3DSS-DX Structure API

0.6

Generated by Doxygen 1.8.5

Wed Nov 9 2016 11:33:29

Contents

Chapter 1

3DSS-DX Structure API

This document describes the structure format output by the 3DSS-DX control software over TCP.

Date

2016 11 09

Version

0.6

Author

Mark Butowski support@pingdsp.com

1.1 DESCRIPTION

Provides C++ data format (structure) definitions for extraction of 3DSS-DX data from a TCP stream generated by a 3DSS-DX sonar.

Data is extracted with two successive TCP stream reads. The first read is a header (DxHeader) with a preamble and the data length in bytes to follow on the next stream read. The second read is the data buffer (DxData) and begins with an identifier (ping count) and timestamps and then provides offset and count information for 6 included data sections.

The data sections include all sonar settings used during data collection, all ASCII (e.g. NMEA and TSS1) strings acquired during the ping (+/-100ms) together with port and starboard sidescan data and sidescan3D data.

Compiler verification functions at the end of this file check for correct structure sizes in bytes.

Note

Only supports little endian machines! (ie x86,x64)

1.2 CHANGES

Version	Date	Release Notes
0.1	2013.11.29	initial release
0.2	2014.02.25	documentation updates
		- fixed reversed comments in SidescanPoint3D
		- fixed angle comment to downward angles as negative
0.3	2014.06.09	added requested info by 3rd parties and futureproofed
		- now providing pingrate
		- time at the end of the ping (useful for selecting
		ascii sentences that arrived during ping)
		- reserved spot for quality factor in 3d points
		- reserved spots for more vector data outputs
		- added system sample rate
		- changed preamble since the changes are breaking
0.4	2015.02.06	documentation updates
		- added definition of NMEA sentences typically provided
		by the sonar
		- added definition of TSS1 sentence
0.5	2015.08.25	added bathymetry points to structure
		- range, angle, amplitude generated from bottom tracked
		and binned sidescan-3d data
0.6	2016.11.09	added recorded to filename and 3DSS control version

1.3 DEMO

Below is a simple demo for using this file:

```
*  int main() {
*
*    using namespace softsonar;
*
*
*    // Create two buffers for the header and the data.
*    char* headerBuf = new char[sizeof(dx::DxHeader)];
*    char* dataBuf = new char[4*1024*1024];
*
*    // Overlay structures over the buffers.
*    dx::DxHeader* header = reinterpret_cast<dx::DxHeader*>(&headerBuf[0]);
*    dx::DxData* data = reinterpret_cast<dx::DxData*>(&dataBuf[0]);
*
*    // Use TcpClient (c++ class that helps with Tcp socket connections)
*    TcpClient client("127.0.0.1", dx::kTcpPort);
*
*    // Connect to localhost on port kTcpPort.
*    bool success = false;
*    success = client.Connect();
*
*    if (!success) {
*      std::cout << "Could not connect!" << std::endl;
*       return -1;
*     }
*
*     // Read pings (infinite).
*     for(;;) {
*
*       // Read the header
*       success = client.Recv(headerBuf, sizeof(dx::DxHeader));
*
*       if (!success || !header->is_preamble_valid()) {
*         std::cout << "Failed to receive header!" << std::endl;
*         return -2;
*       }
*
*       // Read the data
*       success = client.Recv(dataBuf, header->data_count);
*
*       if (!success) {
*         std::cout << "Failed to receive data!" << std::endl;
*         return -3;
*       }
*
*       // Print out the ping id (number)
*       std::cout << "Ping No: " << data->id << std::endl;
*
*       // Print out the time of the ping
*       std::cout << "Time: " << data->time.seconds << "s "
*                 << data->time.nanoseconds << "ns" << std::endl;
*
*       // Get the ASCII sentences from the data and print first one.
*       std::vector<common::AsciiSentence> sentences = data->get_ascii_sentences();
*
*       if (sentences.size() > 0) {
*         std::cout << "First sentence: " << sentences[0].sentence << std::endl;
*       }
*
*       // Get the points from the data and print out the number of points.
*       std::vector<common::SidescanPoint> portSidescan = data->get_port_sidescan_points();
*       std::vector<common::Sidescan3DPoint> portSidescan3 = data->get_port_sidescan3d_points();
*       std::vector<common::SidescanPoint> stbdSidescan = data->get_starboard_sidescan_points();
*       std::vector<common::Sidescan3DPoint> stbdSidescan3 = data->get_starboard_sidescan3d_points();
*
*       std::cout << "No. of points: " << std::setw(10);
*       std::cout << portSidescan.size();
*       std::cout << portSidescan3.size();
*       std::cout << stbdSidescan.size();
*       std::cout << stbdSidescan3.size();
*
*     }
*
*     // Clean up.
*     client.Disconnect();
*
*     delete[] headerBuf;
*     delete[] dataBuf;
*
*     return 0;
*
*   }
*
*
```

1.4 LICENSE

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 AsciiSentence Struct Reference

Represents an ascii sentence received by the sonar.

Public Attributes

- Timestamp timestamp
- char8_t sentence [256]
- uint32_t source
- uint32_t port

4.1.1 Detailed Description

Represents an ascii sentence received by the sonar.

A four part structure including a timestamp, an ascii string, a source identifier, and a port identifier.

The Ascii string may originate from one of three types received by the sonar: a) NMEA sentence that begins with the '$' character, b) TSS1 sentence that begins with the ':' character, c) TSS3 sentence that begins with the ':' character.

In all cases the <CR><LF> expected at the end of each sentence received by the sonar is removed. All other parts of the sentence received by the sonar are preserved and may be parsed as if the sentence originated directly from the underlying sensor, including externally connected sensors (e.g. MRU or GPS) as well as internal sonar sensors such as the internal MRU, and the water temperature sensor.

As an example, the internal sonar sensors produce NMEA strings with the following NMEA sentence formats:

1. Internal MRU sensor sentence format:

```
*          $VNYCM,+YYY.YYY,+PPP.PPP,+RRR.RRR,+mx.mmmm,+my.mmmm,+mz.mmm,
*                  +ax.aaa,+ay.yyy,+az.aaa,+gx.gggggg,+gy.gggggg,+gz.gggggg,
*                  +TTT.T*cc
*
```

   where:

   - YYY.YYY = Yaw in deg
   - PPP.PPP = Pitch in deg

- RRR.RRR = Roll in deg
- TTT.T = Temperature in degrees Celsius
- cc = Checksum

2. Internal Water Temperature sensor sentence format:

```
*         $PDWTR,sss-ssssssss,hhhhhh,A,ppppp.p,O,ww.ww,C,*,cc
*
```

   where:

   - ww.ww = Temperature in degrees Celsius
   - cc = Checksum

3. $–RMC Sentence, Recommended Miniumu GPS:

```
*       $--RMC,hhmmss.ss,A,ddmm.mmm,n,ddmm.mmm,w,x.x,h.h,ddmmyy,m.m,v*cc
*
```

   where:

   - hhmmss.ss = UTC time in hours, minutes, and seconds of the GPS position
   - A = Status (A = valid, V = invalid)
   - ddmm.mmm = Latitude in degrees, minutes, and decimal minutes
   - ns = Latitude location (N = North latitude, S = South latitude)
   - ddmm.mmm = Longitude in degrees, minutes, and decimal minutes
   - w = Longitude location (E = East longitude, W = West longitude)
   - x.x = Speed over ground in knots
   - h.h = Track made good, reference to true north
   - ddmmyy = UTC date of position fix in day, month, and year
   - m.m = Magnetic variation in degrees
   - v = Variation sense (E = East, W = West)

4. $–HDT Sentence, Heading True:

```
*       $--HDT,x.x,T*cc
*
```

   where:

   - x.x = Current heading in degrees
   - T = Indicates true heading

5. $–GGA Sentence, Global Positioning System Fix Data:

```
*       $--GGA,hhmmss.ss,ddmm.mmmmm,k,ddmm.mmmmmm,l,n,qq,pp.p,aaaa.aa,m,gg.gg,m,sss,rrrr*cc
*
```

   where:

   - hhmmss.ss = UTC time in hours, minutes, and seconds of the GPS position
   - ddmm.mmmmm = Latitude in degrees, minutes, and decimal minutes
   - k = Latitude location (N = North latitude, S = South latitude)
   - ddmm.mmmmm = Longitude in degrees, minutes, and decimal minutes

- l = Longitude location (E = East longitude, W = West longitude)
- n = Quality indicator
- qq = Number of satellites used in position solution
- pp.p = Horizontal dilution of precision (HDOP)
- aaaa.aa = Antenna altitude, in meters, re: mean-sea-level (geoid)
- m = Units of antenna altitude (M = meters)
- gg.gg = Geoidal separation (in meters)
- m = Units of geoidal separation (M = meters)
- sss = Age of differential corrections, in seconds
- rrrr = Differential reference station ID

6. $–VTG Sentence, Track Made Good and Ground Speed:

```
*       $--VTG,ttt,T,mmm,M,nnn.nn,N,kkk.kk,K,x*cc
*
```

where:

- ttt = True course over ground (COG) in degrees (000 to 359)
- T = True course over ground indicator (always 'T')
- mmm = Magnetic course over ground in degrees (000 to 359)
- M = Magnetic course over ground indicator (always 'M')
- nnn.nn = Speed over ground in knots
- N = Speed over ground in knots indicator (always 'N')
- kkk.kk = Speed over ground in km/h
- K = Speed over ground in km/h indicator (always 'K')
- x = Mode

7. TSS1 Sentence:

```
*       :XXAAAASMHHHHQMRRRRSMPPPP
*
```

where:

- XX = Horizontal acceleration (hex value), in 3.83 cm/s^2, with a range of zero to 9.81 m/s^2
- AAAA = Vertical acceleration (hex value - 2's complement), in 0.0625 cm/s^2, with a range of –20.48 to +20.48 m/s^2
- S = Space character
- M = Space if positive; minus if negative
- HHHH = Heave, in centimeters, with a range of –99.99 to +99.99 meters
- Q = Status Flag
- M = Space if positive; minus if negative
- RRRR = Roll, in units of 0.01 degrees (ex: 1000 = 10 degrees), with a range of –99.99 to +99.99 degrees
- S = Space character
- M = Space if positive; minus if negative
- PPPP = Pitch, in units of 0.01 degrees (ex: 1000 = 10 degrees), with a range of –99.99 to +99.99 degrees

Since

0.1

4.1.2 Member Data Documentation

4.1.2.1 uint32_t port

An identifier indicating the sonar serial port origin of the sentence.

4.1.2.2 char8_t sentence[256]

The Ascii sentence beginning with either a '$'or ':' character. The character array is null terminated without <CR><L-F>.

4.1.2.3 uint32_t source

An identifier indicating the sonar system origin of the sentence.

4.1.2.4 Timestamp timestamp

The sonar time when first character of the sentence was received by the sonar.

4.2 BathymetryPoint Struct Reference

Represents a bathymetry point.

Public Attributes

- float32_t range
- float32_t angle
- float32_t amplitude
- float32_t reserved1
- float32_t reserved2

4.2.1 Detailed Description

Represents a bathymetry point.

A three part structure including the bathymetry data values, range, amplitude, and angle.

Since

0.5

4.2.2 Member Data Documentation

4.2.2.1 float32_t amplitude

The amplitude of the bathymetry data point after gain and bathymetry processing methods are applied.

4.2.2.2 float32_t angle

The angle in radians of the bathymetry data point. Negative angles are downward from the maximum response axis of the transducer.

4.2.2.3 float32_t range

The range in meters to the bathymetry data point.

4.2.2.4 float32_t reserved1

Reserved for a future value, either SNR or quality factor for the point.

4.2.2.5 float32_t reserved2

Reserved for a future value, either SNR or quality factor for the point.

4.3 Gain Struct Reference

Represents the gain controlling coefficients used by the sonar.

Public Attributes

- float32_t constant
- float32_t linear
- float32_t logarithmic

4.3.1 Detailed Description

Represents the gain controlling coefficients used by the sonar.

A three part structure including gain coefficients that vary with range logarithmically, linearly and not at all. The gain applied for each sample is determined by the range of the sample and the summation of the gain contributions computed from all three coefficients.

The overall gain in dB is computed as: G = constant + linear$*$Range + logarithmic$*$log10(Range)

Since

0.1

4.3.2 Member Data Documentation

4.3.2.1 float32_t constant

Coefficient that specifies a constant or fixed gain in dB.

4.3.2.2 float32_t linear

Coefficient that specifies a gain that varies linearly with range in dB/m.

4.3.2.3 float32_t logarithmic

Coefficient that specifies a gain that varies logarithmically with range in dB/log10(m).

4.4 Sidescan3DPoint Struct Reference

Represents a sidescan 3d point.

Public Attributes

- float32_t range
- float32_t angle
- float32_t amplitude
- float32_t reserved

4.4.1 Detailed Description

Represents a sidescan 3d point.

A three part structure including the sidescan3D data values, range, amplitude, and angle.

Since

0.1

4.4.2 Member Data Documentation

4.4.2.1 float32_t amplitude

The amplitude of the sidescan-3D data point after gain and sidescan 3D processing methods are applied.

4.4.2.2 float32_t angle

The angle in radians of the sidescan-3D data point. Negative angles are downward from the maximum response axis of the transducer.

4.4.2.3 float32_t range

The range in meters to the sidescan-3D data point.

4.4.2.4 float32_t reserved

Reserved for a future value, either SNR or quality factor for the point.

Since

0.3

4.5 Sidescan3DSettings Struct Reference

Represents the sidescan-3d settings used by the sonar.

Public Attributes

- int32_t smoothing
- float32_t tolerance
- float32_t threshold
- int32_t number_of_angles

4.5.1 Detailed Description

Represents the sidescan-3d settings used by the sonar.

A four part structure including the sidescan3D settings, smoothing, tolerance, threshold, and number_of_angles.

Since

0.1

4.5.2 Member Data Documentation

4.5.2.1 int32_t number_of_angles

The number of angles to compute at each range step.

4.5.2.2 int32_t smoothing

The number of receive samples to include in angle computations.

4.5.2.3 float32_t threshold

The receive signal level threshold in dB re FS (fullscale) for angle computations.

4.5.2.4 float32_t tolerance

The tolerance for non-plane wave arrivals in angle computations.

4.6 SidescanPoint Struct Reference

Represents a sidescan point.

Public Attributes

- float32_t range
- float32_t amplitude

4.6.1 Detailed Description

Represents a sidescan point.

A two part structure including the sidescan data values, range and amplitude.

Since

0.1

4.6.2 Member Data Documentation

4.6.2.1 float32_t amplitude

The amplitude of the sidescan data point.

Amplitudes represent the magnitude of the received signal envelope after gain and sidescan processing methods are applied.

4.6.2.2 float32_t range

The range in meters to the sidescan data point.

4.7 SidescanSettings Struct Reference

Represents the sidescan settings used by the sonar.

Public Attributes

- char8_t mode [32]
- char8_t incoherent_method [32]
- char8_t coherent_beams [64]

4.7.1 Detailed Description

Represents the sidescan settings used by the sonar.

A three part structure including the sidescan mode, and mode specific settings.

Since

0.1

4.7.2 Member Data Documentation

4.7.2.1 char8_t coherent_beams[64]

The vertical sidescan beams used in coherent sidescan mode, specified as a list of beam names.

4.7.2.2 char8_t incoherent_method[32]

The method of combining array elements for operation in incoherent mode, specified by name.

4.7.2.3 char8_t mode[32]

The sidescan mode of operation, specified by name.

4.8 SoundVelocity Struct Reference

Represents the speed of sound values in m/s used by the sonar.

Public Attributes

- float32_t bulk
- float32_t face

4.8.1 Detailed Description

Represents the speed of sound values in m/s used by the sonar.

A two part data structure including bulk and face sound velocities expressed in meters per second.

Since

0.1

4.8.2 Member Data Documentation

4.8.2.1 float32_t bulk

Bulk (average) water column sound velocity in m/s.

4.8.2.2 float32_t face

Single point sound velocity at the sonar transducer front face in m/s.

4.9 Timestamp Struct Reference

Represents the elapsed time since January 1, 1970 12:00 AM.

Public Attributes

- uint64_t seconds
- uint32_t nanoseconds
- uint32_t flags

4.9.1 Detailed Description

Represents the elapsed time since January 1, 1970 12:00 AM.

A three part structure including seconds, nanoseconds and a set of status flags.

Since

0.1

4.9.2 Member Data Documentation

4.9.2.1 uint32_t flags

Proprietary status flags.

4.9.2.2 uint32_t nanoseconds

Nanoseconds (remainder from seconds).

4.9.2.3 uint64_t seconds

Seconds since January 1, 1970 12:00 AM.

4.10 TransmitSettings Struct Reference

Represents the transmit settings used by the sonar.

Public Attributes

- float32_t angle
- uint32_t power
- char8_t beamwidth [32]
- char8_t pulse [32]

4.10.1 Detailed Description

Represents the transmit settings used by the sonar.

A four part structure including the sonar transmit settings, angle, power, beamwidth, and pulse.

Since

0.1

4.10.2 Member Data Documentation

4.10.2.1 float32_t angle

The transmit beam angle (direction) in degrees relative to the maximum response axis of the transducer. Positive angles are downward.

4.10.2.2 char8_t beamwidth[32]

The transmit beamwidth (beampattern) specified by name.

4.10.2.3 uint32_t power

The transmit power as a percentage of maximum.

4.10.2.4 char8_t pulse[32]

The transmit pulse waveform specified by name.

4.11 TriggerSettings Struct Reference

Represents the trigger settings used by the sonar.

Public Attributes

- char8_t source [32]
- float32_t countinuousdutycycle
- float32_t reserved

4.11.1 Detailed Description

Represents the trigger settings used by the sonar.

A two part structure including the sonar trigger settings, source and continuousdutycycle.

Since

0.1

4.11.2 Member Data Documentation

4.11.2.1 float32_t countinuousdutycycle

The duty cycle specifying the trigger repetition rate as a fraction of the maximum rate for the current sonar range setting. Only applicable when the trigger source is set to Continuous.

4.11.2.2 float32_t reserved

Reserved - likely ping rate in the future.

Since

0.3

4.11.2.3 char8_t source[32]

The trigger source used by the sonar specified by name.

4.12 DxData Struct Reference

Represents all data and sonar settings for an entire sonar ping.

Public Member Functions

- const common::AsciiSentence ∗ get_ascii_sentence_ptr () const
- int get_ascii_sentence_count () const
- std::vector
  < common::AsciiSentence > get_ascii_sentences () const
- const common::SidescanPoint ∗ get_port_sidescan_point_ptr () const
- int get_port_sidescan_point_count () const
- std::vector
  < common::SidescanPoint > get_port_sidescan_points () const
- const common::SidescanPoint ∗ get_starboard_sidescan_point_ptr () const
- int get_starboard_sidescan_point_count () const
- std::vector
  < common::SidescanPoint > get_starboard_sidescan_points () const
- const common::Sidescan3DPoint ∗ get_port_sidescan3d_point_ptr () const
- int get_port_sidescan3d_point_count () const
- std::vector
  < common::Sidescan3DPoint > get_port_sidescan3d_points () const
- const common::Sidescan3DPoint ∗ get_starboard_sidescan3d_point_ptr () const
- int get_starboard_sidescan3d_point_count () const
- std::vector
  < common::Sidescan3DPoint > get_starboard_sidescan3d_points () const
- const common::BathymetryPoint ∗ get_port_bathymetry_point_ptr () const
- int get_port_bathymetry_point_count () const
- std::vector
  < common::BathymetryPoint > get_port_bathymetry_points () const

- const common::BathymetryPoint ∗ get_starboard_bathymetry_point_ptr () const
- int get_starboard_bathymetry_point_count () const
- std::vector
  < common::BathymetryPoint > get_starboard_bathymetry_points () const
- std::string get_recorded_filename () const
- std::string get_recorded_version () const

Public Attributes

- uint64_t id
- common::Timestamp time
- common::Timestamp time_range_zero
- dx::DxParameters parameters
- dx::DxSystemInfo system_info

4.12.1 Detailed Description

Represents all data and sonar settings for an entire sonar ping.

A multi-part function and data structure describing the entire ping. It includes the data id (ping number), the data timestamp, the range zero timestamp, the sonar settings, and access functions for retreiving data from the following substructures:

1. Ascii sentences
2. Port sidescan data points
3. Starboard sidescan data points
4. Port Sidescan3D data points
5. Starboard Sidescan3D data points

Since

0.1

4.12.2 Member Function Documentation

4.12.2.1 int get_ascii_sentence_count ( ) const [inline]

Number of ascii sentences present.

4.12.2.2 const common::AsciiSentence∗ get_ascii_sentence_ptr ( ) const [inline]

Zero length array (can be omitted). The first variable length data section starts here (typically the AsciiSentences, ie ascii_sentence_offset should point to this) Pointer to the start/first ascii sentence within the structure.

4.12.2.3 std::vector< common::AsciiSentence > get_ascii_sentences ( ) const [inline]

Helper method which returns the Ascii sentence data as a std::vector.

Note

Typically a compiler will automatically return value optimize and inline this code. Hence its recommended to use this std::vector over directly using the ascii_sentence_ptr() and sentence_count().

4.12.2.4 int get_port_bathymetry_point_count ( ) const [inline]

Number of port-side bathymetry points present.

Since

0.5

4.12.2.5 const common::BathymetryPoint* get_port_bathymetry_point_ptr ( ) const [inline]

Pointer to the start/first port-side bathymetry point within the structure.

Since

0.5

4.12.2.6 std::vector< common::BathymetryPoint > get_port_bathymetry_points ( ) const [inline]

Helper method which returns the port-side bathymetry points as a std::vector.

Note

std::vector constructor copies the data out of the structure into a new allocated array

Since

0.5

4.12.2.7 int get_port_sidescan3d_point_count ( ) const [inline]

Number of port-side sidescan-3d points present.

4.12.2.8 const common::Sidescan3DPoint* get_port_sidescan3d_point_ptr ( ) const [inline]

Pointer to the start/first port-side sidescan-3d point within the structure.

4.12.2.9 std::vector< common::Sidescan3DPoint > get_port_sidescan3d_points ( ) const [inline]

Helper method which returns the port-side sidescan-3d points as a std::vector.

Note

Typically a compiler will automatically return value optimize and inline this code. Hence its recommended to use this std::vector over directly using the point_ptr() and point_count().

4.12.2.10 int get_port_sidescan_point_count ( ) const [inline]

Number of port-side sidescan points present.

4.12.2.11 const common::SidescanPoint* get_port_sidescan_point_ptr ( ) const [inline]

Pointer to the start/first port-side sidescan point within the structure.

4.12.2.12 std::vector< common::SidescanPoint > get_port_sidescan_points ( ) const [inline]

Helper method to return the port-side sidescan data as a std::vector.

Note

Typically a compiler will automatically return value optimize and inline this code. Hence its recommended to use this std::vector over directly using the point_ptr() and point_count().

4.12.2.13 std::string get_recorded_filename ( ) const [inline]

Helper method which returns the recorded filename as a std::string.

Since

0.6

4.12.2.14 std::string get_recorded_version ( ) const [inline]

Helper method which returns the recorded version as a std::string.

Since

0.6

4.12.2.15 int get_starboard_bathymetry_point_count ( ) const [inline]

Number of starboard-side bathymetry points present.

Since

0.5

4.12.2.16 const common::BathymetryPoint∗ get_starboard_bathymetry_point_ptr ( ) const [inline]

Pointer to the start/first starboard-side bathymetry point within the structure.

Since

0.5

4.12.2.17 std::vector<common::BathymetryPoint> get_starboard_bathymetry_points ( ) const [inline]

Helper method which returns the starboard-side bathymetry points as a std::vector.

Note

std::vector constructor copies the data out of the structure into a new allocated array

Since

0.5

4.12.2.18 int get_starboard_sidescan3d_point_count ( ) const [inline]

Number of starboard-side sidescan-3d points present.

4.12.2.19 const common::Sidescan3DPoint∗ get_starboard_sidescan3d_point_ptr ( ) const [inline]

Pointer to the start/first starboard-side sidescan-3d point within the structure.

4.12.2.20 std::vector<common::Sidescan3DPoint> get_starboard_sidescan3d_points ( ) const [inline]

Helper method which returns the starboard-side sidescan-3d points as a std::vector.

Note

Typically a compiler will automatically return value optimize and inline this code. Hence its recommended to use this std::vector over directly using the point_ptr() and point_count().

4.12.2.21 int get_starboard_sidescan_point_count ( ) const [inline]

Number of starboard-side sidescan points present.

4.12.2.22 const common::SidescanPoint∗ get_starboard_sidescan_point_ptr ( ) const [inline]

Pointer to the start/first starboard-side sidescan point within the structure.

4.12.2.23 std::vector<common::SidescanPoint> get_starboard_sidescan_points ( ) const [inline]

Helper method which returns the starboard-side sidescan points as a std::vector.

Note

Typically a compiler will automatically return value optimize and inline this code. Hence its recommended to use this std::vector over directly using the point_ptr() and point_count().

4.12.3 Member Data Documentation

4.12.3.1 uint64_t id

The data id (ping number).

4.12.3.2 dx::DxParameters parameters

The sonar settings in use at the time the data was acquired.

4.12.3.3 dx::DxSystemInfo system_info

The sonar system info.

4.12.3.4 common::Timestamp time

Time at which the trigger occured.

4.12.3.5 common::Timestamp time_range_zero

The zero range time. (ie the time at which the sonar range=0)

Note

This timestamp, along with the bulk sound velocity and range value, allows the exact time to be computed from the range values for any sidescan or sidescan3D data point.

4.13 DxHeader Struct Reference

The header transmitted before each data structure.

Public Member Functions

- bool is_preamble_valid ()

Public Attributes

- uint8_t preamble [16]
- uint32_t data_count

4.13.1 Detailed Description

The header transmitted before each data structure.

A two part structure including a unque preamble and the data count in bytes for the data buffer to follow. A preamble validation method is also included.

Since

0.1

4.13.2 Member Function Documentation

4.13.2.1 bool is_preamble_valid ( ) `[inline]`

Checks and return if the preamble is valid.

4.13.3 Member Data Documentation

4.13.3.1 uint32_t data_count

The size of the data structure to follow in bytes.

4.13.3.2 uint8_t preamble[16]

A unique 16 byte preamble.

4.14 DxParameters Struct Reference

Represents the parameters at the time of the ping.

Public Attributes

- float32_t range
- common::TriggerSettings trigger
- common::SoundVelocity sound_velocity
- common::Gain port_gain
- common::Gain starboard_gain
- common::SidescanSettings port_sidescan
- common::SidescanSettings starboard_sidescan
- common::Sidescan3DSettings port_sidescan3d
- common::Sidescan3DSettings starboard_sidescan3d
- common::TransmitSettings port_transmit
- common::TransmitSettings stardboard_transmit
- char8_t reserved [68]

4.14.1 Detailed Description

Represents the parameters at the time of the ping.

An eleven part structure including all sonar settings in use at the time the data was acquired. Substructures as defined above are used to group settings according to sonar function.

Since

0.1

4.14.2 Member Data Documentation

4.14.2.1 common::Gain port_gain

The sonar port-side receive gain setting.

4.14.2.2 common::SidescanSettings port_sidescan

The sonar port-side sidescan settings.

4.14.2.3 common::Sidescan3DSettings port_sidescan3d

The sonar port-side sidescan-3d settings.

4.14.2.4 common::TransmitSettings port_transmit

The sonar port-side transmit settings.

4.14.2.5 float32_t range

The sonar range setting.

4.14.2.6 char8_t reserved[68]

Temporary reserved bytes and also padding to 8byte boundary.

Since

0.3

4.14.2.7 common::SoundVelocity sound_velocity

The sonar sound velocity setting.

4.14.2.8 common::Gain starboard_gain

the sonar starboard-side receive gain settings.

4.14.2.9 common::SidescanSettings starboard_sidescan

The sonar starboard-side sidescan settings.

4.14.2.10 common::Sidescan3DSettings starboard_sidescan3d

The sonar starboard-side sidescan-3d settings.

4.14.2.11 common::TransmitSettings stardboard_transmit

The sonar starboard-side transmit settings.

4.14.2.12 common::TriggerSettings trigger

The sonar trigger settings.

4.15 DxSystemInfo Struct Reference

Contains information specific to this sonar system.

Public Attributes

- char8_t id [32]
- float32_t acoustic_frequency
- float32_t sample_rate
- float32_t maximum_ping_rate
- float32_t port_sidescan_range_resolution
- float32_t starboard_sidescan_range_resolution
- float32_t port_sidescan3d_range_resolution
- float32_t starboard_sidescan3d_range_resolution
- float32_t port_transducer_angle
- float32_t starboard_transducer_angle
- char8_t reserved [60]

4.15.1 Detailed Description

Contains information specific to this sonar system.

A ten part structure including information about the sonar.

Since

0.1

4.15.2 Member Data Documentation

4.15.2.1 float32_t acoustic_frequency

The acoustic frequency of this sonar in Hertz.

4.15.2.2 char8_t id[32]

A 32 character null terminated string identifying the sonar.

4.15.2.3 float32_t maximum_ping_rate

The maximum ping rate for the sonar in Hertz for the current settings.

4.15.2.4 float32_t port_sidescan3d_range_resolution

The range resolution in meters for the port sidescan-3d data.

4.15.2.5 float32_t port_sidescan_range_resolution

The range resolution in meters for the port sidescan data for this ping.

4.15.2.6 float32_t port_transducer_angle

The angle of the port transducer in degrees as mounted in the sonar housing relative to horizontal and with positive values indicating downward.

4.15.2.7 char8_t reserved[60]

Temporary reserved bytes and also padding to 8byte boundary.

4.15.2.8 float32_t sample_rate

The sample rate for the sidescan data in Hertz.

4.15.2.9 float32_t starboard_sidescan3d_range_resolution

The range resolution in meters for the starboard sidescan-3d data.

4.15.2.10 float32_t starboard_sidescan_range_resolution

The range resolution in meters for the starboard sidescan data.

4.15.2.11 float32_t starboard_transducer_angle

The angle of the starboard transducer in degrees as mounted in the sonar housing relative to horizontal and with positive values indicating downward.

Chapter 5

File Documentation

5.1 pingdsp-3dss.hpp File Reference

3DSS-DX data interface header file for C++ compilers.

Classes

- struct Timestamp

  Represents the elapsed time since January 1, 1970 12:00 AM.
- struct SoundVelocity

  Represents the speed of sound values in m/s used by the sonar.
- struct Gain

  Represents the gain controlling coefficients used by the sonar.
- struct AsciiSentence

  Represents an ascii sentence received by the sonar.
- struct SidescanSettings

  Represents the sidescan settings used by the sonar.
- struct Sidescan3DSettings

  Represents the sidescan-3d settings used by the sonar.
- struct TransmitSettings

  Represents the transmit settings used by the sonar.
- struct TriggerSettings

  Represents the trigger settings used by the sonar.
- struct SidescanPoint

  Represents a sidescan point.
- struct Sidescan3DPoint

  Represents a sidescan 3d point.
- struct BathymetryPoint

  Represents a bathymetry point.
- struct DxParameters

  Represents the parameters at the time of the ping.
- struct DxSystemInfo

  Contains information specific to this sonar system.

- struct DxHeader

  The header transmitted before each data structure.
- struct DxData

  Represents all data and sonar settings for an entire sonar ping.

Macros

- #define PINGDSP_3DSS_VERSION "0.6"

Typedefs

- typedef float float32_t
- typedef int8_t char8_t

Variables

- const int **kTcpPort** = 23848
- const uint8_t **kPreamble** [ ]
- **sizeof** (common::TriggerSettings)+sizeof(common

5.1.1 Detailed Description

3DSS-DX data interface header file for C++ compilers.

5.1.2 Macro Definition Documentation

5.1.2.1 #define PINGDSP_3DSS_VERSION "0.6"

The version of this source file.

5.1.3 Typedef Documentation

5.1.3.1 typedef int8_t char8_t

Typedef char8_t as a simple UTF-8 character.

5.1.3.2 typedef float float32_t

Typedef and compile time statements which ensure that float is of compatable with IEEE-754 32bit binary format. std-::numeric_traits<float>::is_iec559 std::numeric_traits<float>::digits == 24?

Index