].
  int32_t ECEFX;  // GPS data, ECEF X coordinate [ 1 cm].
  int32_t ECEFY;  // GPS data, ECEF Y coordinate [ 1 cm].
  int32_t ECEFZ;  // GPS data, ECEF Z coordinate [ 1 cm].
  uint32_t POS_ACC;  // GPS data, position accuracy estimate [ 1 cm].
  int32_t LONG;  // GPS data, longitude [1e-7deg].
  int32_t LAT;  // GPS data, latitude [1e-7deg].
  int32_t HGT_ELL;  // GPS data, height above ellipsoid [ 1 mm].
  int32_t HGT_SEA;  // GPS data, height above mean sea level [ 1 mm].
  uint32_t HOR_ACC;  // GPS data, horizontal accuracy estimate [ 1 mm].
  uint32_t VERT_ACC;  // GPS data, vertical accuracy estimate [ 1 mm].
  uint8_t STATUS1;  // GPS data, 0=no fix; 1=dead reckoning; 2=2D-fix; 3=3D-fix; 4=GPS+dead -
reckoning;5=Time only fix;255=undefined fix.
  uint8_t STATUS2;  // GPS data, 1 .. gpsfixOK; 2 .. diffSoln; 4 .. wknSet; 8 .. towSet.
  uint8_t STATUS3;  // GPS data, 0= no DGPS; 1=PR+PRR Correction; 2=PR+PRR+CP Correction; -
3=high accuracy PR+PRR+CP Correction.
  int8_t LEAP_SEC;  // GPS data, leap seconds (GPS-UTC) [ 1 s ].
  uint32_t systime;  // internal time in units of units.time_unit
  uint8_t SYNC_STATUS;  // PPS pulse and GPS data receive indicator bit 0: at least one -
valid gps information seen bit 1: valid gps information seen during last second bit 7: gps de-
tected, gps data received within last second.
  uint8_t SAT_NUM;  // GPS data, number of satelites used.
  uint16_t GEOM_DILUT;  // GPS data, geometric dilution of position [0.01].
  uint16_t TIME_DILUT;  // GPS data, time dilution [0.01].
  uint16_t VERT_DILUT;  // GPS data, vertical dilution (1D) [0.01].
  uint16_t HOR_DILUT;  // GPS data, horizontal dilution (2D) [0.01].
  uint16_t POS_DILUT;  // GPS data, position dilution (3D) [0.01].
  int64_t NORTHING;  // GPS data, northing [mm].
  int64_t EASTING;  // GPS data, easting [mm].
  int64_t HEIGHT;  // GPS data, height [mm].
  uint32_t FRAME_ANGLE;  // GPS data, frame angle, unit see units.frame_circle_count.
  uint32_t PPS_TIME;  // GPS data, internal time of PPS trigger edge in units of -
units.time_unit.
  uint32_t PPS_CNT;  // GPS data, counter of PPS, increments with each detected PPS trigger -
edge.
  uint32_t RCV_TIME;  // GPS data, internal time of receiving the time information in units -
of units.time_unit.
  uint32_t VALID_MASK;  // GPS data, valid mask.
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_incl

scanlib::hk_incl — inclination sensor

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_incl {

  enum @30 { id_main = = 10006, id_sub = = 0 };
```

```
  // public data members
  int16_t ROLL;  // inclination angle along x-axis [0.001 deg ]
  int16_t PITCH;  // inclination angle along y-axis [0.001 deg ]
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_incl_4axes

scanlib::hk_incl_4axes — inclination sensor

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_incl_4axes {

  enum @31 { id_main = = 10012, id_sub = = 0 };

  // public data members
  int32_t ROLL;  // inclination angle along x-axis [0.001 deg ]
  int32_t PITCH;  // inclination angle along y-axis [0.001 deg ]
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ph_data

scanlib::hk_ph_data — protective housing data

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ph_data {

  enum @32 { id_main = = 10014, id_sub = = 0 };

  // public data members
  uint16_t Version_Major;  // major version ID of protective housing firmware
  uint16_t Version_Minor;  // minor version ID of protective housing firmware
  uint16_t Version_Bugfix;  // bugfix ID of protective housing firmware
  uint16_t Version_Build;  // build ID of protective housing firmware
  uint16_t Date_Year;  // Year of protective housing internal RTC.
  uint8_t Date_Month;  // Month of protective housing internal RTC.
  uint8_t Date_Day;  // Day of protective housing internal RTC.
  uint8_t Time_Seconds;  // Seconds of protective housing internal RTC.
```

```
  uint8_t Time_Minutes;  // Minutes of protective housing internal RTC.
  uint8_t Time_Hours;  // Hours of protective housing internal RTC.
  uint32_t systime;  // [systime] = hk_ph_units.systime
  uint32_t OnTime;  // OnTime[0]: time since power up in seconds OnTime[1]: total operating -
time in seconds.
  uint16_t pwr_24V;  // [pwr_24V] = hk_ph_units.voltage
  uint16_t pwr_I_24V;  // [pwr_I_24V] = hk_ph_units.current
  uint32_t pwr_P_24V;  // [pwr_P_24V] = hk_ph_units.power
  uint16_t pwr_U_TECp;  // [pwr_U_TECp] = hk_ph_units.voltage
  uint16_t pwr_U_TECm;  // [pwr_U_TECm] = hk_ph_units.voltage
  uint16_t uc_24V;  // [uc_24V] = hk_ph_units.voltage
  uint16_t uc_P_24V;  // [uc_P_24V] = hk_ph_units.power
  uint16_t uc_I_24V;  // [pwr_I_24V] = hk_ph_units.current
  uint16_t uc_12V;  // [uc_12V] = hk_ph_units.voltage
  uint16_t uc_5V;  // [uc_5V] = hk_ph_units.voltage
  uint16_t uc_3V3;  // [uc_3V3] = hk_ph_units.voltage
  int16_t uc_Proc_T;  // [uc_Proc_T] = hk_ph_units.temp
  int16_t uc_Temp;  // [uc_Temp] = hk_ph_units.temp
  uint16_t uc_RH;  // [uc_RH] = hk_ph_units.rh
  int16_t uc_TD;  // [uc_TD[n]] = hk_ph_units.temp
  int16_t uc_TDq;  // [uc_TDq] = hk_ph_units.temp
  int16_t uc_Tsoll;  // [uc_Tsoll] = hk_ph_units.temp
  uint16_t uc_I_FAN_C1;  // [uc_I_FAN_C1]= hk_ph_units.current
  uint16_t uc_I_FAN_C2;  // [uc_I_FAN_C1]= hk_ph_units.current
  uint16_t uc_I_FAN_TI;  // [uc_I_FAN_TI]= hk_ph_units.current
  uint16_t uc_I_FAN_TO;  // [uc_I_FAN_TO]= hk_ph_units.current
  uint16_t uc_FAN_SPEED;  // [uc_FAN_SPEED[n]] = RPM
  uint32_t uc_PWM_F;  // [uc_PWM_F] = Hz
  uint8_t uc_DCCq;  // [uc_DCCq] = hk_ph_units.dc
  uint8_t uc_DCHq;  // [uc_DCHq] = hk_ph_units.dc
  uint16_t uc_DACValue;  // [uc_DACValue]= 1
  uint16_t uc_Status;  // Statusinformationen 2 Oktetts.
  uint32_t uc_Error;  // Errorinformationen 4 Oktetts.
  uint16_t Reserved0;
  uint16_t Reserved1;
  uint16_t Reserved2;
  uint16_t Reserved3;
  uint16_t Reserved4;
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ph_data_1

scanlib::hk_ph_data_1 — protective housing data

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ph_data_1 {

  enum @33 { id_main = = 10014, id_sub = = 1 };

  // public data members
  uint16_t Version_Major;  // major version ID of protective housing firmware
```

```
  uint16_t Version_Minor;  // minor version ID of protective housing firmware
  uint16_t Version_Bugfix;  // bugfix ID of protective housing firmware
  uint16_t Version_Build;  // build ID of protective housing firmware
  uint16_t Date_Year;  // Year of protective housing internal RTC.
  uint8_t Date_Month;  // Month of protective housing internal RTC.
  uint8_t Date_Day;  // Day of protective housing internal RTC.
  uint8_t Time_Seconds;  // Seconds of protective housing internal RTC.
  uint8_t Time_Minutes;  // Minutes of protective housing internal RTC.
  uint8_t Time_Hours;  // Hours of protective housing internal RTC.
  uint32_t systime;  // [systime] = hk_ph_units.systime
  uint32_t OnTime;  // OnTime[0]: time since power up in seconds OnTime[1]: total operating -
time in seconds.
  uint16_t pwr_24V;  // [pwr_24V] = hk_ph_units.voltage
  uint16_t pwr_I_24V;  // [pwr_I_24V] = hk_ph_units.current
  uint32_t pwr_P_24V;  // [pwr_P_24V] = hk_ph_units.power
  uint16_t pwr_U_TECp;  // [pwr_U_TECp] = hk_ph_units.voltage
  uint16_t pwr_U_TECm;  // [pwr_U_TECm] = hk_ph_units.voltage
  uint16_t uc_24V;  // [uc_24V] = hk_ph_units.voltage
  uint16_t uc_P_24V;  // [uc_P_24V] = hk_ph_units.power
  uint16_t uc_I_24V;  // [pwr_I_24V] = hk_ph_units.current
  uint16_t uc_12V;  // [uc_12V] = hk_ph_units.voltage
  uint16_t uc_5V;  // [uc_5V] = hk_ph_units.voltage
  uint16_t uc_3V3;  // [uc_3V3] = hk_ph_units.voltage
  int16_t uc_Proc_T;  // [uc_Proc_T] = hk_ph_units.temp
  int16_t uc_Temp;  // [uc_Temp] = hk_ph_units.temp
  uint16_t uc_RH;  // [uc_RH] = hk_ph_units.rh
  int16_t uc_TD;  // [uc_TD[n]] = hk_ph_units.temp
  int16_t uc_TDq;  // [uc_TDq] = hk_ph_units.temp
  int16_t uc_Tsoll;  // [uc_Tsoll] = hk_ph_units.temp
  uint16_t uc_I_FAN_C1;  // [uc_I_FAN_C1]= hk_ph_units.current
  uint16_t uc_I_FAN_C2;  // [uc_I_FAN_C1]= hk_ph_units.current
  uint16_t uc_I_FAN_TI;  // [uc_I_FAN_TI]= hk_ph_units.current
  uint16_t uc_I_FAN_TO;  // [uc_I_FAN_TO]= hk_ph_units.current
  uint16_t uc_FAN_SPEED;  // [uc_FAN_SPEED[n]] = RPM
  uint32_t uc_PWM_F;  // [uc_PWM_F] = Hz
  uint8_t uc_DCCq;  // [uc_DCCq] = hk_ph_units.dc
  uint8_t uc_DCHq;  // [uc_DCHq] = hk_ph_units.dc
  uint16_t uc_DACValue;  // [uc_DACValue]= 1
  uint16_t uc_Status;  // Statusinformationen 2 Oktetts.
  uint32_t uc_Error;  // Errorinformationen 4 Oktetts.
  uint16_t Reserved0;
  uint16_t Reserved1;
  uint16_t Reserved2;
  uint16_t Reserved3;
  uint16_t Reserved4;
  int16_t uc_Accl_X_mean;  // [uc_Accl_X_mean] = hk_ph_units.accl_unit
  int16_t uc_Accl_X_min;  // [uc_Accl_X_min] = hk_ph_units.accl_unit
  int16_t uc_Accl_X_max;  // [uc_Accl_X_max] = hk_ph_units.accl_unit
  uint16_t uc_Accl_X_cnt;  // [uc_Accl_X_cnt] = 1
  int16_t uc_Accl_Y_mean;  // [uc_Accl_Y_mean] = hk_ph_units.accl_unit
  int16_t uc_Accl_Y_min;  // [uc_Accl_Y_min] = hk_ph_units.accl_unit
  int16_t uc_Accl_Y_max;  // [uc_Accl_Y_max] = hk_ph_units.accl_unit
  uint16_t uc_Accl_Y_cnt;  // [uc_Accl_X_cnt] = 1
  int16_t uc_Accl_Z_mean;  // [uc_Accl_Z_mean] = hk_ph_units.accl_unit
  int16_t uc_Accl_Z_min;  // [uc_Accl_Z_min] = hk_ph_units.accl_unit
  int16_t uc_Accl_Z_max;  // [uc_Accl_Z_max] = hk_ph_units.accl_unit
  uint16_t uc_Accl_Z_cnt;  // [uc_Accl_X_cnt] = 1*
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ph_units

scanlib::hk_ph_units — protective housing units

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ph_units {

  enum @34 { id_main = = 10013, id_sub = = 0 };

  // public data members
  float voltage;  // nominally 0.001V
  float current;  // nominally 0.001A
  float power;  // nominally 0.001W
  float temp;  // nominally 0.01°C
  float rh;  // nominally %
  float dc;  // nominally %
  float systime;  // nominally 12.5ns
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_ph_units_1

scanlib::hk_ph_units_1 — protective housing units

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_ph_units_1 {

  enum @35 { id_main = = 10013, id_sub = = 1 };

  // public data members
  float voltage;  // nominally 0.001V
  float current;  // nominally 0.001A
  float power;  // nominally 0.001W
  float temp;  // nominally 0.01°C
  float rh;  // nominally %
  float dc;  // nominally %
  float systime;  // nominally 12.5ns
  float accl_unit;  // nominally 0.001g
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_pwr

scanlib::hk_pwr — power supply unit (24 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_pwr {

  enum @36 { id_main = = 10003, id_sub = = 0 };

  // public data members
  int16_t V_IN_1;  // power supply unit, input voltage port 1 [10 mV]
  int16_t V_IN_2;  // power supply unit, input voltage port 2 [10 mV]
  int16_t V_IN_3;  // power supply unit, input voltage battery port [10 mV]
  uint16_t V_IN_SEL;  // power supply unit, selected input voltage [10 mV]
  uint16_t A_IN_SEL;  // power supply unit, input current [ 1 mA]
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  int16_t RESERVED_09;
  uint8_t RESERVED_10;
  uint8_t RESERVED_11;
};
```

Description

This package belongs to the predefined selectors:

status, deprecated

Struct template hk_pwr_1

scanlib::hk_pwr_1 — power supply unit (24 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_pwr_1 {

  enum @37 { id_main = = 10007, id_sub = = 0 };

  // public data members
  int16_t V_IN_1;  // power supply unit, input voltage port 1 [10 mV]
  int16_t V_IN_2;  // power supply unit, input voltage port 2 [10 mV]
  int16_t V_IN_3;  // power supply unit, input voltage battery port [10 mV]
  uint16_t V_IN_SEL;  // power supply unit, selected input voltage [10 mV]
  uint16_t A_IN_SEL;  // power supply unit, input current [ 1 mA]
  uint16_t RESERVED_05;
  uint16_t RESERVED_06;
  uint16_t RESERVED_07;
  uint16_t RESERVED_08;
  int16_t RESERVED_09;
  uint16_t RESERVED_10;
  uint8_t RESERVED_11;
```

```
};
```

Description

This package belongs to the predefined selectors:

status

Struct template hk_rtc

scanlib::hk_rtc — built in real time clock of scanning device, local time

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_rtc {

  enum @38 { id_main = = 10000, id_sub = = 0 };

  // public data members
  uint8_t year;  // years since 1900
  uint8_t month;  // months since January
  uint8_t day;  // day of the month
  uint8_t hour;  // hours after midnight
  uint8_t minute;  // minutes after the hour
  uint8_t second;  // seconds after the minute
};
```

Description

This package belongs to the predefined selectors:

status, legacy

Struct template hk_rtc_sys

scanlib::hk_rtc_sys — real time clock, local time (10 Bytes)

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct hk_rtc_sys {

  enum @39 { id_main = = 10000, id_sub = = 1 };

  // public data members
  uint8_t year;  // years since 1900
  uint8_t month;  // months since January
  uint8_t day;  // day of the month
  uint8_t hour;  // hours after midnight
  uint8_t minute;  // minutes after the hour
  uint8_t second;  // seconds after the minute
  uint32_t systime;  // internal time in units of units.time_unit
};
```

Description

This package belongs to the predefined selectors:

status, legacy

Struct template IMU_data

scanlib::IMU_data — IMU raw data.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct IMU_data {

  enum @0 { id_main = = 61, id_sub = = 0 };

  // public data members
  uint32_t systime;  // time stamp of IMU data in units of time_unit
  int16_t x_gyro;  // gyro raw data x-axis in units of gyro_unit
  int16_t y_gyro;  // gyro raw data y-axis ..
  int16_t z_gyro;  // gyro raw data z-axis ..
  int16_t x_accl;  // accelerometer x-axis in units of accl_unit
  int16_t y_accl;  // accelerometer y-axis ..
  int16_t z_accl;  // accelerometer z-axis ..
};
```

Description

This package belongs to the predefined selectors:

data

Struct template inclination_wyler

scanlib::inclination_wyler — wyler inclination sensor

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct inclination_wyler {

  enum @40 { id_main = = 79, id_sub = = 0 };

  // public data members
  uint32_t systime;  // approximate systime when the sample was taken in units of -
uints.time_unit
  float angle_axis_0;  // angle of first axis measured in degrees
  float angle_axis_1;  // angle of second axis measured in degrees
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template line_start_dn

scanlib::line_start_dn — start of a line scan in down direction

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct line_start_dn {

  enum @41 { id_main = = 25, id_sub = = 0 };
};
```

Description

before data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template line_start_segment_1

scanlib::line_start_segment_1 — start of the 1 segment of a scan line

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct line_start_segment_1 {

  enum @42 { id_main = = 99, id_sub = = 0 };
};
```

Description

before data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template line_start_segment_2

scanlib::line_start_segment_2 — start of the 2 segment of a scan line

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
```

```
struct line_start_segment_2 {

  enum @43 { id_main = = 100, id_sub = = 0 };
};
```

Description

before data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template line_start_segment_3

scanlib::line_start_segment_3 — start of the 3 segment of a scan line

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct line_start_segment_3 {

  enum @44 { id_main = = 101, id_sub = = 0 };
};
```

Description

before data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template line_start_up

scanlib::line_start_up — start of a line scan in up direction

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct line_start_up {

  enum @45 { id_main = = 24, id_sub = = 0 };
};
```

Description

before data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template line_stop

scanlib::line_stop — end of line scan

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct line_stop {

  enum @46 { id_main = = 17, id_sub = = 0 };
};
```

Description

sent, when a scan with changing frame angle escapes the specified frame range after data packets belonging to this line

This package belongs to the predefined selectors:

protocol, scan

Struct template meas_start

scanlib::meas_start — measurement has started

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct meas_start {

  enum @47 { id_main = = 31, id_sub = = 0 };
};
```

Description

This package belongs to the predefined selectors:

protocol, scan

Struct template meas_stop

scanlib::meas_stop — measurement has stopped

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct meas_stop {

  enum @48 { id_main = = 32, id_sub = = 0 };
};
```

Description

This package belongs to the predefined selectors:

protocol, scan

Struct template monitoring_info

scanlib::monitoring_info — current setting of subdivider for monitoring data stream

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct monitoring_info {

  enum @49 { id_main = = 39, id_sub = = 0 };

  // public data members
  uint16_t msm_line;  // subdivider for measurements within scan line
  uint16_t msm_frame;  // subdivider for scan lines within frames
};
```

Description

For explanation please read the user manual of scanner device

This package belongs to the predefined selectors:

attribute, scan

Struct template mta_settings

scanlib::mta_settings — default parameters for MTA processing

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct mta_settings {

  enum @50 { id_main = = 74, id_sub = = 0 };

  // public data members
  uint8_t zone;  // user requested MTA zone for data analysis.
  float gate_low;  // lower range gate in meters, valid if zone == 0
  float gate_high;  // upper range gate in meters, valid if zone == 0
  float zone_width;  // width of the multiple time around zones in meter
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument, scan

Struct template mta_settings_1

scanlib::mta_settings_1 — default parameters for MTA processing

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct mta_settings_1 {

  enum @51 { id_main = = 81, id_sub = = 0 };

  // public data members
  uint8_t zone;  // user requested MTA zone for data analysis.
  float gate_low;  // lower range gate in meters, valid if zone == 0
  float gate_high;  // upper range gate in meters, valid if zone == 0
  float zone_width;  // width of the multiple time around zones in meter
  float modulation_depth;  // modulation of shot phase in meter
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument, scan

Struct template mta_settings_2

scanlib::mta_settings_2 — default parameters for MTA processing

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct mta_settings_2 {

  enum @52 { id_main = = 81, id_sub = = 1 };

  // public data members
  uint8_t zone;  // user requested MTA zone for data analysis.
  float gate_low;  // lower range gate in meters, valid if zone == 0
  float gate_high;  // upper range gate in meters, valid if zone == 0
  float zone_width;  // width of the multiple time around zones in meter
  float modulation_depth_code;  // depth of pulse position modulation in meter
  float modulation_depth_dithering;  // depth of dithering in meter
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument, scan

Class ostream_packets

scanlib::ostream_packets — convert to string representation

Synopsis

```
// In header: <riegl/ridataspec.hpp>


class ostream_packets : public scanlib::basic_packets {
public:
  // types
  typedef basic_packets::iterator_type iterator_type;  // embedded iterator type definition

  // construct/copy/destruct
  ostream_packets(std::ostream &);
  ~ostream_packets();
};
```

Description

This class converts a package into a string representation. By default all packets are converted. The selector can be used to filter out the intended subset of packets. Dont forget to specify the header packet in the selector!

`ostream_packets` public construct/copy/destruct

1. ```
   ostream_packets(std::ostream & out);
   ```

   construct with stream reference

   Parameters: `out` a reference to an out stream

2. ```
   ~ostream_packets();
   ```

Struct package_id

scanlib::package_id — enumerated package identifiers

Synopsis

```
// In header: <riegl/ridataspec.hpp>


struct package_id {

  enum type { unknown = = 0, IMU_data = = 1, atmosphere = = 4,
              atmosphere_1 = = 5, atmosphere_2 = = 6, atmosphere_3 = = 7,
              atmosphere_4 = = 8, beam_geometry = = 10,
              biaxial_geometry = = 11, counter_sync = = 22,
              device_mounting = = 32, extents = = 36, frame_start_dn = = 43,
              frame_start_up = = 44, frame_stop = = 45, header = = 47,
              header_ext = = 48, hk_bat = = 49, hk_bat_1 = = 50,
              hk_bat_2 = = 51, hk_ctr = = 53, hk_ctr_1 = = 54, hk_gps = = 58,
              hk_gps_hr = = 59, hk_gps_ts = = 60, hk_gps_ts_status = = 61,
              hk_gps_ts_status_dop = = 62, hk_gps_ts_status_dop_ucs = = 63,
              hk_incl = = 65, hk_incl_4axes = = 66, hk_ph_data = = 68,
              hk_ph_data_1 = = 69, hk_ph_units = = 70, hk_ph_units_1 = = 71,
              hk_pwr = = 72, hk_pwr_1 = = 73, hk_rtc = = 86,
              hk_rtc_sys = = 87, inclination_wyler = = 99,
              line_start_dn = = 113, line_start_segment_1 = = 114,
              line_start_segment_2 = = 115, line_start_segment_3 = = 116,
              line_start_up = = 117, line_stop = = 118, meas_start = = 121,
```

```
              meas_stop = = 122, monitoring_info = = 123,
              mta_settings = = 124, mta_settings_1 = = 125,
              mta_settings_2 = = 126, pps_sync = = 145, pps_sync_ext = = 146,
              pps_sync_hr = = 147, pps_sync_hr_ext = = 148, pwm_sync = = 150,
              scan_rect_fov = = 174, scan_segments_fov = = 175,
              scanner_pose = = 176, scanner_pose_hr = = 177,
              scanner_pose_hr_1 = = 178, scanner_pose_ucs = = 179,
              scanner_pose_ucs_1 = = 180, units = = 190, units_1 = = 191,
              units_2 = = 192, units_3 = = 193, units_IMU = = 194,
              unsolicited_message = = 195, unsolicited_message_1 = = 196,
              void_data = = 198 };
  // construct/copy/destruct
  package_id(package_id::type);
  package_id(unsigned, unsigned);
  package_id(const char *);
  template<typename P> explicit package_id(const P &);
  package_id();
  package_id(const std::string &);

  // public member functions
  operator type() const;
  std::string string() const;

  // public data members
  unsigned main;  // external main id
  unsigned sub;  // external sub id
};
```

Description

Packets can be refered by string name, by major and minor numbers, and by an C++ enumeration. This class ties together these various identifiers. While the string names and major, minor numbers are unique outside the RiVLib context, the enumerations are only valid in the context of a particular library version. Consequently if you want to externally store a packet id, either string name or major and minor numbers should be used. The enumerated values on the other hand consist of a contiguous range and therefore can be used as an index into an array. Please note however, that the documentation does not list the entire set of values, the internal packets are omitted. This means, that you cannot infer the numerical value of the enumeration from its listed position in the documentation. The enumerations are used as the index into the selector bitset that can be set for packet dispatching.

`package_id` public construct/copy/destruct

1. ```
   package_id(package_id::type type_id);
   ```

   constructor from type

   Construct a packet_id type enumeration.
   Parameters: `type_id` package type identifier

2. ```
   package_id(unsigned main_, unsigned sub_);
   ```

   constructor from id's

   Construct a packet_id from externally unique main and sub id's Will map to unknown resp. main=0, sub=0 if requesting undefined packet.
   Parameters: `main_` main packet id
   `sub_` sub packet id

3. ```
   package_id(const char * name);
   ```

   constructor from string name
   ```

Construct a packet_id from externally unique packet name Will map to unknown resp. main=0, sub=0 if requesting undefined packet.
Parameters: `name` string name of packet

4. ```
   template<typename P> explicit package_id(const P & p);
   ```

   constructor from packet reference

   Construct a packet_id from a packet instance
   Parameters: `p` packet instance

5. ```
   package_id();
   ```

   default constructor

   Construct a id of invalid type

6. ```
   package_id(const std::string & name);
   ```

   constructor from string name

   Construct a packet_id from externally unique packet name Will map to unknown resp. main=0, sub=0 if requesting undefined packet.
   Parameters: `name` string name of packet

`package_id` public member functions

1. ```
   operator type() const;
   ```

   convert to enumeration

   This operator converts the external id's into enumeration format.

2. ```
   std::string string() const;
   ```

   get string representation

   This function converts the external id's into its string name.

Struct template pps_sync

scanlib::pps_sync — pulse per second, external time synchronisation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct pps_sync {

  enum @53 { id_main = = 12, id_sub = = 0 };

  // public data members
  uint32_t systime;  // internal time of external PPS pulse in units of units.time_uint
  uint32_t pps;  // absolute time of pps pulse since begin of week or start of day [msec]
```
```

```
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template pps_sync_ext

scanlib::pps_sync_ext — pulse per second, external time synchronisation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct pps_sync_ext {

  enum @54 { id_main = = 12, id_sub = = 1 };

  // public data members
  uint32_t systime;  // internal time of external PPS pulse in units of units.time_unit
  uint32_t pps;  // absolute time of pps pulse since begin of week or start of day [msec]
  uint16_t year;  // UTC year in units of year.
  uint8_t mon;  // UTC month, 1..12 in units of mon.
  uint8_t day;  // UTC day of month, 1..31 in units of day.
  uint8_t hour;  // UTC hour of day, 0 .. 23 in units of hour.
  uint8_t min;  // UTC minute of hour, 0 .. 59 in units of min.
  uint8_t sec;  // UTC second of minute, 0 .. 59 in units of sec.
  uint8_t flags;  // B0 .. Time of week valid; B1 .. valid week number; B2 .. leap seconds -
known.
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template pps_sync_hr

scanlib::pps_sync_hr — pulse per second, external time synchronisation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct pps_sync_hr {

  enum @55 { id_main = = 70, id_sub = = 0 };

  // public data members
  uint64_t systime;  // internal time of external PPS pulse in units of -
units.time_unit_hi_prec
  uint64_t pps;  // absolute time of pps pulse since begin of week or start of day in usec
```

```
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template pps_sync_hr_ext

scanlib::pps_sync_hr_ext — pulse per second, external time synchronisation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct pps_sync_hr_ext {

  enum @56 { id_main = = 70, id_sub = = 1 };

  // public data members
  uint64_t systime;  // internal time of external PPS pulse in units of -
units.time_unit_hi_prec
  uint64_t pps;  // absolute time of pps pulse since begin of week or start of day in usec
  uint16_t year;  // UTC year in units of year.
  uint8_t mon;  // UTC month, 1..12 in units of mon.
  uint8_t day;  // UTC day of month, 1..31 in units of day.
  uint8_t hour;  // UTC hour of day, 0 .. 23 in units of hour.
  uint8_t min;  // UTC minute of hour, 0 .. 59 in units of min.
  uint8_t sec;  // UTC second of minute, 0 .. 59 in units of sec.
  uint8_t flags;  // B0 .. Time of week valid; B1 .. valid week number; B2 .. leap seconds -
known.
};
```

Description

This package belongs to the predefined selectors:

data, scan

Struct template pwm_sync

scanlib::pwm_sync — external pwm signal input

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct pwm_sync {

  enum @57 { id_main = = 97, id_sub = = 0 };

  // public data members
  uint32_t systime;
  uint32_t pulseWidth;
```

```
  uint8_t polarity;
  uint32_t pulseCounter;
};
```

Description

This package belongs to the predefined selectors:

status

Struct template scan_rect_fov

scanlib::scan_rect_fov — scan pattern description

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scan_rect_fov {

  enum @58 { id_main = = 5, id_sub = = 0 };

  // public data members
  float theta_min;  // minimum of scan angle along line scan axis [deg]
  float theta_max;  // maximum of scan angle along line scan axis [deg]
  float theta_incr;  // nominal scan angle increment along line scan axis [deg]
  float phi_min;  // minimum of scan angle along frame scan axis [deg]
  float phi_max;  // maximum of scan angle along frame scan axis [deg]
  float phi_incr;  // nominal scan angle increment along frame scan axis [deg]
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template scan_segments_fov

scanlib::scan_segments_fov — scan pattern description

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scan_segments_fov {
  // member classes/structs/unions

  struct sequence_definition {

    // public data members
    float min;    // starting scan angle for this segemnt in degrees
    float max;    // ending scan angle for this segment in degrees
    float incr;    // nominal scan increment for this segment in degrees
  };
```

```
  enum @59 { id_main = = 105, id_sub = = 0 };

  enum @60 { angles_max_size = = 3 };

  // public data members
  std::size_t angles_size;
  sequence_definition angles;
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct sequence_definition

scanlib::scan_segments_fov::sequence_definition

Synopsis

```
// In header: <riegl/ridataspec.hpp>



struct sequence_definition {

  // public data members
  float min;  // starting scan angle for this segemnt in degrees
  float max;  // ending scan angle for this segment in degrees
  float incr;  // nominal scan increment for this segment in degrees
};
```

Struct template scanner_pose

scanlib::scanner_pose — scanner pose (position and orientation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scanner_pose {

  enum @61 { id_main = = 54, id_sub = = 0 };

  // public data members
  float LAT;  // Latitude [deg].
  float LON;  // Longitude [deg].
  float HEIGHT;  // height above ellipsoid [m]
  float HMSL;  // height above mean sea level [m]
  float roll;  // roll angle about scanner x-axis [deg]
  float pitch;  // pitch angle about scanner y-axis [deg]
  float yaw;  // yaw angle about scanner z-axis [deg]
  float hAcc;  // horizontal accuracy [m]
  float vAcc;  // vertical accuracy [m]
  float rAcc;  // roll angle accuracy [deg]
  float pAcc;  // pitch angle accuracy [deg]
```

```
  float yAcc;  // yaw angle accuracy [deg]
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template scanner_pose_hr

scanlib::scanner_pose_hr — scanner pose (position and orientation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scanner_pose_hr {

  enum @62 { id_main = = 72, id_sub = = 0 };

  // public data members
  double LAT;  // Latitude [deg].
  double LON;  // Longitude [deg].
  double HEIGHT;  // height above ellipsoid [m]
  double HMSL;  // height above mean sea level [m]
  float roll;  // roll angle about scanner x-axis [deg]
  float pitch;  // pitch angle about scanner y-axis [deg]
  float yaw;  // yaw angle about scanner z-axis [deg]
  float hAcc;  // horizontal accuracy [m]
  float vAcc;  // vertical accuracy [m]
  float rAcc;  // roll angle accuracy [deg]
  float pAcc;  // pitch angle accuracy [deg]
  float yAcc;  // yaw angle accuracy [deg]
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template scanner_pose_hr_1

scanlib::scanner_pose_hr_1 — scanner pose (position and orientation

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scanner_pose_hr_1 {

  enum @63 { id_main = = 72, id_sub = = 1 };

  // public data members
  double LAT;  // Latitude in deg.
```

```
  double LON;  // Longitude in deg.
  double HEIGHT;  // height above ellipsoid in m
  double HMSL;  // height above mean sea level in m
  float roll;  // roll angle about scanner x-axis in deg
  float pitch;  // pitch angle about scanner y-axis in deg
  float yaw;  // yaw angle about scanner z-axis in deg
  float hAcc;  // horizontal accuracy in m
  float vAcc;  // vertical accuracy in m
  float rAcc;  // roll angle accuracy in deg
  float pAcc;  // pitch angle accuracy in deg
  float yAcc;  // yaw angle accuracy in deg
  float height_baro;  // altitude determined based on the atmospheric pressure according to -
the standard atmosphere laws in m.
};
```

Description

This package belongs to the predefined selectors:

attribute, scan

Struct template scanner_pose_ucs

scanlib::scanner_pose_ucs — scanner pose in user coordinate system (ucs);

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scanner_pose_ucs {

  enum @64 { id_main = = 71, id_sub = = 0 };

  // public data members
  char ucs_name;  // coordinate system name, zero terminated string
  char proj_name;  // projection name, zero terminated string
  float scale_factor;  // scale factor (just for information)
  double northing;  // "northing" in units of m
  double easting;  // "easting" in units of m
  double height;  // height in units of m
  float roll;  // roll angle about scanner x-axis in units of deg
  float pitch;  // pitch angle about scanner y-axis in units of deg
  float yaw;  // yaw angle about scanner z-axis in units of deg
  float hAcc;  // horizontal accuracy in units of m
  float vAcc;  // vertical accuracy in units of m
  float rAcc;  // roll angle accuracy in units of deg
  float pAcc;  // pitch angle accuracy in units of deg
  float yAcc;  // yaw angle accuracy in units of deg
};
```

Description

the user coordinate system is a leveled coordinate system rotated about the height axis by a certain well-known amount alpha; positive angles alpha rotate the ucs counter-clockwise altough in rotated ucs the wording "northing" and "easting " is not precise, the terms are still used

This package belongs to the predefined selectors:

attribute, scan

Struct template scanner_pose_ucs_1

scanlib::scanner_pose_ucs_1 — scanner pose in user coordinate system (ucs);

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct scanner_pose_ucs_1 {

  enum @65 { id_main = = 71, id_sub = = 1 };

  // public data members
  char ucs_name;  // coordinate system name, zero terminated string
  char proj_name;  // projection name, zero terminated string
  float scale_factor;  // scale factor (just for information)
  double northing;  // "northing" in units of m
  double easting;  // "easting" in units of m
  double height;  // height in units of m
  float roll;  // roll angle about scanner x-axis in units of deg
  float pitch;  // pitch angle about scanner y-axis in units of deg
  float yaw;  // yaw angle about scanner z-axis in units of deg
  float hAcc;  // horizontal accuracy in units of m
  float vAcc;  // vertical accuracy in units of m
  float rAcc;  // roll angle accuracy in units of deg
  float pAcc;  // pitch angle accuracy in units of deg
  float yAcc;  // yaw angle accuracy in units of deg
  float height_baro;  // altitude determined based on the atmospheric pressure according to -
the standard atmosphere laws in m.
};
```

Description

the user coordinate system is a leveled coordinate system rotated about the height axis by a certain well-known amount alpha; positive angles alpha rotate the ucs counter-clockwise altough in rotated ucs the wording "northing" and "easting " is not precise, the terms are still used

This package belongs to the predefined selectors:

attribute, scan

Struct template units

scanlib::units — units information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct units {

  enum @66 { id_main = = 2, id_sub = = 0 };

  // public data members
  float range_unit;  // length of 1 LSB of range in meter
  uint32_t line_circle_count;  // number of LSBs per full rotation about line axis
  uint32_t frame_circle_count;  // number of LSBs per full rotation about frame axis
```

```
  float time_unit;  // duration of 1 LSB of time in seconds
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template units_1

scanlib::units_1 — units information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct units_1 {

  enum @67 { id_main = = 2, id_sub = = 1 };

  // public data members
  float range_unit;  // length of 1 LSB of range in meter
  uint32_t line_circle_count;  // number of LSBs per full rotation about line axis
  uint32_t frame_circle_count;  // number of LSBs per full rotation about frame axis
  float time_unit;  // duration of 1 LSB of time in [sec]
  float amplitude_unit;  // width of 1 LSB of amplitude and reflectance in dB
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template units_2

scanlib::units_2 — units information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct units_2 {

  enum @68 { id_main = = 2, id_sub = = 2 };

  // public data members
  float range_unit;  // length of 1 LSB of range in meter
  uint32_t line_circle_count;  // number of LSBs per full rotation about line axis
  uint32_t frame_circle_count;  // number of LSBs per full rotation about frame axis
  float time_unit;  // duration of 1 LSB of time in [sec]
  float amplitude_unit;  // width of 1 LSB of amplitude and reflectance in dB
  float time_unit_hi_prec;  // duration of 1 LSB of high-precision systime in [sec]
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template units_3

scanlib::units_3 — units information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct units_3 {

  enum @69 { id_main = = 2, id_sub = = 3 };

  // public data members
  float range_unit;  // length of 1 LSB of range in meter
  uint32_t line_circle_count;  // number of LSBs per full rotation about line axis
  uint32_t frame_circle_count;  // number of LSBs per full rotation about frame axis
  float time_unit;  // duration of 1 LSB of time in [sec]
  float amplitude_unit;  // width of 1 LSB of amplitude and reflectance in dB
  float time_unit_hi_prec;  // duration of 1 LSB of high-precision systime in [sec]
  uint32_t time_bits_hi_prec;  // number of valid bits in hi-prec timestamp
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template units_IMU

scanlib::units_IMU — units information

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct units_IMU {

  enum @70 { id_main = = 60, id_sub = = 0 };

  // public data members
  float gyro_unit;  // nominally 0.005 deg/s
  float accl_unit;  // nominally 0.032667 m/s²
  float timestamp_unit;  // nominally 20 µs
};
```

Description

This package belongs to the predefined selectors:

attribute, instrument

Struct template unsolicited_message

scanlib::unsolicited_message — spontaneous information and error messages

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct unsolicited_message {

  enum @71 { id_main = = 41, id_sub = = 0 };

  // public data members
  uint32_t systime;  // systime of message in units of uints.time_unit
  uint16_t year;  // year
  uint8_t month;  // month 1 .. 12
  uint8_t day;  // day 1 .. 31
  uint8_t hour;  // hour 0 .. 23
  uint8_t minute;  // minute 0 .. 59
  uint8_t second;  // second 0 .. 59 (60 if leap second)
  int8_t utc_offset;  // diff. between localtime and UTC
  uint8_t type;  // (INFO=1,WARNING=2,ERROR=3,FATAL=4)
  uint32_t intended_followup_actions;  // bitmapped actions (1 .. shutting down laser power -
supply 2 .. stopping current scan 4 .. aborting current scan 8 .. initiating shutdown 16 .. -
normal operation initiated)
  uint32_t id;  // unique (error) number
  char message;  // displayable user info
  uint32_t RESERVED_00;
  char RESERVED_01;
};
```

Description

This package belongs to the predefined selectors:

notify

Struct template unsolicited_message_1

scanlib::unsolicited_message_1 — spontaneous information and error messages

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct unsolicited_message_1 {

  enum @72 { id_main = = 41, id_sub = = 1 };

  // public data members
  uint32_t systime;  // systime of message in units of uints.time_unit
  uint16_t year;  // year
  uint8_t month;  // month 1 .. 12
  uint8_t day;  // day 1 .. 31
  uint8_t hour;  // hour 0 .. 23
```

```
  uint8_t minute;  // minute 0 .. 59
  uint8_t second;  // second 0 .. 59 (60 if leap second)
  int8_t utc_offset;  // diff. between localtime and UTC
  uint8_t type;  // (INFO=1,WARNING=2,ERROR=3,FATAL=4)
  uint32_t intended_followup_actions;  // bitmapped actions (1 .. shutting down laser power -
supply 2 .. stopping current scan 4 .. aborting current scan 8 .. initiating shutdown 16 .. -
normal operation initiated)
  uint32_t id;  // unique (error) number
  char message;  // displayable user info
  uint32_t RESERVED_00;
  char RESERVED_01;
  uint32_t RESERVED_02;
};
```

Description

This package belongs to the predefined selectors:

notify

Struct template void_data

scanlib::void_data — empty helper package

Synopsis

```
// In header: <riegl/ridataspec.hpp>

template<typename it>
struct void_data {

  enum @73 { id_main = = 65535, id_sub = = 65535 };
};
```

Description

This package is used for data alignment purposes.

This package belongs to the predefined selectors:

protocol

Type definition selector_type

selector_type — class selector

Synopsis

```
// In header: <riegl/ridataspec.hpp>


typedef std::bitset< 202 > selector_type;
```

Description

A selector is a bitset, indexed by the package id. The package id to use for indexing is one of the enum values defined in struct package_id. A couple of predefined sets are available. Selectors can be understood as sets whose elements are package (id's). Selector can be combined using set operations. Please look for bitset in your documentation of the C++ standard library to find out more.

Please note, that while it is possible to create arbitrary selector combinations, care should be taken to always include the mandatory header packet. Selectors are used with the basic_packets class.

Global select_all

scanlib::select_all — select all packets

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_all;
```

Description

This selector selects all documented packets.

Global select_none

scanlib::select_none — select no packets

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_none;
```

Description

This selector selects no packets.

Global select_protocol

scanlib::select_protocol — select packets of protocol class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_protocol;
```

Description

This selector selects the following documented packets:

frame_start_dn, frame_start_up, frame_stop, header, header_ext, line_start_dn, line_start_segment_1, line_start_segment_2, line_start_segment_3, line_start_up, line_stop, meas_start, meas_stop, void_data

Global select_attribute

scanlib::select_attribute — select packets of attribute class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_attribute;
```

Description

This selector selects the following documented packets:

atmosphere, atmosphere_1, atmosphere_2, atmosphere_3, atmosphere_4, beam_geometry, biaxial_geometry, device_mounting, extents, monitoring_info, mta_settings, mta_settings_1, mta_settings_2, scan_rect_fov, scan_segments_fov, scanner_pose, scanner_pose_hr, scanner_pose_hr_1, scanner_pose_ucs, scanner_pose_ucs_1, units, units_1, units_2, units_3, units_IMU

Global select_instrument

scanlib::select_instrument — select packets of instrument class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_instrument;
```

Description

This selector selects the following documented packets:

beam_geometry, biaxial_geometry, device_mounting, extents, mta_settings, mta_settings_1, mta_settings_2, units, units_1, units_2, units_3, units_IMU

Global select_scan

scanlib::select_scan — select packets of scan class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_scan;
```

Description

This selector selects the following documented packets:

atmosphere, atmosphere_1, atmosphere_2, atmosphere_3, atmosphere_4, counter_sync, frame_start_dn, frame_start_up, frame_stop, inclination_wyler, line_start_dn, line_start_segment_1, line_start_segment_2, line_start_segment_3, line_start_up, line_stop, meas_start, meas_stop, monitoring_info, mta_settings, mta_settings_1, mta_settings_2, pps_sync, pps_sync_ext, pps_sync_hr, pps_sync_hr_ext, scan_rect_fov, scan_segments_fov, scanner_pose, scanner_pose_hr, scanner_pose_hr_1, scanner_pose_ucs, scanner_pose_ucs_1

Global select_data

scanlib::select_data — select packets of data class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_data;
```

Description

This selector selects the following documented packets:

IMU_data, counter_sync, inclination_wyler, pps_sync, pps_sync_ext, pps_sync_hr, pps_sync_hr_ext

Global select_deprecated

scanlib::select_deprecated — select packets of deprecated class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_deprecated;
```

Description

This selector selects the following documented packets:

hk_bat, hk_pwr

Global select_status

scanlib::select_status — select packets of status class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_status;
```

Description

This selector selects the following documented packets:

hk_bat, hk_bat_1, hk_bat_2, hk_ctr, hk_ctr_1, hk_gps, hk_gps_hr, hk_gps_ts, hk_gps_ts_status, hk_gps_ts_status_dop, hk_gps_ts_status_dop_ucs, hk_incl, hk_incl_4axes, hk_ph_data, hk_ph_data_1, hk_ph_units, hk_ph_units_1, hk_pwr, hk_pwr_1, hk_rtc, hk_rtc_sys, pwm_sync

Global select_notify

scanlib::select_notify — select packets of notify class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_notify;
```

Description

This selector selects the following documented packets:

unsolicited_message, unsolicited_message_1

Global select_legacy

scanlib::select_legacy — select packets of legacy class

Synopsis

```
// In header: <riegl/ridataspec.hpp>

const selector_type select_legacy;
```

Description

This selector selects the following documented packets:

hk_rtc, hk_rtc_sys

Macro RIDATASPEC_VERSION

RIDATASPEC_VERSION — The version of ridataspec.

Synopsis

```
// In header: <riegl/ridataspec.hpp>

RIDATASPEC_VERSION
```

Description

The version macro evaluates to "2,76,1147" for this version of the library. The three fields are: major,minor and build numbers. A change in major number typically will indicate changes in the API or kind of packages.

Header <riegl/buffer.hpp>

```


SCANLIB_RXPMARKER_ELEMENT_TYPE
```

```
namespace scanlib {
  template<typename it> class basic_buffer;

  typedef basic_buffer< SCANLIB_RXPMARKER_ELEMENT_TYPE * > buffer;  // specialisation of -
basic_buffer for most common use
}
```

Class template basic_buffer

scanlib::basic_buffer — buffer class

Synopsis

```
// In header: <riegl/buffer.hpp>

template<typename it>
class basic_buffer {
public:
  // types
  typedef std::iterator_traits< it >::value_type           value_type;
  typedef std::size_t                                      size_type;
  typedef value_type *                                     iterator;
  typedef const value_type *                               const_iterator;
  typedef std::iterator_traits< iterator >::difference_type difference_type;

  // construct/copy/destruct
  basic_buffer();
  ~basic_buffer();

  // public member functions
  iterator begin();
  iterator end();
  const_iterator begin() const;
  const_iterator end() const;
  size_type size() const;
  size_type max_size() const;
  bool empty() const;
  void resize(size_type, value_type = value_type());
};
```

Description

The buffer class, despite its name is only a reference to a memory range within the decoder buffer. The purpose of this is to avoid unnecessary copies of the data.

`basic_buffer` public construct/copy/destruct

1. ```
   basic_buffer();
   ```

2. ```
   ~basic_buffer();
   ```

`basic_buffer` public member functions

1. ```
   iterator begin();
   ```

   Returns: iterator to begin of buffer

2. ```
   iterator end();
   ```

   Returns: iterator to end of buffer

3. ```
   const_iterator begin() const;
   ```

   Returns: const iterator to begin of buffer for reading

4. ```
   const_iterator end() const;
   ```

Returns: const iterator to end of buffer space for reading

5.
```
size_type size() const;
```

Returns: size of buffer space

6.
```
size_type max_size() const;
```

Returns: capacity of buffer space

7.
```
bool empty() const;
```

Returns: buffer is empty indicator

8.
```
void resize(size_type s, value_type z = value_type());
```

resize the buffer, adjust begin and end
Parameters: s new size
z optional initialization value

Macro SCANLIB_RXPMARKER_ELEMENT_TYPE

SCANLIB_RXPMARKER_ELEMENT_TYPE — The basic data type for RXP streams. Need to be 32 bit.

Synopsis

```
// In header: <riegl/buffer.hpp>

SCANLIB_RXPMARKER_ELEMENT_TYPE
```

Header <riegl/connection.hpp>

The abstract base classes for read and write connection protocols.

```
namespace scanlib {
  class basic_connection;
  class basic_rconnection;
  class cancelled;
}
```

Class basic_connection

scanlib::basic_connection — base class for connections

Synopsis

```
// In header: <riegl/connection.hpp>


class basic_connection {
public:
  // types
```

```
  typedef std::size_t size_type;
  typedef uint64_t    pos_type;

  // construct/copy/destruct
  basic_connection();
  basic_connection(const basic_connection &);
  const basic_connection & operator=(const basic_connection &);
  ~basic_connection();

  // public member functions
  virtual void open();
  virtual void close();
  virtual void cancel();
  bool eoi() const;
  operator void *();

  // public data members
  std::string id;  // An id describing the connection, e.g. the file name from the remote -
side.
};
```

Description

Basic connection class for read and write protocols.

`basic_connection` public construct/copy/destruct

1. ```
   basic_connection();
   ```

   The default constructor is protected to enforce derivation from this class.
   Postconditions: id is initialized to a default value of "UNNAMED_<year><month><day>_<hour><minute><second>"

2. ```
   basic_connection(const basic_connection &);
   ```

3. ```
   const basic_connection & operator=(const basic_connection &);
   ```

4. ```
   ~basic_connection();
   ```

`basic_connection` public member functions

1. ```
   virtual void open();
   ```

   The open and close may be implemented by a derived object where the constructor needs to block to establish or shut down the connection.

2. ```
   virtual void close();
   ```

3. ```
   virtual void cancel();
   ```

   cancel outstanding operation

   Cause a blocked read or write to throw an exception of type "cancelled".

4. ```
   bool eoi() const;
   ```

5.
```
operator void *();
```

Class basic_rconnection

scanlib::basic_rconnection — abstract protocol class for read connections.

Synopsis

```
// In header: <riegl/connection.hpp>


class basic_rconnection : public scanlib::basic_connection {
public:
  // construct/copy/destruct
  basic_rconnection();
  ~basic_rconnection();

  // public static functions
  static std::shared_ptr< basic_rconnection >
  create(const std::string &, const std::string & = std::string());

  // public member functions
  size_type readsome(void *, size_type);
  basic_rconnection & read(void *, size_type);
  size_type gcount() const;
  virtual pos_type tellg() const;
  pos_type size() const;
  virtual basic_rconnection & seekg(pos_type);
  void open_logfile(const std::string &);
  void close_logfile();
  virtual std::string continuation() const;
  virtual void request_shutdown();

  // protected member functions
  virtual size_type more_input(void *, size_type) = 0;
};
```

Description

This is the abstract base class for read connections

`basic_rconnection` public construct/copy/destruct

1.
```
basic_rconnection();
```

2.
```
~basic_rconnection();
```

`basic_rconnection` public static functions

1.
```
static std::shared_ptr< basic_rconnection >
create(const std::string & uri,
       const std::string & continuation = std::string());
```

virtual constructor

Parameters:	`continuation`	resume information for aborted transfers
	`uri`	connection uri
Returns:	connection class matching the protocol specified in uri	

`basic_rconnection` public member functions

1.
```
size_type readsome(void * buf, size_type count);
```

Read at maximum count bytes into user provided buffer. Function may block if no bytes available.
Parameters: `buf` user provided buffer
`count` size of buffer in bytes
Returns: number of actually read bytes or zero if sending side has finished sending.
Throws: scanlib::cancelled

2.
```
basic_rconnection & read(void * buf, size_type count);
```

Read count bytes into user provided buffer. Function will block if no bytes available.
Parameters: `buf` user provided buffer
`count` size of buffer in bytes
Returns: a reference to the read object.
Throws: scanlib::cancelled

3.
```
size_type gcount() const;
```

Returns: return the number of bytes read during last succesful read

4.
```
virtual pos_type tellg() const;
```

Returns: total number of octets already read

5.
```
pos_type size() const;
```

The number of octetts of the stream if its file based, else zero.

6.
```
virtual basic_rconnection & seekg(pos_type pos);
```

Seek to a possition that has been retrieved with tellg if the stream is file based, else throws

7.
```
void open_logfile(const std::string & name);
```

Set a filname for data logging. The rxp stream will be logged in binary to this file.
Parameters: `name` filname of logfile

8.
```
void close_logfile();
```

Close the logfile.

9.
```
virtual std::string continuation() const;
```

Get context to continue an aborted transfer. The returned string can be feed into create to resume a transfer.
Returns: the continuation string

10.
```
virtual void request_shutdown();
```

Signal a request to stop sending to the remote peer. After requesting a shutdown, the application is expected to read data until end of file is detected.

`basic_rconnection` protected member functions

1.
```
virtual size_type more_input(void * buf, size_type count) = 0;
```

Class cancelled

scanlib::cancelled — exception, thrown when cancelled

Synopsis

```
// In header: <riegl/connection.hpp>


class cancelled : public runtime_error {
public:
  // construct/copy/destruct
  explicit cancelled(const std::string &);
  ~cancelled();
};
```

Description

`cancelled` public construct/copy/destruct

1.
```
explicit cancelled(const std::string & message);
```

Parameters: `message` A description of the cancellation request.

2.
```
~cancelled();
```

Header <riegl/fileconn.hpp>

The file connection classes

```
namespace scanlib {
  class file_rconnection;
}
```

Class file_rconnection

scanlib::file_rconnection — the file connection class

Synopsis

```
// In header: <riegl/fileconn.hpp>


class file_rconnection : public scanlib::basic_rconnection {
public:
  // construct/copy/destruct
  file_rconnection(const std::string &, const std::string & = std::string());
  ~file_rconnection();
```

```
  // public member functions
  virtual file_rconnection & seekg(pos_type);
  virtual void cancel();
  virtual void request_shutdown();

  // protected member functions
  virtual size_type more_input(void *, size_type);
};
```

Description

`file_rconnection` public construct/copy/destruct

1. ```
   file_rconnection(const std::string & file_uri,
                    const std::string & continuation = std::string());
   ```

   constructor for file connection
   Parameters: `continuation` not applicable to file connections
   `file_uri` a file specifier e.g. file:filename.rxp

2. ```
   ~file_rconnection();
   ```

`file_rconnection` public member functions

1. ```
   virtual file_rconnection & seekg(pos_type pos);
   ```

   Seek to a possition that has been retrieved with tellg.

2. ```
   virtual void cancel();
   ```

   The cancel request note: this function is not really useful for files

3. ```
   virtual void request_shutdown();
   ```

   The shutdown request note: this function is not really useful for files

`file_rconnection` protected member functions

1. ```
   virtual size_type more_input(void * buf, size_type count);
   ```

Header <riegl/pointcloud.hpp>

The pointcloud extractor and helper classes.

```
namespace scanlib {
  class pointcloud;
  class target;
}
```

Class pointcloud

scanlib::pointcloud — pointcloud class

Synopsis

```
// In header: <riegl/pointcloud.hpp>


class pointcloud : public compressed_packets {
public:
  // types
  typedef std::vector< target >::size_type target_count_type;

  // the echo type classifications, each echo belongs to exactly one class
  enum echo_type { single, first, interior, last, none };

  // construct/copy/destruct
  pointcloud(bool = false);
  pointcloud(std::ostream &, bool = false);
  pointcloud(const pointcloud &);
  ~pointcloud();

  // public member functions
  void set_rangegate(unsigned = 0, double = 0, double = 0);

  // protected member functions
  virtual void on_measurement_starting();
  virtual void on_shot();
  virtual void on_shot_end();
  virtual void on_echo_transformed(echo_type);
  virtual void on_pps_synchronized();
  virtual void on_pps_sync_lost();

  // public data members
  static const double pi;  // the value of PI
  unsigned num_facets;  // number of facets (instrument constant)
  double group_velocity;  // group velocity of laser beam
  double unambiguous_range;  // maximum unambiguoas range (mta zone width)
  bool sync_to_pps;  // flag indicating the time reference to use
  double time;  // the timestamp of the current shot in sec (ref. depends on sync_to_pps)
  double time_sorg;  // time of start of range gate of last shot
  double beam_direction;  // beam direction of current shot (unity direction vector)
  double beam_origin;  // (virtual) beam origin of current shot (unit is meter)
  unsigned facet;  // mirror facet or segment of current shot
  std::vector< target > targets;  // all targets belonging to the last laser shot
  target_count_type target_count;  // the number of targets belonging to the last laser shot
  std::string serial;  // scanner serial number
  std::string type_id;  // scanner type
  std::string build;  // scanner build id
};
```

Description

The pointcloud class combines several (undocumented) raw basic packets into meaningful XYZ data with calibration already applied. This makes the pointcloud class the main access point to data for client applications. The class provides several virtual functions that the client application is expected to override. For every laser echo discovered in the raw data stream from the scanning device an invocation of the on_echo_transformed is done. The function provides as a parameter the type of echo. Each echo unambiguously belongs to one of the possible classes of type echo_type. Please note that for specifying all echoes that are farthest away you need to look for both single *and* last echoes. Similar reasoning is for closest echoes. I.e. specifying only last will give you the echo points that are farthest away but only if they are part of multi-return. The time reference for the time stamps are user selectable. PPS time stamps from a GPS receiver or internal time of the scanner device can be choosen. The point data is accesible through the targets member variable. This variable is an array that always holds all echoes that belong to the same laser shot. The echo that belongs to the current call to on_echo_transformed is always accesible by means of the variable target_count. I.e to access the top of the array specify targets[target_count-1]. This design makes it possible for the client application to either act on each echo separately

or evaluate all echoes that belong to a laser shot at once when the last echo for this shot is seen. The targets array is reset to empty with each new laser shot.

`pointcloud` public construct/copy/destruct

1. ```
   pointcloud(bool sync_to_pps_ = false);
   ```

   constructor
   Parameters: `sync_to_pps_` use external time reference for time

2. ```
   pointcloud(std::ostream & out, bool sync_to_pps_ = false);
   ```

   constructor
   Parameters: `out` write pointcloud data in ASCII to a stream
   `sync_to_pps_` use external time reference for time

3. ```
   pointcloud(const pointcloud &);
   ```

4. ```
   ~pointcloud();
   ```

`pointcloud` public member functions

1. ```
   void set_rangegate(unsigned zone = 0, double near = 0, double far = 0);
   ```

   override the default rangegate of the sensor
   Parameters: `far` if zone=0 specifies the farthmost end of the rangegate
   `near` if zone=0 specifies the closer end of the rangegate
   `zone` select the mta zone. (if all parameters are zero, select sensor default)

`pointcloud` protected member functions

1. ```
   virtual void on_measurement_starting();
   ```

   callback when the measurement is about to start, i.e. metadata is ready this packet is similar to on_meas_start but not the same, since the latter is generated by the device while this one considers more cases

2. ```
   virtual void on_shot();
   ```

   callback when a laser shot is available, will be invoked even if no echoes are available for this shot

3. ```
   virtual void on_shot_end();
   ```

   callback after all data belonging to a shot is available i.e. all echo packets (same as last echo)

4. ```
   virtual void on_echo_transformed(echo_type echo);
   ```

   callback when a complete laser point is available the point information is accesible from the targets member
   Parameters: `echo` the classification of the last echo

5. ```
   virtual void on_pps_synchronized();
   ```

   callback to indicate that at least two regular pps pulses have been seen

6.
```
virtual void on_pps_sync_lost();
```

callback to indicate that pps pulses have stopped

Class target

scanlib::target — geometric data and attributes of a point

Synopsis

```
// In header: <riegl/pointcloud.hpp>


class target {
public:
  // construct/copy/destruct
  target();

  // public data members
  float vertex;  // x-y-z coordinates
  double echo_range;  // echo range in units of meter
  unsigned short zone_index;  // zone index, zero offset
  double time;  // time stamp in [s]
  float amplitude;  // relative amplitude in [dB]
  float reflectance;  // relative reflectance in [dB]
  float deviation;  // a measure of pulse shape distortion
  float background_radiation;  // a measure of background radiation
  bool is_pps_locked;  // time is a valid pps time
};
```

Description

This class holds the geometric x,y,z coordinates of a measured point and its attributes such as amplitude and reflectance.

`target` public construct/copy/destruct

1.
```
target();
```

`target` public public data members

1.
```
float vertex;
```

Cartesian coordinates in scanners own coordinate system. vertex[i] = beam_origin[i] + echo_range * beam_direction[i]

2.
```
double echo_range;
```

The distance from the laser origin to the target. Note that this is not the same as sqrt(vertex[0]^2 + vertex[1]^2 + vertex[2]^2).

3.
```
unsigned short zone_index;
```

The zone this target belongs to. Note: zone == zone_index + 1

4.
```
double time;
```

The epoch (i.e. where time equals zero) depends on a flag that has to be supplied in the constructor of the pointcloud object.

5. ```
   float amplitude;
   ```

   This is the ratio of the received power to the power received at the detection limit expressed in dB.

6. ```
   float reflectance;
   ```

   This is the ratio of the received power to the power that would be received from a white diffuse target at the same distance expressed in dB. The reflectance represents a range independent property of the target. The surface normal of this target is assumed to be in parallel to the laser beam direction.

7. ```
   float deviation;
   ```

   A larger value for deviation indicates larger distortion.

8. ```
   float background_radiation;
   ```

   A measure of background radiation. This information is optional. For instruments that do not deliver this value, the field is set to NaN.

9. ```
   bool is_pps_locked;
   ```

   The external PPS signal was found to be synchronized at the time of the current laser shot.

Header <riegl/rdtpconn.hpp>

The R-iegl D-ata T-transfer P-rotocol classes

```
namespace scanlib {
  class rdtp_rconnection;
}
```

Class rdtp_rconnection

scanlib::rdtp_rconnection

Synopsis

```
// In header: <riegl/rdtpconn.hpp>


class rdtp_rconnection : public scanlib::basic_rconnection {
public:
  // construct/copy/destruct
  rdtp_rconnection(const std::string &, const std::string & = std::string());
  rdtp_rconnection(const rdtp_rconnection &);
  rdtp_rconnection operator=(const rdtp_rconnection &);
  ~rdtp_rconnection();

  // public member functions
  virtual void open();
  virtual void close();
  virtual void cancel();
  virtual void request_shutdown();
```
```

```
  // protected member functions
  virtual size_type more_input(void *, size_type);
  virtual std::string continuation() const;
};
```

Description

`rdtp_rconnection` public construct/copy/destruct

1. ```
   rdtp_rconnection(const std::string & rdtp_uri,
                    const std::string & continuation = std::string());
   ```

   constructor for rdtp read connection
   Parameters: `continuation` a continuation string from a previous connnection

2. ```
   rdtp_rconnection(const rdtp_rconnection &);
   ```

3. ```
   rdtp_rconnection operator=(const rdtp_rconnection &);
   ```

4. ```
   ~rdtp_rconnection();
   ```

`rdtp_rconnection` public member functions

1. ```
   virtual void open();
   ```

   open the connection, override of base open

2. ```
   virtual void close();
   ```

   close the connection, override of base close

3. ```
   virtual void cancel();
   ```

   cancel request, override of base cancel

4. ```
   virtual void request_shutdown();
   ```

   request for shutdown of the connection, override of base

`rdtp_rconnection` protected member functions

1. ```
   virtual size_type more_input(void * buf, size_type count);
   ```

2. ```
   virtual std::string continuation() const;
   ```

   The continuation string
   Returns: the continuation string for use in a later resume attempt

Header <riegl/rmsmarker.hpp>

The decoder marker classes for multiplexed streams.

```
namespace scanlib {
  class decoder_rmsmarker;
}
```

Class decoder_rmsmarker

scanlib::decoder_rmsmarker — The RMS decoder class.

Synopsis

```
// In header: <riegl/rmsmarker.hpp>


class decoder_rmsmarker {
public:
  // construct/copy/destruct
  decoder_rmsmarker(std::shared_ptr< basic_rconnection >, unsigned = 0,
                    std::vector< std::string > = std::vector< std::string >());

  // public member functions
  operator const void *() const;
  bool eoi() const;
  uint16_t get(buffer &);

  // public data members
  std::map< uint16_t, std::string > jobs;  // array of job names
  std::map< uint16_t, std::string > ids;  // array of joob identifier names as fetched from -
the original device
};
```

Description

This class demultiplexes several rxp streams from an rms stream. Then each rxp stream is processed as an escaped raw binary input stream and binary packets that are properly segemented by the package markers are provided on the output. The class also undoes the escaping that was inserted on the transmitting side. Each package is retuned together with a number denoting the rxp stream it is belonging to.

`decoder_rmsmarker` public construct/copy/destruct

1.
```
decoder_rmsmarker(std::shared_ptr< basic_rconnection > rconnection_,
                  unsigned max_jobs_ = 0,
                  std::vector< std::string > jobs_ = std::vector< std::string >());
```

This constructor accepts an rconnection as a shared pointer Use this constructor to delegate management of object lifetime to the decoder.

Parameters:

`jobs_`	a list of jobnames to filter the returned jobs. The order of names given will correspond with the jobnr returned in the get function.
`max_jobs_`	the maximum number of different jobs that will be be returned. If set to zero all jobs will be returned.
`rconnection_`	the source data connection

`decoder_rmsmarker` public member functions

1.
```
operator const void *() const;
```

alternative form for end of input testing

2. ```
   bool eoi() const;
   ```

   return true if end of input has been reached

3. ```
   uint16_t get(buffer & b);
   ```

   get the next available binary data packet
   Parameters: b a buffer proxy
   Returns: jobnr of packet

Header <riegl/rxpmarker.hpp>

The decoder (and encoder) marker buffer classes.

```
SCANLIB_RXPMARKER_ELEMENT_TYPE
```

```
namespace scanlib {
  class decoder_rxpmarker;
}
```

Class decoder_rxpmarker

scanlib::decoder_rxpmarker — The RXP decoder class.

Synopsis

```
// In header: <riegl/rxpmarker.hpp>


class decoder_rxpmarker : public scanlib::basic_rxpmarker {
public:
  // construct/copy/destruct
  decoder_rxpmarker(basic_rconnection &, size_type = 1024, size_type = 340,
                    bool = true);
  decoder_rxpmarker(std::shared_ptr< basic_rconnection >, size_type = 1024,
                    size_type = 340, bool = true);

  // public member functions
  operator const void *() const;
  bool eoi() const;
  virtual uint16_t get(buffer &);
};
```

Description

This class consumes an escaped raw binary input stream and provides binary packets that are properly segemented by the package markers that designate the package boundaries. The class also undoes the escaping that was inserted on the transmitting side.

`decoder_rxpmarker` public construct/copy/destruct

1. ```
   decoder_rxpmarker(basic_rconnection & rconnection, size_type size = 1024,
                     size_type headroom = 340, bool with_preamble = true);
   ```
   ```

This constructor accepts a rconnection by reference.

Parameters:		
	`headroom`	internal headroom for packet unescaping (leave default)
	`rconnection`	the source data connection
	`size`	internal buffer space for packets (leave default)
	`with_preamble`	do not skip preample (leave default)

2.
```
decoder_rxpmarker(std::shared_ptr< basic_rconnection > rconnection,
                  size_type size = 1024, size_type headroom = 340,
                  bool with_preamble = true);
```

This constructor accepts an rconnection as a shared pointer Use this constructor to delegate management of object lifetime to the decoder.

Parameters:		
	`headroom`	internal headroom for packet unescaping (leave default)
	`rconnection`	the source data connection
	`size`	internal buffer space for packets (leave default)
	`with_preamble`	do not skip preample (leave default)

`decoder_rxpmarker` public member functions

1.
```
operator const void *() const;
```

alternative form for end of input testing

2.
```
bool eoi() const;
```

return true if end of input has been reached

3.
```
virtual uint16_t get(buffer & b);
```

get the next available binary data packet

Parameters:		
	`b`	a buffer proxy

Macro SCANLIB_RXPMARKER_ELEMENT_TYPE

SCANLIB_RXPMARKER_ELEMENT_TYPE

Synopsis

```
// In header: <riegl/rxpmarker.hpp>

SCANLIB_RXPMARKER_ELEMENT_TYPE
```

Header <riegl/scanlib.hpp>

Convenience include header, including all rivlib headers

Ctrllib

Ctrllib Usage

*Riegl*s V-Line™ scanner are configured via a TCP configuration port (usually port 20002). All configuration and controlling is done via a command/reply structure: ASCII commands are sent to the scanner and ASCII replies inform about values and function results. Please refer to the scanner manual for a detailed description of the command syntax. Also, a list of all available methods and properties can be found in this documentation.

Configuration of and controlling the scanner mainly consist of

- setting and getting property values (e.g. setting the scan parameters, measurement programs, etc.) and
- calling methods/executing commands (e.g. starting and stopping data acquisition).

The library provides a class to support instrument configuration via this configuration port. Setting and getting of property values are done with

- get_property(*`property_name`* [in], *`property_value`* [out]) and
- set_property(*`property_name`* [in], *`property_value`* [in]) methods

of this class. Please note that property values are set/returned as string values and thus must be converted into their native representation (integer, floating-point, etc.) when used for computation. Commands are executed using

- execute_command(*`method_name`* [in], *`method_arguments`* [in], *`result_as_string`* [out]).

The argument list is a single string containing the arguments separated by , (comma). Please note: Error handling in general is making use of C++'s `throw/catch` mechanism. In case a scanner command cannot be executed properly (communication fault, unknown property/method, etc.), an exception derived from `std::exception` is thrown. You can always get the error message from evaluating the `what()` member of the exception object thrown.

Example:

```
#include <riegl/ctrllib.hpp>
#include <stdexcept>

using namespace ctrllib;

int main(int argc, char* argv[])
{
    try {
        ctrl_client_session session;

        session.open("s9991234");

        std::string inst_ident;
        session.get_property("INST_IDENT", inst_ident);

        session.set_property("HMI_SOUND", "0");
        session.set_property("TIMEZONE", "\"Europe/Vienna\"");

        session.execute_command("MEAS_BUSY", "0");
        session.execute_command("SCN_SET_SCAN", "30, 130, 10");
        session.execute_command("MEAS_START", "");
        /* ...more commands... */

        session.close();
    }
    catch(exception& e) {
        cerr << e.what() << endl;
        return 1;
    }
```

```
    catch(...) {
        cerr << "unknown exception" << endl;
        return 1;
    }
    return 0;
}
```

Scanner also support a Telnet port (port number 23): This port is largely equivalent to the configuration port 20002, but supports some additional features like editing of input text command auto completion. This port is mainly designed to be used with manual command input, e.g. when used with a terminal program, and is recommended for developers of interfaces to the scanner (testing of command sequences) or to get familiar with the scanner. The scanner control class described in this documentation does not use the Telnet port.

Ctrllib Reference

Header <riegl/ctrllib.hpp>

The scanner control library

```
namespace ctrllib {
  class ctrl_client_session;
  class operation_cancelled;
}
```

Class ctrl_client_session

ctrllib::ctrl_client_session — scanner control session

Synopsis

```
// In header: <riegl/ctrllib.hpp>


class ctrl_client_session {
public:
  // construct/copy/destruct
  ctrl_client_session();
  ~ctrl_client_session();

  // public member functions
  void open(const std::string &);
  void close();
  void cancel();
  void execute_command(const std::string &, const std::string &,
                       std::string * = 0, std::string * = 0,
                       std::vector< std::string > * = 0);
  void set_property(const std::string &, const std::string &,
                    std::string * = 0, std::vector< std::string > * = 0);
  void get_property(const std::string &, std::string &, std::string * = 0,
                    std::vector< std::string > * = 0);
};
```

Description

The ctrl_client_session class represents a scanner's configuration port.

`ctrl_client_session` public construct/copy/destruct

1. ```
   ctrl_client_session();
   ```

   constructor for scanner control object no communication is established yet (see open)

2. ```
   ~ctrl_client_session();
   ```

   destructor for scanner control object

`ctrl_client_session` public member functions

1. ```
   void open(const std::string & authority);
   ```
   ```

open a connection to the scanner instrument The open function will locate the scanner by means of the authority information, which may be given as "host:port", where host and port either are in text or numeric forms.

Parameters: `authority` [in]: the address of the scanner the authority string is expected in format "<host_name_or_ip_address>[:<service_name_or_port_number>]" Examples: "192.168.0.234", "192.168.0.234:20002", "s9991234" or "s9991234:20002" if port number is not specified, 20002 is assumed as default

2.
```
void close();
```

The close method will close the connection to the scanner.

3.
```
void cancel();
```

cancel a pending operation

4.
```
void execute_command(const std::string & command,
                     const std::string & arguments, std::string * result = 0,
                     std::string * status = 0,
                     std::vector< std::string > * remarks = 0);
```

Execute a command (function call). The execute_command function calls a function of the scanner interface. The value is returned as string value:

- string properties are enclosed in quotation marks "..."
- integer and floating point values must be converted
- boolean properties are strings 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:
`arguments`	[in] the arguments for the function. Specify "" for no arguments to be passed to command.
`command`	[in] the name of the command/function to execute/call
`remarks`	[out] received comment lines from instrument (optional).
`result`	[out] result value as string (optional). Vectors are returned as value list separated by , (comma).
`status`	[out] control port status (optional). Format is "<...>".

5.
```
void set_property(const std::string & name, const std::string & value,
                  std::string * status = 0,
                  std::vector< std::string > * remarks = 0);
```

Set a property value. The set_property function sets a new value for a property. The new value is given as string value:

- string properties are to be enclosed in quotation marks "..."
- integer and floating point values must be converted into a string
- boolean properties are 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:
`name`	[in] the name of the property to set
`remarks`	[out] received comment lines from instrument (optional).
`status`	[out] control port status (optional). Format is "<...>".
`value`	[in] the new property value

6.
```
void get_property(const std::string & name, std::string & value,
                  std::string * status = 0,
```

```
                    std::vector< std::string > * remarks = 0);
```

Get a property value. The get_property function returns the value of a property. The value is returned as string value:

- string properties are enclosed in quotation marks "..."
- integer and floating point values must be converted
- boolean properties are strings 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:

`name`	[in] the name of the property to retrieve
`remarks`	[out] received comment lines from instrument (optional).
`status`	[out] control port status (optional). Format is "<...>".
`value`	[out] the property value as string

Class operation_cancelled

ctrllib::operation_cancelled — brief exception class thrown when an operation has been cancelled

Synopsis

```
// In header: <riegl/ctrllib.hpp>


class operation_cancelled : public exception {
public:

  // private member functions
  virtual const char * what() const;
};
```

Description

`operation_cancelled` private member functions

1. ```
   virtual const char * what() const;
   ```
   ```

Part III. Programming Language Neutral Interface

Table of Contents

Scanifc

Scanifc Usage

*Riegl*s V-Line™ scanner deliver measurement data in a stream of binary coded packets. These packets need to be converted into a format that is directly usable from an application program. The stream of data packets contains quite some information that normally is of little interest to the end user of the great majority of applications. Also not every application uses the C++ language but still needs access to the data. This is the domain of the language neutral interface of the RiVLib™.

This part of the library consists of a few header files, `scanifc.h` and the implicitely used `riegl/detail/baseifc_t.h`, `riegl/detail/pointsifc_t.h`, a shared object library `scanifc-mt` and the corresponding import library (for windows only).

The typical usage pattern for this interface is to

1. Open a connection to a scanner device or file

2. Read points in a loop until end

3. Close the connection when done

The data is available as a cloud of points each consisting of XYZ range data, attributes and a time stamp.

In addition to the point related data interface there is an additional means to get access to other packets as well. It is possible to specify a file that is to receive a subset of the packets, converted into ASCII format. This data however is not directly available to the application program, but need to be read from the file after the connection has been closed. If you need a more flexible interface, consider writing a intermediary dll using a C++ compiler that interfaces to RiVLib™ internally and exposes a plain C interface to your application program.

Scanifc Reference

Header <riegl/detail/baseifc_t.h>

Basic data types used in the DLL interface.

```
typedef IMPLEMENTATION_DEFINED scanifc_uint8_t;  // unsigned 8 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_int8_t;  // signed 8 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_uint16_t;  // unsigned 16 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_int16_t;  // signed 16 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_uint32_t;  // unsigned 32 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_int32_t;  // signed 32 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_uint64_t;  // unsigned 64 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_int64_t;  // signed 64 bit integer
typedef IMPLEMENTATION_DEFINED scanifc_uintmax_t;  // unsigned largest available bitsize inte-
ger
typedef IMPLEMENTATION_DEFINED scanifc_intmax_t;  // signed largest available bitsize integer
typedef IMPLEMENTATION_DEFINED scanifc_float32_t;  // 32 bit floating point
typedef IMPLEMENTATION_DEFINED scanifc_float64_t;  // 64 bit floating point
typedef char * scanifc_sz;
typedef const char * scanifc_csz;  // pointer to zero delimited string
typedef scanifc_int32_t scanifc_bool;  // pointer to constant zero delimited string
```

Header <riegl/detail/pointsifc_t.h>

Compound data types used in the DLL interface.

```
struct scanifc_xyz32_t;
struct scanifc_attributes_t;

typedef struct scanifc_xyz32_t scanifc_xyz32;  // a point in cartesian coordinates
typedef struct scanifc_attributes_t scanifc_attributes;  // per point attributes
typedef scanifc_uint64_t scanifc_time_ns;
```

Struct scanifc_xyz32_t

scanifc_xyz32_t — a point in cartesian coordinates

Synopsis

```
// In header: <riegl/detail/pointsifc_t.h>


struct scanifc_xyz32_t {

  // public data members
  scanifc_float32_t x;  // x coordinates
  scanifc_float32_t y;  // y coordinates
  scanifc_float32_t z;  // z coordinates
};
```

Struct scanifc_attributes_t

scanifc_attributes_t — per point attributes

Synopsis

```
// In header: <riegl/detail/pointsifc_t.h>


struct scanifc_attributes_t {

  // public data members
  scanifc_float32_t amplitude;  // relative amplitude in [dB]
  scanifc_float32_t reflectance;  // relative reflectance in [dB]
  scanifc_uint16_t deviation;  // a measure of pulse shape distortion.
  scanifc_uint16_t flags;  // some bit-mapped attribute values
  scanifc_float32_t background_radiation;  // background radiation.
};
```

Description

`scanifc_attributes_t` public public data members

1. ```
   scanifc_float32_t reflectance;
   ```

   The relative reflectance is the received power relative to the power that would be received from a white diffuse target at the same distance. The surface normal of this target is assumed to be in parallel to the laser beam direction.

2. ```
   scanifc_uint16_t flags;
   ```

   The flags combine some attributes into a single value for compact storage. bit0-1: 0 .. single echo 1 .. first echo 2 .. interior echo 3 .. last echo bit2: reserved bit3: waveform available bit4: 1 .. pseudo echo with fixed range 0.1m bit5: 1 .. sw calculated target bit6: 1 .. pps not older than 1.5 sec bit7: 1 .. time in pps timeframe bit8-9: .. facet or segment number 0 to 3 bits10-15: reserved

Type definition scanifc_time_ns

scanifc_time_ns — time stamp in [ns]

Synopsis

```
// In header: <riegl/detail/pointsifc_t.h>


typedef scanifc_uint64_t scanifc_time_ns;
```

Description

The epoch (i.e. where time_ns eqals zero) depends on a flag that has to be supplied in the open call.

Header <riegl/scanifc.h>

The shared object (DLL) interface definitions.

```


SCANIFC_API
```

```


typedef IMPLEMENTATION_DEFINED point3dstream_handle;  // a handle to a stream of 3d point-
cloud data
```

```
typedef IMPLEMENTATION_DEFINED rmsstream_handle;  // a handle to a multiplexed stream of 3d -
pointcloud data
SCANIFC_API int
scanifc_get_library_version(scanifc_uint16_t *, scanifc_uint16_t *,
                            scanifc_uint16_t *);
SCANIFC_API int scanifc_get_library_info(scanifc_csz *, scanifc_csz *);
SCANIFC_API int
scanifc_get_last_error(scanifc_sz, scanifc_uint32_t, scanifc_uint32_t *);
SCANIFC_API int
scanifc_point3dstream_open(scanifc_csz, scanifc_bool, point3dstream_handle *);
SCANIFC_API int
scanifc_point3dstream_open_with_logging(scanifc_csz, scanifc_bool,
                                        scanifc_csz, point3dstream_handle *);
SCANIFC_API int
scanifc_point3dstream_open_rms(scanifc_csz, scanifc_bool, scanifc_csz,
                               scanifc_csz, point3dstream_handle *);
SCANIFC_API int
scanifc_point3dstream_add_demultiplexer(point3dstream_handle, scanifc_csz,
                                        scanifc_csz, scanifc_csz);
SCANIFC_API int
scanifc_point3dstream_set_rangegate(point3dstream_handle, scanifc_uint16_t,
                                    scanifc_float32_t, scanifc_float32_t);
SCANIFC_API int
scanifc_point3dstream_read(point3dstream_handle, scanifc_uint32_t,
                           scanifc_xyz32 *, scanifc_attributes *,
                           scanifc_time_ns *, scanifc_uint32_t *,
                           scanifc_bool *);
SCANIFC_API int
scanifc_point3dstream_tellg(point3dstream_handle, scanifc_uint64_t *);
SCANIFC_API int scanifc_point3dstream_cancel(point3dstream_handle);
SCANIFC_API int scanifc_point3dstream_close(point3dstream_handle);
SCANIFC_API int
scanifc_rmsstream_open(scanifc_csz, scanifc_bool, scanifc_uint16_t,
                       scanifc_csz *, rmsstream_handle *);
SCANIFC_API int
scanifc_rmsstream_read(rmsstream_handle, scanifc_uint32_t, scanifc_xyz32 *,
                       scanifc_attributes *, scanifc_time_ns *,
                       scanifc_uint32_t *, scanifc_bool *,
                       scanifc_uint16_t *);
SCANIFC_API int scanifc_rmsstream_tellg(rmsstream_handle, scanifc_uint64_t *);
SCANIFC_API int
scanifc_rmsstream_get_jobname(rmsstream_handle, scanifc_uint16_t,
                              scanifc_csz *);
SCANIFC_API int scanifc_rmsstream_cancel(rmsstream_handle);
SCANIFC_API int scanifc_rmsstream_close(rmsstream_handle);
```

Macro SCANIFC_API

SCANIFC_API — Public interface tag.

Synopsis

```
// In header: <riegl/scanifc.h>

SCANIFC_API
```

Function scanifc_get_library_version

scanifc_get_library_version

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_get_library_version(scanifc_uint16_t * major,
                            scanifc_uint16_t * minor,
                            scanifc_uint16_t * build);
```

Description

Get version number from library. This version number usually is different from the version number of the distribution set of the library. This version number is about the API of the shared object (DLL). A change in major version number generally indicates a breaking change in the API or semantics. A change in minor version usually does not require a change of user code.

Parameters:	`build`	build number (revision control)
	`major`	major version number
	`minor`	minor version number
Returns:	0 for success, !=0 for failure	

Function scanifc_get_library_info

scanifc_get_library_info

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_get_library_info(scanifc_csz * build_version, scanifc_csz * build_tag);
```

Description

Get extended version information.

The build_version allows traceability to the SCM system. The build tag contains additional information about the build.

Parameters:	`build_tag`	[out] additional build information
	`build_version`	[out] SCM build version
Returns:	0 for success, !=0 for failure	

Function scanifc_get_last_error

scanifc_get_last_error

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_get_last_error(scanifc_sz message_buffer,
                       scanifc_uint32_t message_buffer_size,
```

```
                        scanifc_uint32_t * message_size);
```

Description

Get last error message. A failing function (indicated by !=0 return) usually will set an error description string, that can be retrieved with this function.

Parameters:	`message_buffer`	[in,out] user supplied buffer for the zero delimited string.
	`message_buffer_size`	[in] size of the user supplied buffer
	`message_size`	[out] size of the message, if larger than message_buffer_size the message is truncated to fit within the buffer.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_open

scanifc_point3dstream_open

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_open(scanifc_csz uri, scanifc_bool sync_to_pps,
                           point3dstream_handle * h3ds);
```

Description

Open 3D pointcloud data stream. The open function returns a handle that references the stream. The stream is read sequentially from start to end.

Parameters:	`h3ds`	[out] a handle identifying this particular stream, for use in the read function.
	`sync_to_pps`	[in] if 0 does not use pps timestamps embedded in the rxp stream. If set to 1 requires pps timestamps.
	`uri`	[in] unified resource identifier of the 'rxp' stream containing the pointcloud data.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_open_with_logging

scanifc_point3dstream_open_with_logging

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_open_with_logging(scanifc_csz uri,
                                        scanifc_bool sync_to_pps,
                                        scanifc_csz log,
                                        point3dstream_handle * h3ds);
```

Description

Open 3D pointcloud data stream with logging. The open function returns a handle that references the stream. The stream is read sequentially from start to end. A log of the data is written to the file system.

Parameters:	`h3ds`	[out] a handle identifying this particular stream, for use in the read function.
	`log`	[in] a filename, where binary data will be logged.
	`sync_to_pps`	[in] if 0 does not use pps timestamps embedded in the rxp stream. If set to 1 requires pps timestamps.
	`uri`	[in] unified resource identifier of the 'rxp' stream containing the pointcloud data.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_open_rms

scanifc_point3dstream_open_rms

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_open_rms(scanifc_csz uri, scanifc_bool sync_to_pps,
                               scanifc_csz log, scanifc_csz jobname,
                               point3dstream_handle * h3ds);
```

Description

Open 3D pointcloud data stream from rms. The open function returns a handle that references the stream. The stream is read sequentially from start to end. A log of the data is written to the file system.

Parameters:	`h3ds`	[out] a handle identifying this particular stream, for use in the read function.
	`jobname`	[in] a name of the job to select from rms file. Optional parameter, zero if not required (the first will be selected).
	`log`	[in] a filename, where binary data will be logged. Optional parameter, zero if not required.
	`sync_to_pps`	[in] if 0 does not use pps timestamps embedded in the rxp stream. If set to 1 requires pps timestamps.
	`uri`	[in] unified resource identifier of the 'rms' stream containing the pointcloud data.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_add_demultiplexer

scanifc_point3dstream_add_demultiplexer

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_add_demultiplexer(point3dstream_handle h3ds,
                                        scanifc_csz filename,
                                        scanifc_csz selections,
                                        scanifc_csz classes);
```

Description

Append a demultiplexer. (Note: don't confuse with rms, which is a different concept.) A demultiplexer will write selected packages from the rxp stream to the file with name 'filename' in ASCII format. The demultiplexer will run while 'read' is executed, so it is necessary to call into read until no more data is available, even if the result values from read are of no interest. The 'selections' or 'classes' may be 0 if not used.

Parameters:	`classes`	[in] a space delim. list of (package) class names.
	`filename`	[in] the name of the target file for this de-multiplexer.
	`h3ds`	[in] the stream heandle that has been returned from open.
	`selections`	[in] a space delim. list of package names to be used.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_set_rangegate

scanifc_point3dstream_set_rangegate

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_set_rangegate(point3dstream_handle h3ds,
                                    scanifc_uint16_t zone,
                                    scanifc_float32_t near,
                                    scanifc_float32_t far);
```

Description

Override the default rangegate of the sensor. This function is only meaningful for sensor types that support "multiple turn around" zones, i.e. the laser clock is faster than the ambiguity range of the rangegate would allow. The ambiguity can be resolved by operator asstitence in specifying the intended range.

Parameters:	`far`	[in] the far end of the range gate (valid if zone=0)
	`h3ds`	[in] the stream heandle that has been returned from open.
	`near`	[in] the closer range of the range gate (valid if zone=0)
	`zone`	[in] a value specifying the MTA zone (0 is special, see below)
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_read

scanifc_point3dstream_read

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_read(point3dstream_handle h3ds, scanifc_uint32_t want,
                           scanifc_xyz32 * pxyz32,
                           scanifc_attributes * pattributes,
                           scanifc_time_ns * ptime, scanifc_uint32_t * got,
                           scanifc_bool * end_of_frame);
```

Description

Read some points from the stream. The read function will fill point data into user supplied buffers. The buffer pointers may be zero, in which case this particular buffer will not be filled. After the end of a frame it is possible to call into read again to obtain the next frame. The end of all available data is reached when both: "got" and "end_of_frame" are zero at the same time.

Parameters:	`end_of_frame`	[out] != 0 if end of frame detected
	`got`	[out] number of points returned (may be smaller than want).

	`h3ds`	[in] the stream heandle that has been returned from open.
	`pattributes`	[out] pointer to amplitude and quality buffer
	`ptime`	[out] pointer to timestamp buffer
	`pxyz32`	[out] pointer to xyz buffer
	`want`	[in] the size of the result buffers (count of points).
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_tellg

scanifc_point3dstream_tellg

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_point3dstream_tellg(point3dstream_handle h3ds, scanifc_uint64_t * pos);
```

Description

Total number of octets already read. This function may be used for debugging or as a progress indicator if reading from files for which the size is known in advance.

Parameters:	`h3ds`	[in] the stream heandle that has been returned from open.
	`pos`	[out] number of octets
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_cancel

scanifc_point3dstream_cancel

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int scanifc_point3dstream_cancel(point3dstream_handle h3ds);
```

Description

Cancel a blocking read. This function needs to be called from a different thread.

Parameters:	`h3ds`	[in] the stream heandle that has been returned from open.
Returns:	0 for success, !=0 for failure	

Function scanifc_point3dstream_close

scanifc_point3dstream_close

Synopsis

```
// In header: <riegl/scanifc.h>
```

```
SCANIFC_API int scanifc_point3dstream_close(point3dstream_handle h3ds);
```

Description

Close a pointcloud stream. This function must be called when done with the stream to release the resources associated with the handle.

Parameters: `h3ds` [in] the stream heandle that has been returned from open.
Returns: 0 for success, !=0 for failure

Function scanifc_rmsstream_open

scanifc_rmsstream_open

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_rmsstream_open(scanifc_csz uri, scanifc_bool sync_to_pps,
                       scanifc_uint16_t jobs_size, scanifc_csz * jobs,
                       rmsstream_handle * hrms);
```

Description

Open 3D pointcloud data stream from an rms stream. The open function returns a handle that references the stream. The stream is read sequentially from start to end.

Parameters:

`hrms`	[out] a handle identifying this particular stream, for use in the read function.
`jobs`	[in] array of size jobs_size with job names.
`jobs_size`	[in] number of jobs descriptors. Zero selects all jobs.
`sync_to_pps`	[in] if 0 does not use pps timestamps embedded in the rxp stream. If set to 1 requires pps timestamps.
`uri`	[in] unified resource identifier of the 'rms' stream containing the pointcloud data.

Returns: 0 for success, !=0 for failure

Function scanifc_rmsstream_read

scanifc_rmsstream_read

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_rmsstream_read(rmsstream_handle hrms, scanifc_uint32_t want,
                       scanifc_xyz32 * pxyz32,
                       scanifc_attributes * pattributes,
                       scanifc_time_ns * ptime, scanifc_uint32_t * got,
                       scanifc_bool * end_of_frame, scanifc_uint16_t * jobnr);
```

Description

Read some points from the stream. The read function will fill point data into user supplied buffers. The buffer pointers may be zero, in which case this particular buffer will not be filled.

Parameters:	`end_of_frame`	[out] != 0 if end of frame detected
	`got`	[out] number of points returned (may smaller than want).
	`hrms`	[in] the stream heandle that has been returned from open.
	`jobnr`	[out] job number
	`pattributes`	[out] pointer to amplitude and quality buffer
	`ptime`	[out] pointer to timestamp buffer
	`pxyz32`	[out] pointer to xyz buffer
	`want`	[in] the size of the result buffers (count of points).
Returns:	0 for success, !=0 for failure	

Function scanifc_rmsstream_tellg

scanifc_rmsstream_tellg

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_rmsstream_tellg(rmsstream_handle hrms, scanifc_uint64_t * pos);
```

Description

Total number of octets already read. This function may be used for debugging or as a progress indicator if reading from files for which the size is known in advance.

Parameters:	`hrms`	[in] the stream heandle that has been returned from open.
	`pos`	[out] number of octets
Returns:	0 for success, !=0 for failure	

Function scanifc_rmsstream_get_jobname

scanifc_rmsstream_get_jobname

Synopsis

```
// In header: <riegl/scanifc.h>


SCANIFC_API int
scanifc_rmsstream_get_jobname(rmsstream_handle hrms, scanifc_uint16_t jobnr,
                              scanifc_csz * jobname);
```

Description

Get job name. This function may be called after scanifc_rmsstream_read has dicovered a new jobnr.

Parameters:	`hrms`	[in] the stream heandle that has been returned from open.
	`jobname`	[out] jobname
	`jobnr`	[in] index of desired jobname
Returns:	0 for success, !=0 for failure	

Function scanifc_rmsstream_cancel

scanifc_rmsstream_cancel

Synopsis

```
// In header: <riegl/scanifc.h>

SCANIFC_API int scanifc_rmsstream_cancel(rmsstream_handle hrms);
```

Description

Cancel a blocking read. This function needs to be called from a different thread.

Returns: 0 for success, !=0 for failure

Function scanifc_rmsstream_close

scanifc_rmsstream_close

Synopsis

```
// In header: <riegl/scanifc.h>

SCANIFC_API int scanifc_rmsstream_close(rmsstream_handle hrms);
```

Description

Close a rms stream. This function must be called when done with the stream to release the resources associated with the handle.

Returns: 0 for success, !=0 for failure

Ctrlifc

Ctrlifc Usage

This part of the library offers the same functionality of ctrllib but packed in a language neutral interface because not every application uses the C++ language but still needs to configure and control the scanner device.

Ctrlifc Reference

Header <riegl/ctrlifc.h>

The Riegl Scanner Control (DLL) interface definitions.

Basic data types used in the DLL interface.

```
CTRLIFC_API
```

```
typedef IMPLEMENTATION_DEFINED ctrlifc_uint16_t;  // unsigned 16 bit integer
typedef IMPLEMENTATION_DEFINED ctrlifc_int32_t;
typedef IMPLEMENTATION_DEFINED ctrlifc_session_handle;  // a handle to a stream of 3d point-
cloud data
typedef char * ctrlifc_sz;  // A zero delimited string.
typedef char ** ctrlifc_szlist;  // A list of zero delimited strings.
typedef const char * ctrlifc_csz;  // A zero delimited constant string.
typedef const char ** ctrlifc_cszlist;  // A list of zero delimited string constants.
CTRLIFC_API int
ctrlifc_get_library_version(ctrlifc_uint16_t *, ctrlifc_uint16_t *,
                            ctrlifc_uint16_t *);
CTRLIFC_API int ctrlifc_get_library_info(ctrlifc_csz *, ctrlifc_csz *);
CTRLIFC_API int ctrlifc_construct_session(ctrlifc_session *);
CTRLIFC_API int ctrlifc_destruct_session(ctrlifc_session);
CTRLIFC_API int ctrlifc_get_last_error(ctrlifc_session, ctrlifc_csz *);
CTRLIFC_API int ctrlifc_connect(ctrlifc_session, ctrlifc_csz);
CTRLIFC_API int ctrlifc_disconnect(ctrlifc_session);
CTRLIFC_API int
ctrlifc_set_property(ctrlifc_session, ctrlifc_csz, ctrlifc_csz,
                     ctrlifc_cszlist *, ctrlifc_int32_t *, ctrlifc_csz *);
CTRLIFC_API int
ctrlifc_get_property(ctrlifc_session, ctrlifc_csz, ctrlifc_csz *,
                     ctrlifc_cszlist *, ctrlifc_int32_t *, ctrlifc_csz *);
CTRLIFC_API int
ctrlifc_execute_command(ctrlifc_session, ctrlifc_csz, ctrlifc_csz,
                        ctrlifc_csz *, ctrlifc_cszlist *, ctrlifc_int32_t *,
                        ctrlifc_csz *);
```

Macro CTRLIFC_API

CTRLIFC_API — Public interface tag.

Synopsis

```
// In header: <riegl/ctrlifc.h>

CTRLIFC_API
```

Function ctrlifc_get_library_version

ctrlifc_get_library_version

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_get_library_version(ctrlifc_uint16_t * major,
                            ctrlifc_uint16_t * minor,
                            ctrlifc_uint16_t * build);
```

Description

Get version number from library.

This version number usually is different from the version number of the distribution set of the library. This version number is about the API of the shared object (DLL). A change in major version number generally indicates a breaking change in the API or semantics. A change in minor version usually does not require a change of user code.

Parameters:	`build`	build number (revision control)
	`major`	major version number
	`minor`	minor version number
Returns:	0 for succes, !=0 for failure	

Function ctrlifc_get_library_info

ctrlifc_get_library_info

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_get_library_info(ctrlifc_csz * build_version, ctrlifc_csz * build_tag);
```

Description

Get extended version information.

The build_version allows traceability to the SCM system. The build tag contains additional information about the build.

Parameters:	`build_tag`	[out] additional build information
	`build_version`	[out] SCM build version
Returns:	0 for succes, !=0 for failure	

Function ctrlifc_construct_session

ctrlifc_construct_session

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int ctrlifc_construct_session(ctrlifc_session * s);
```

Description

Create a scanner control session.

The create_session function returns a handle that references the session.

Parameters: s [out] a handle identifying this particular session.
Returns: 0 for success, !=0 for failure

Function ctrlifc_destruct_session

ctrlifc_destruct_session

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int ctrlifc_destruct_session(ctrlifc_session s);
```

Description

Close a session.

This function must be called when done with the ssession to release the resources associated with the handle.

Parameters: s [in] the ssession heandle that has been returned from construct.
Returns: 0 for success, !=0 for failure

Function ctrlifc_get_last_error

ctrlifc_get_last_error

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_get_last_error(ctrlifc_session s, ctrlifc_csz * message_buffer);
```

Description

Get last error message.

A failing function (indicated by !=0 return) usually will set an error description string, that can be retrieved with this function.

Parameters: s [in] the session handle
Returns: 0 for succes, !=0 for failure

Function ctrlifc_connect

ctrlifc_connect

Synopsis

```
// In header: <riegl/ctrlifc.h>
```

```
CTRLIFC_API int ctrlifc_connect(ctrlifc_session s, ctrlifc_csz authority);
```

Description

Establish a session to a scanner.

The connect function will locate the scanner by means of the authority information, which may be given as "host:port", where host and port either are in text or numeric forms.

Parameters:	`authority`	[in] the address of the scanner
	`s`	[in] the session handle
Returns:	0 for success, !=0 for failure	

Function ctrlifc_disconnect

ctrlifc_disconnect

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int ctrlifc_disconnect(ctrlifc_session s);
```

Description

Close connection to scanner.

The disconnect function will close the connection to the scanner.

Parameters:	`s`	[in] the session handle
Returns:	0 for success, !=0 for failure	

Function ctrlifc_set_property

ctrlifc_set_property

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_set_property(ctrlifc_session s, ctrlifc_csz property,
                     ctrlifc_csz value, ctrlifc_cszlist * remarks,
                     ctrlifc_int32_t * remarks_count, ctrlifc_csz * status);
```

Description

Set a property value.

The set_property function sets a new value for a property. The new value is given as string value:

- string properties are to be enclosed in quotation marks "..."
- integer and floating point values must be converted into a string

- boolean properties are 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:		
	`remarks`	[out] received comment lines from instrument (optional).
	`remarks_count`	[out] number of comment lines from instrument.
	`s`	[in] the session handle
	`status`	[out] control port status (optional). Format is "<...>".
	`value`	[in] the new property value

Function ctrlifc_get_property

ctrlifc_get_property

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_get_property(ctrlifc_session s, ctrlifc_csz property,
                     ctrlifc_csz * value, ctrlifc_cszlist * remarks,
                     ctrlifc_int32_t * remarks_count, ctrlifc_csz * status);
```

Description

Get a property value.

The get_property function returns the value of a property. The value is returned as string value:

- string properties are enclosed in quotation marks "..."
- integer and floating point values must be converted
- boolean properties are strings 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:		
	`remarks`	[out] received comment lines from instrument (optional).
	`remarks_count`	[out] number of comment lines from instrument.
	`s`	[in] the session handle
	`status`	[out] control port status (optional). Format is "<...>".
	`value`	[out] the property value as string

Function ctrlifc_execute_command

ctrlifc_execute_command

Synopsis

```
// In header: <riegl/ctrlifc.h>


CTRLIFC_API int
ctrlifc_execute_command(ctrlifc_session s, ctrlifc_csz command,
                        ctrlifc_csz arguments, ctrlifc_csz * result,
                        ctrlifc_cszlist * remarks,
```

```
                        ctrlifc_int32_t * remarks_count,
                        ctrlifc_csz * prompt);
```

Description

Execute a command (function call).

The execute_command function calls a function of the scanner interface. The value is returned as string value:

- string properties are enclosed in quotation marks "..."
- integer and floating point values must be converted
- boolean properties are strings 1 (TRUE) and 0 (FALSE)
- vectors elements are separated using , (comma)

Parameters:		
	`arguments`	[in] the arguments for the function. Specify "" for no arguments to be passed to command.
	`command`	[in] the name of the command/function to execute/call
	`remarks`	[out] received comment lines from instrument (optional).
	`remarks_count`	[out] number of comment lines from instrument.
	`result`	[out] result value as string (optional). Vectors are returned as value list separated by , (comma). In case of failure, this parameter contains the error message.
	`s`	[in] the session handle

Part IV. Tools

Table of Contents

The rxpexcise program

Usage

rxpexcise is a command line oriented executable to excise part of data from a large rxp file based on timestamp information and store the result into a new rxp file.

Three different time bases are avilable for cutting: internal (relative) system time, real time clock chip of the scanner and external gps synchronized time (when enabled in the scanner).

The argument to the `--cut` option is a comma separated list of floating point values specifying cutting points. The physical unit of these values is a second. It is possible to specify more than one cut sequence, each going to a different output file. The first value is the time instance where output is turned off. The next value turns it on again. This sequence can be repeated, so to excise from different sections of the original file. If the first value is zero this has the special meaning that the output is turned of from the very beginning of the data stream.

The command invocation syntax is

`rxpexcise` [*`-shortopt`* [*`arg`*] | *`--longopt`* [*`=arg`*]]... *`uri`*
where options can be given in their long or short form. Some options have arguments and/or default values for their arguments. Some options can appear more than once. The `uri` is a Uniform Resource Identifier as defined in this document.

Tip
The following list of options can also be obtained with **rxpexcise --help**.

```
--version                     show version
-h [ --help ]                 show help
-o [ --out-file ] arg         output file
--cut arg                     comma separated list of times in seconds
                            first value switches off, next on, ...
-t [ --timebase ] arg (=sys) arg is one of:
                           sys       systime time stamps
                           pps       pps (GPS time)
                           rtc       scanner real time clock
```

Examples

Display the pps time stamps available in the data stream.

rxpexcise -tpps file:localfile.rxp

Extract from pps second 85487 to 85490 to first file and from second 85492 to 85494 to second file.

rxpexcise -tpps file:localfile.rxp -ocutted.rxp --cut=0,85487,85490 --cut=0,85492,85494

The rft program

Usage

rft is a command line oriented executable to perform file transfers between a client computer and a V-Line scanner device, making use of the RDTP protocol.

The commands are fully resumable, i.e. if disrupted for some reason during a transfer, giving the very same command after restoration of the systems will resume operation without transfering the already received data again. Of courese the already received data must still be in the same place where it has been before disruption.

The command invocation syntax is

`rft` *`subcommand`*
where subcommand is one of **ls**, **cp**, **mkdir**, **rm** and **glob**. The subcommands each have their own syntax. To get information about them use **rft subcommand --help**. The subcommands are similar to the corresponding posix commands of the same name. The **glob** subcommand is of use when a plain list of files matching a particular filename pattern is desired.

Tip
Use **rft --help** or **rft subcommand --help** to get more information.

Examples

Make a backup copy of the scanner to local directory `backup/`

rft cp -r rft://192.168.0.42 backup/

Copy license option files to the scanner.

rft cp option.txt option.txt.sig /intern/licenses/

List the contents of the /usb subdirectory on the scanner.

rft ls rft://192.168.0.42/intern

Glossary

RXP, Riegl Extended Packages	A stream format for tagged variably sized binary data.
RMS, Riegl Multiplexed Stream	An interleaved (multiplexed) stream of multiple RXP streams, for the purpose of optimizing seeking times of hard disks.
RDTP, Riegl Data Transfer Protocol	A data transfer protocol for encapsulation of RXP streams, with with resume capability and command embedding.

Appendix A. Legal notice

Appendix B. Revision history

Revision History

Revision 1.13 2013-06-11

Updated for 1.30 library version. Changes in supported compilers and library component layout.

Revision 1.12 2012-01-24

Updated for 1.24 library version. MTA processing.

Revision 1.11 2011-01-17

Updated for 1.11 library version.

Revision 1.4 2010-06-08

Updated for 1.5 library version.

Revision 1.3 2010-02-09

Addition of rms support documentation.

Revision 1.2 2010-02-02

Fixup of minor documentation errors.

Revision 1.1 2009-11-11

Addition of control library.

Revision 1.0 2009-11-04

First Release.

Revision 0.1 2008-12-18

First draft.