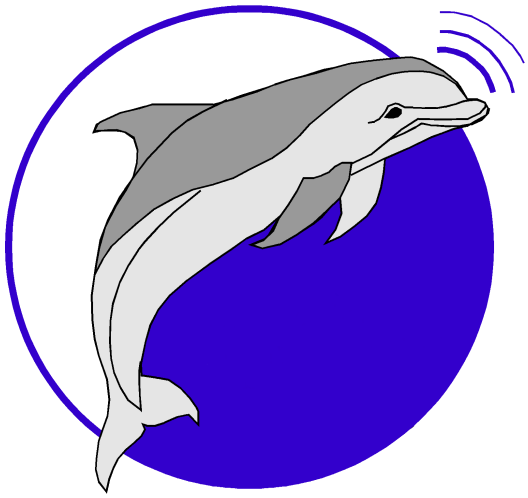


Sea Scan PC Real Time Output Protocol Manual V1.1.1

Marine Sonic Technology, Ltd.
Gloucester, VA

This product was designed and developed by a team of engineers at Marine Sonic Technology, Ltd.

Marine Sonic Technology, Ltd.
5508 George Washington Memorial Highway
P.O. Box 730, White Marsh, VA 23183-0730
(804) 693-9602
(800) 447-4804

Technical Support

For technical support call (800) 447-4804 or visit our web site at http://www.marinesonic.com.

Copyright

Limitations on Warranty & Liability

Marine Sonic Technology, Ltd. warrants that the disk(s) on which this software is recorded is free from defects in materials and workmanship under normal use for 90 (ninety) days after the date of original purchase.

Please refer to this product's Warranty for further information concerning the limitations on warranty and liability of this product and its associated software.

Trademarks

Sea Scan® is a registered trademark of Marine Sonic Technology, Ltd.

SHARPS™ is a trademark of Marine Sonic Technology, Ltd.

Table of Contents

Section 1 Introduction 2

Section 2 Receiving Data From Sea Scan PC 4

Section 3 The Sonar Data Stream In Depth 6

1 Variable Types 6
2 General Packet Structure 6
3 Data Packet Types 7
Sync Data Packet 8
Sonar Data Packet 8
Navigation Data Packet 10
Orientation Data Packet 11
Fathometer Data Packet 12
Magnetometer Data Packet 12
Mark Data Packet 12
NMEA Data Packet 13
Types of Data Sent By Sea Scan PC 13
4 Synchronizing the Data Stream 13

Section 4 Appendix 16

1 SDS.H 16
2 Configuring Sea Scan PC Real Time Output 22
Overview 22
Settings 23
Status 24
3 Testing With Fake Data 24
4 Contact Information 25

Section 1

- **Introduction**

1 Introduction

This manual describes the Sonar Data Stream and its role in outputting data in real time from Sea Scan PC version 1.8.0 and newer. Sea Scan PC versions 1.8.0 and newer have the capability of streaming most of the data it collects through a network connection to another application. In order to accomplish this Sea Scan PC contains a UDP/IP (TCP/IP in version 1.8.1 or older) client object that digests recently collected data, places it into the configured protocol, and sends it as fast as it can through the network to a server application that is responsible for receiving the data and processing it.

Section 2

- **Receiving Data From Sea Scan PC**

2 Receiving Data From Sea Scan PC

To receive data from Sea Scan PC one needs to construct an application that implements a UDP/IP Server that will accept at least 1 incoming connection. If you are unfamiliar with the concept of the client/server model for UDP/IP, please contact Marine Sonic Technology, Ltd. and we will do our best to assist you in understanding it.

The order that the applications are started in does not matter as Sea Scan PC will attempt to connect on every packet transmission if it is not currently connected, however Marine Sonic Technology, Ltd. recommends that the Server application be started first. Please refer to the **Configuring Sea Scan PC Real Time Output** 22 section in the Appendix to learn how to use Sea Scan PC to configure the Real Time output. You may also wish to read the Sea Scan PC Operator's Manual v1.8.0 for more information about the operation of Sea Scan PC. This and many more manuals and documents can be downloaded from out web site listed in the **Contact Information** 25 section of the Appendix.

Once the two applications have been started and the Sea Scan PC has been configured properly to transmit Sonar Data Stream packets, Sea Scan PC will connect to the Server application and begin transmitting SYNC packets every 1 second. This will establish the Sonar Data Stream. As soon as other data is received and processed, it will be transmitted over the data stream.

For receiving actual data on the bench you may want to put Sea Scan PC into training mode. This will cause Sea Scan PC to generate fake side scan sonar data, navigation data, and fathometer data. To put Sea Scan PC into training mode, please refer to the **Testing With Fake Data** 24 section in the Appendix.

Section 3

- **The Sonar Data Stream In Depth**

3 The Sonar Data Stream In Depth

The Sonar Data Stream is a data format that allows the inclusion of sampled sonar data of various types, navigational and positioning data, fathometer (depth & altitude) data, magnetometer data, as well as markers or notable events all in the same data stream. The most important concept to understand in the Sonar Data Stream is the time stamping system. Every piece of data is stamped with a millisecond accurate (theoretically) timestamp that is reference to Greenwich Mean Time. This allows the data to be processed by other applications in accordance to the time at which the data was collected.

The Sonar Data Stream was engineered to be serialized and streamed into/out of a file, over network connections of various types, and over asynchronous and synchronous serial links. It includes a synchronization scheme that allows for a stream that is interrupted to resume receiving again at a known good location in the data.

This section provides a fairly good description of the Sonar Data Stream and how it should be used. You may find looking at the C/C++ header file more informative or easier to digest. Please refer to the **SDS.H** 16 part of the Appendix for the header file. If you would like more information, need help with some of the concepts of the Sonar Data Stream or would like to propose an addition or modification please **contact us** 25.

3.1 Variable Types

All of the data contained in the sonar data stream is stored as "Little Endian" byte order. The variable sizes that are used in the **header file** 16 as well as in the following section(s) are listed below.

Type	Size in Bytes	Signed/Unsigned
BYTE	1	unsigned
WORD	2	unsigned
DWORD	4	unsigned
int	4	signed
short int	2	signed
float	4	NA
double	8	NA

3.2 General Packet Structure

Each Sonar Data Stream packet contains a header and a data payload (optional). The packet header is 16 bytes in size and is formatted like this:

Element	Variable Type	Description
dwSize	DWORD	The size of the entire data packet, not including this header.

dwTimeStamp	DWORD	The number of milliseconds since Midnight GMT.
dwTag	DWORD	This describes the contents of the data packet and how the header will be used.
byMisc[3]	BYTE	Used for miscellaneous storage.
byChecksum	BYTE	This is a simple 8 bit checksum calculated by adding all of the bytes in the packet header (for a total of 15 bytes) into an 8 bit number. This checksum excludes the byChecksum variable (the last byte).

In addition to the header a packet may have a data payload. The tag is used to describe the packet type. In the tag's description there will be information on how and what the data is. A packet's data always follow the header.

Each packet header must have the size, tag, and time stamp filled out. The miscellaneous storage should be zeroed out for every packet type except for the Sync packet. The Sync packet should have the miscellaneous storage filled with a value of 0xAA.

There are currently 7 possible values for the tag. These values are listed in the next section, Data Packet Types 7. Additional data packets may be defined (if inserting your own data into the sonar data stream). Packets that contain an invalid tag should be ignored. This can be accomplished by simply skipping over the data payload using the dwSize variable in the packet's header.

Built into the Sonar Data stream is a couple of mechanisms to insure the stream has valid data. The first method, which is mentioned above, is to read only the tags that your software understands. Another mechanism is the 8 bit checksum if the packet header that is at the end of the header. This will give a program a good indication of the validity of a packet received. If the checksum should come out to be bad, **re-synchronizing the data stream** 13 will be necessary.

Skipping back and forth in the data stream can be accomplished in chunks. By using the dwSize variable which is at the beginning of every sonar data stream packet you can rewind and skip ahead to different sections of the data stream. Please note that skipping ahead is only an option if the data stream is being read from a non-actively written to file on disk.

3.3 Data Packet Types

The are 7 basic packets types defined right now in the sonar data stream. Each of these packets has an associated tag that is 32 bits in size. List below are the 7 types and their tags.

Type	Tag	Description
Sonar	0x534E5200	This is a sonar data packet. It can store the data types of many different sonar systems as well as many different data formats.
Sync	0x53594E43	This tag represents a sync data packet. This packet is sent at regular intervals and is used to synchronize the data stream.
Navigation	0x4E415600	This data packet allows the storage of navigation data.
Orientation	0x4F524E54	This data packet stores orientation of an object from the origin of the navigation data.

Fathometer	0x46415400	This data packet stores fathometer information such as depth & altitude of an object.
Magnetometer	0x4D414700	This data packet stores magnetometer readings.
Mark	0x4D41524B	This data packet allows marks to be placed in the sonar data. The marks are time as well as geographically referenced.
NMEA	0x4E4D4541	The NMEA data packet stores strings containing commands and data whose format is based on the NMEA 0183 standard.

3.3.1 Sync Data Packet

The Sync data packet allows the data stream to be synchronized. It can be used to find a good starting point in corrupted data or it can be used to start a frame. Skipping to the next Sync packet will the advance the frame X amount of time, where X is the Sync packet interval. The SYNC packets should be sent at regular intervals which are specified in the packet's data payload.

Every stream should start with SYNC packet. This is to prevent data from unnecessarily being skipped over by the synchronization process. To learn more about how to synchronize the data stream please refer to **Synchronizing the Data Stream** 13 later in this section.

Please note that the Sync data packet is the only packet that uses the miscellaneous bytes in the packet header. The bytes should be assigned the 0xAA value. This value insures that the data stream can be re-synchronized should it become corrupted. For more information see the Synchronizing the Data Stream section. This will yield 3 consecutive 0xAA values.

A Sync data payload is defined in the table below:

Element	Variable Type	Description
dwReference	DWORD	This is the number of seconds elapsed since midnight (00:00:00), January 1, 1970
wInterval	WORD	This is the sync packet repetition interval in milliseconds.

3.3.2 Sonar Data Packet

The Sonar data packet allows the storage of multiple types of sonar data. The sonar data starts with an sonar data channel structure followed by the actual sonar data itself. Each individual channel has its own Sonar data packet.

The sonar data channel structure is as follows:

Element	Variable Type	Description
wSonarType	WORD	This is the type of sonar/ transducer (Port, Starboard, Forward Looking, Scanning, etc.)

fFreqHz	float	This is the acoustic frequency in Hertz of this channel.
fSpeedofSound	float	This is the speed of sound in water. It is in meters per second.
fRangeDelayMs	float	This is the delay from the start of the transmit pulse to the first sample in milliseconds.
fRange	float	This is the range in milliseconds that this set of data represents.
bySampleSizeBytes	BYTE	This is the size in byes of each sonar sample. This will depend heavily on the type of data being stored.
wDataType	WORD	This is the type of data stored in this data set. Please see the possible data types that are listed below.
dwSamples	DWORD	This is the number of samples in this channel.

The Sonar Type can be one of several types. The different types of data is listed below:

Sonar Type	wSonarType Value	Description
Port Side Scan	0x0001	Port facing side scan sonar.
Starboard Side Scan	0x0002	Starboard facing side scan sonar.
Forward	0x0003	Forward looking sonar.
Multibeam	0x0004	Multi-Beam sonar.
Scanning	0x0005	Scanning sonar.
Sub-Bottom	0x0006	Sub-Bottom Profiling sonar.
User Defined	0x0100	User defined sonar. If you don't know what it is, don't parse it.

For sonar types such as scanning sonar, the **orientation data packet** 11 describes the direction in which the scanning sonar is pointing and it's orientation with reference to the ship or other object.

The data type also has many different flavors. The data type affects the sample size. You can use the bySampleSizeBytes along with the dwSamples to calculate the exact number of bytes that follows the sonar data channel structure. The different data types are listed in the table below:

Data Type	wDataType Value	bySampleSizeBytes Value	Comments
Unsigned 8 Bit	0x0001	1	

Signed 8 Bit	0x0002	1	
Unsigned 12 Bit	0x0003	2	The upper 4 bits should be 0.
Signed 12 Bit	0x0004	2	The upper 4 bits should be 0 (except for the sign)
Unsigned 16 Bit	0x0005	2	
Signed 16 Bit	0x0006	2	
Floating Point	0x0007	4	
Complex Integer	0x0008	4	A complex integer is stored as two consecutive signed 16 bit integers, the real element first, followed by the imaginary element.
Complex Float	0x0009	8	A complex float is stored as two consecutive floating point numbers, the real element first, followed by the imaginary element.

3.3.3 Navigation Data Packet

The navigation data packet allows the storage of navigation data. This navigation data is strictly geographically defined using standard latitude/longitude coordinates. The navigation data payload structure is listed below.

Element	Variable Type	Description
wSource	WORD	Origin of this navigation data.
dLatitude	double	The latitude encoded as decimal degrees.
dLongitude	double	The longitude encoded as decimal degrees.
fCOG	float	The course over ground in degrees. (Stored as TRUE not magnetic)
fHeading	float	The heading in degrees (Stored as TRUE not magnetic)
fSOG	float	The speed over ground in meters per second.

It provides variables to describe the navigation from several different sources. The sources are stored in the wSource variable. A table of these sources and their identifier value can be found below.

Navigation Source	wSource Value	Description
Ship	0x0001	The navigation data came from the survey ship.
Sonar	0x0002	The navigation data came from the sonar itself (usually a towfish or AUV).
Sensor 1	0x0003	The navigation data came from sensor number 1.
Sensor 2	0x0004	The navigation data came from sensor number 2.
Sensor 3	0x0005	The navigation data came from sensor number 3.
Sensor 4	0x0006	The navigation data came from sensor number 4.

The Sensor sources are not well defined right now. The can be used to describe up to 4 different sources that are different from the ship or sonar.

3.3.4 Orientation Data Packet

The orientation data packet describes the orientation of the sonar in relation to the navigation data. The source of the navigation data is described in the data payload which is described below.

Element	Variable Type	Description
wSource	WORD	This is the origin of the navigation data which this packet adds to.
fX	float	X Offset in meters form the origin.
fY	float	Y Offset in meters form the origin.
fZ	float	Z Offset in meters from the origin.
fRoll	float	Roll in degrees.
fPitch	float	Pitch in degrees
fYaw	float	Yaw in degrees.
fHeave	float	Heave in centimeters.

This orientation packet can be used for many purposes. One of these purposes can describe the cable layback of a side scan sonar system. Another use can be describing the direction that a scanning sonar head is pointing in.

Please see the sources listed in **Navigation Data Packet** 10 section for possible values for wSource.

3.3.5 Fathometer Data Packet

The fathometer data packet basically stores depth and altitude information for a particular origin. This data can be used along with the navigation data to give the user an accurate geographical location. The fathometer data payload is described below.

Element	Variable Type	Description
wSource	WORD	This is the origin of the fathometer data.
fDepth	float	This is the depth in meters.
fAltitude	float	This is the altitude in meters.

Please see the sources listed in **Navigation Data Packet** 10 section for possible values for wSource.

3.3.6 Magnetometer Data Packet

The magnetometer data packet stores readings from a magnetometer sensor. It contains a source variable to indicate where the magnetometer is located. Please see the sources listed in **Navigation Data Packet** 10 section for possible values for wSource. Below is the structure of the magnetometer data payload.

Element	Variable Type	Description
wSource	WORD	This variable stores the source location of the magnetometer data.
fReading1	float	This is the reading number 1 from the magnetometer in Gammas.
fReading2	float	This is the reading number 2 from the magnetometer.

3.3.7 Mark Data Packet

The marker data packet is used to store information particular to a geographical point at a particular time. This can be used to place annotations, markers, and way points in the sonar data stream. The marker data packet contains a description string as well as a location in Latitude/Longitude plus offset information. The marker data packet consists of the marker data payload structure and is followed by a NULL terminated string. The length of the string is contained within the marker data payload. A description of the payload can be found below:

Element	Variable Type	Description
dLat	double	Origin Latitude of the mark.
dLon	double	Origin Longitude of the mark.

fXOffset	float	Offset in meters in the X direction from the origin.
fYOffset	float	Offset in meters in the Y direction from the origin.
fZOffset	float	Offset in meters in the Z direction from the origin.
wLength	WORD	Length of the string in the characters including the NULL terminator.

3.3.8 NMEA Data Packet

The NMEA data packet stores strings containing commands and data whose format is based on the NMEA 0183 standard. The NMEA data packet consists of the NMEA data payload structure and is followed by a NULL terminated string. The NMEA data payload contains the length of the string and contains a source variable to indicate the device from which the string originated. Please see the sources listed in **Navigation Data Packet** 10 section for possible values for wSource. Below is the structure of the NMEA data payload.

Element	Variable Type	Description
wSource	WORD	This variable stores the source location of the magnetometer data.
wLength	WORD	Length of the string in the characters including the NULL terminator.

3.3.9 Types of Data Sent By Sea Scan PC

Sea Scan PC does not send all of the types of data that the sonar data stream supports. Below is a list of data packets that Sea Scan PC does send:

1) Sync
2) Sonar - Port Side Scan Sonar
3) Sonar - Starboard Side Scan Sonar
4) Navigation Data - Ship Source
5) Navigation Data - Fish Source
6) Fathometer Data - Sonar Source

3.4 Synchronizing the Data Stream

There are two specific times at which you will want to synchronize the sonar data stream. The first is when an application first starts reading data from its network connection. The second is when a sonar data stream becomes unsynchronized (a header checksum fails).

In order to synchronize the data stream, start by parsing the stream for the serial synchronization bit stream (0x53594E43AAAAAA). The bit stream consists of the SYNC tag and the serial synchronization DWORD. This unique bit stream identifies a SYNC packet. After the synchronization bit stream is confirmed valid the 4 bytes before make the dwSize variable. You should now rewind to

the start of the packet header and confirm that the header is a valid one. You can then read the reference time stamp to get the exact date and time of this packet as well as when to expect the next SYNC packet.

Section 4

- **Appendix**

4 Appendix

Enter topic text here.

4.1 SDS.H

```
//
*************************************************************************
*
// ** Sonar Data Stream
// **
// ** Preliminary Specification
// ** Created by in alpabetical order:
// **       Chesapeake Technology, Inc.
// **       Marine Sonic Technology, Ltd.
// **       Other People Too Hopefully
// **
// ** This file describes an open standard for a Sonar Data Stream.
// ** The idea is to break up all of the data needed to be stored or
// ** transmitted into common groups.  Groups that do not need to be used
can
// ** be ignored if they exist in the data stream.
// **
// ** All data being recorded is time based so a large emphasis is placed
// ** on a synchronization packet being introduced into the data stream
// ** at regular intervals in order to keep all of the different data
// ** types synchronized.
// **
//
*************************************************************************
*
// ** - Version History -
// **
// ** APRIL 14, 2006
// ** V0.1
// ** - Created (Pre Release Version)
// ** APRIL 19, 2006
// ** V0.1.1
// ** - Added complex number types
// ** - Moved the Time Reference to the SYNC packet
// ** - Changed the dwMisc meaning from interval to additional SYNC Tag
// ** identifier (configured as a bit stream synchronization pattern).
// ** - Removed the version number from the end of the TAG
// ** V0.1.2
// ** - Modified the SDS sync packet header (moved the dwTag order)
// ** - Remove the *pData pointers in every packet that use them
// **   (sonar, strings, etc...) and added the description of the packet
and
// **   where to place the data.
// ** - Added fSOG to the NAV packet (was missing)
// ** - Added fSpeedOfSound to the SonarDataChannel structure.
// ** - Changed all DATASOURCE_FISH to DATASOURCE_SONAR because some
systems
// **   are fish less.
// **
// ** V0.1.3
// ** - Added a NMEA packet for storing NMEA strings received by the sonar
for
// **   later processing by the topside and debugging.  Also for packing
```

```
// **   Host Remote NMEA reply sentences into the Sonar Data Stream for use
// **   on towed systems that only have 1 communications channel that both
// **   command & data must ride on.
// **
// ** V0.1.4
// ** - Add the implied invalid value for data packets (99999.99)
//
*************************************************************************
*

//#include "standard.h"   // standard definitions & types

#ifndef __SDS_H__
#define __SDS_H__

// ****************************************


// Uncomment if this is supported by your compiler
// otherwise you must ensure that the structures are packed to 1 byte
boundaries
//#pragma pack(push,1)
#pragma pack(1)

// NOTE:    All data and variables are to be stored as "Little Endian" byte
order
//               Variable sizes are as follows:
//               BYTE        - 1 Byte
//               WORD        - 2 Bytes
//               DWORD       - 4 Bytes
//               int         - 4 Bytes
//               short int   - 2 Bytes
//               float       - 4 Bytes
//               double      - 8 Bytes

// - Packet Header -
// Each packet contains a header and data(optional)
// The header describes what the packet contains and its size in bytes

// NOTE:  Each packet must have the Size, Tag, and Timestamp filled out.
Misc. should
// be set to 0 unless otherwise noted.

struct SDS_HEADER
{
      DWORD dwSize;           // Size of the Data Packet (not including the
header).
      DWORD dwTimeStamp;      // Time Stamp is the number of milliseconds
since Midnight GMT.
      DWORD dwTag;            // Tag describes the contents of the packet
or how the header will be used.
      BYTE  byMisc[3];        // Misc is for miscellaneous use and for data
that is less than 3 bytes.
      BYTE  byCheckSum;       // Simple 8 bit checksum.
};

// NOTE: Additional data packets can be defined.  Manufacturer specific
packets can be defined
// with a TAG beginning with 0x00###### which will give 24 bits worth of
choices.  Tags that are
// unknown to the reader should be ignored.
```

```
//

// - SYNC Packet -
// The SYNC packet allows the data stream to be synchronized
// it can be used to find a good starting point in corrupted data
// or it can be used to start a frame ... skipping to the next
// SYNC packet will advance the frame X amount of time
// The SYNC packets should be sent/formed at regular intervals
// specified in the packet.
//
// The SYNC packet is also used to Synchronize real time with the time
stamp time.
//
// NOTE:    Every file should have a SYNC packet at the start of the file.
//                If the SYNC packet is not at the beginning then a file
reader should advance to the
//                nearest SYNC packet and start reading from there.
//
// TAG:           0x53594E43 or ASCII "SYNC"
// MISC:    Contains a serial synchronization bit stream: 0xAAAAAA
//                Bit Stream: 101010101010101010101010
//                Also serves the purpose of increasing the uniqueness of
the SYNC packet TAG
// DATA:    Contains a real time reference in order to synchronize the time
stamp in the header wiht
//                a real date and time.  The time and date contained is
referenced to GMT time.

#define SDS_TAG_SYNC    0x53594E43

struct SDS_DATA_SYNC
{
      DWORD dwReference;      // Numbers of seconds elapsed since midnight
(00:00:00), January 1, 1970
                              // Most Windows and UNIX (POSIX) based
systems have the time() function
                              // available.  The reference is simply the
output from time().
                              // These systems normally carry functions to
convert this time reference
                              // to many different formats.
      WORD  wInterval;        // Sync repetition interval in milliseconds
at which the SYNC packet is sent.
                              // The default interval is 1 second.
};

//
// - Sonar Data Packet -
// The sonar data packet allows the storage of multiple types of sonar
data.
// The sonar data starts with a SDS_SONAR_DATA_CHANNEL structure followed
by
// the actual sonar data itself.  Each individual channel has its own SDS
// Sonar Data packet.
//
// TAG: 0x534E5200 or ASCII "SNR"
// MISC: 0

// - Sonar Types -
// Pick only 1 of these to go in the wSonarType variable field
//
```

```
// Custom and manufacturer specific sonar types begin at
SONARTYPE_USER_DEFINED

#define SDS_TAG_SNR           0x534E5200

#define SONARTYPE_PORT_SIDESCAN                 0x0001
#define SONARTYPE_STARBOARD_SIDESCAN            0x0002
#define SONARTYPE_FORWARD                       0x0003
#define SONARTYPE_MULTIBEAM                     0x0004
#define SONARTYPE_SCANNING                      0x0005
#define SONARTYPE_SUBBOTTOM                     0x0006
#define SONARTYPE_USER_DEFINED                  0x0100

// - Data Types -
// Pick only 1 of these to go in the wDataType varaible field
//

// 8 Bit Data is stored 1 BYTE Wide
#define DATATYPE_UNSIGNED8BIT                   0x0001
#define DATATYPE_SIGNED8BIT                     0x0002

// 12 Bit Data is stored 2 BYTEs Wide (the upper 4 bits should be 0)
#define DATATYPE_UNSIGNED12BIT                  0x0003
#define DATATYPE_SIGNED12BIT                    0x0004

// 16 Bit Data is stored 2 BYTEs Wide
#define DATATYPE_UNSIGNED16BIT                  0x0005
#define DATATYPE_SIGNED16BIT                    0x0006

// Floating Point is stored 4 BYTEs Wide and is a floating point number
available in 32 bit C++ compiler
#define DATATYPE_FLOAT                          0x0007

// Complex Integer is stored as a two consecutive signed 16 bit integers,
the real elemet first, followed by
// the imaginary element.
// For example: real0 complex0 real1 complex1 real2 complex2 ...
#define DATATYPE_COMPLEX_INT                    0x0008

// Complex Floating Point is stored as two consecutive floating point
numbers, the real element first
// , followed by the imaginary element.  See Complex Integer (above) for an
example.
#define DATATYPE_COMPLEX_FLOAT                  0x0009

struct SDS_DATA_SONAR_CHANNEL
{
      WORD  wSonarType;             // Type of sonar/transducer (Port,
Starboard, Forward Looking, Multibeam, Scanning, etc.)
      float fFreqHz;                // Acoustic frequency in Hz of this
channel
      float fSpeedOfSound;          // Speed of sound in water in meters
per second.
      float fRangeDelayMs;          // Delay from the start of the transmit
pulse to first sample in milliseconds
      float fRangeMs;               // Range in milliseconds that this set
of data represents
      BYTE  bySampleSizeBytes;      // Size in bytes of each sonar sample
      WORD  wDataType;              // Type of data stored in this data set
(UNSIGNED, SIGNED, 8 Bit, 12 Bit, 16 Bit, Complex, etc.)
      DWORD dwSamples;              // Number of samples in this channel
```

```
};

// - Sources for the Data Packets that contain the variable wSource -
//

#define DATASOURCE_SHIP                    0x0001
#define DATASOURCE_SONAR                   0x0002
#define DATASOURCE_SENSOR1                 0x0003
#define DATASOURCE_SENSOR2                 0x0004
#define DATASOURCE_SENSOR3                 0x0005
#define DATASOURCE_SENSOR4                 0x0006

// Invalid Values for the Data Packets
#define INVALID_FLOAT_VALUE                99999.99F

// - Navigation Data Packet -
// The navigation data packet allows the storage of navigation data
//
// TAG: 0x4E415600 or ASCII "NAV"
// MISC:

#define SDS_TAG_NAV           0x4E415600

struct SDS_DATA_NAV
{
      WORD  wSource;          // Origin of this navigation data (Ship,
Towfish, etc.)
      double      dLatitude;  // Latitude ecoded as decimal degrees
      double      dLongitude; // Longitude encoded as decimal degrees
      float fCOG;             // Course Over Ground in degrees (true NOT
magnetic)
      float fHeading;         // Heading in degrees (true NOT magnetic)
      float fSOG;             // Speed Over Ground (in meters per second)
};

// - Orientation Data Packet -
// The orientation from the origin of the navigation data
//
// TAG: 0x4F524E54 or ASCII "ORNT"
// MISC: 0

#define SDS_TAG_ORNT    0x4F524E54

struct SDS_DATA_ORIENTATION
{
      WORD  wSource;          // Origin of the navigation data (Ship,
Towfish, etc.)
      float fX;               // X offset in meters from the origin
      float fY;               // Y offset in meters from the origin
      float fZ;               // Z offset in meters from the origin
      float fRoll;            // Roll in degrees
      float fPitch;           // Pitch in degrees
      float fYaw;             // Yaw in degrees
      float fHeave;           // Heave in cm
};

// - Fathometer Data Packet -
// The depth and altitude information of the origin of the data
//
// TAG:     0x46415400 or ASCII "FAT"
// MISC: 0
```

```
#define SDS_TAG_FAT           0x46415400

struct SDS_DATA_FATHOMETER
{
      WORD  wSource;          // Origin of the fathometer data (Ship,
Towfish, etc.)
      float fDepth;           // Depth in meters
      float fAltitude;        // Altitude in meters
};

// - Magnetometer Data Packet -
// The reading from a magnetometer sensor
//
// TAG:     0x4D414700 or ASCII "MAG"
// MISC: 0

#define SDS_TAG_MAG           0x4D414700

struct SDS_DATA_MAGNETOMETER
{
      WORD  wSource;               // Origin of the magnetometer data
(Ship, Towfish, etc.)
      float fReading;              // Magnetometer reading in Gammas
      float fReadingSigStrength;   // Magnetometer signal strength units??

};

// - Marker/Annotation Data Packet -
// The marker annotation data packet contains a description string as well
as location in Lat/Lon
// and offset information.
// The marker annotation data packet begins with a SDS_DATA_MARK structure
and is followed by an array of
// ASCII characters that form the description string.  The string can be a
maximum of 1024 characters
// including the NULL terminator.
//
// TAG: 0x4D41524B or ASCII "MARK"
// MISC: 0

#define SDS_TAG_MARK    0x4D41524B

struct SDS_DATA_MARK
{
      double      dLat;       // Origin Latitude of the Mark
      double      dLon;       // Origin Longitude of the Mark
      float fXOffset;         // Offset in meters in the X direction from
the origin
      float fYOffset;         // Offset in meters in the Y direction from
the origin
      float fZOffset;         // Offset in meters in the Z direction from
the origin
      WORD  wLength;          // Length of the string in characters
including the NULL terminator (maximum 1024)
};

// - NMEA Sentece Data Packet -
// The NMEA sentence data packet contains NMEA formatted strings that are
received and
// sent by the sonar or device that is generating the Sonar Data Stream.
```

```
This can be
// used for later processing by the topside and debugging.  Also for
packing Host Remote
// NMEA reply sentences into the Sonar Data Stream for use on towed systems
that have
// only 1 communications channel that both command & data must ride on.
The NMEA sentence
// data packet begins with the SDS_DATA_NMEA structure and is follwed by an
array of ASCII
// characters that form the NMEA string.  The string can be a maximum of
1024 characters
// including the NULL terminator.
//
// TAG: 0x4E4D4541 or ASCII "NMEA"
// MISC: 0

#define SDS_TAG_NMEA    0x4E4D4541

#define DATASOURCE_SHIP                    0x0001
#define DATASOURCE_SONAR                   0x0002
#define DATASOURCE_SENSOR1                 0x0003
#define DATASOURCE_SENSOR2                 0x0004
#define DATASOURCE_SENSOR3                 0x0005
#define DATASOURCE_SENSOR4                 0x0006

struct SDS_DATA_NMEA
{
      WORD wSource;                 // Source of the NMEA sentence
      WORD wLength;                 // Length of the string in characters
including the NULL terminator (maximum 1024)
};

// Uncomment if this is supported by your compiler
// otherwise you must ensure that the structures are packed to 1 byte
boundaries
//#pragma pack(pop)
#pragma pack()

#endif      // __SDS_H__
```

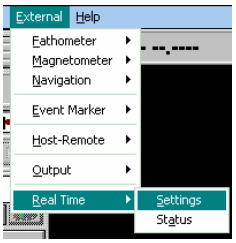
4.2 Configuring Sea Scan PC Real Time Output

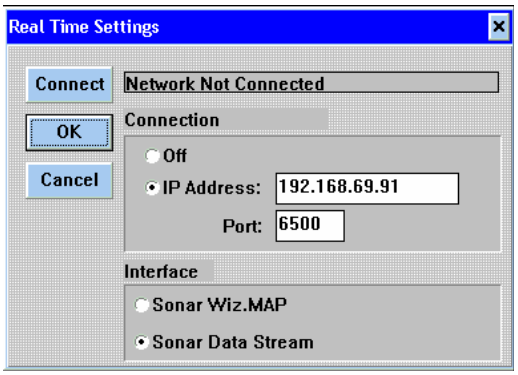
4.2.1 Overview

Sea Scan PC version 1.8.0 adds the ability to output the data that is collected in real time to another program using the host computers network connection. This ability is primarily used for connecting to Real Time Mosaicing applications such as Sonar Wiz.MAP from Chesapeake Technology. In order to use this feature you will be required to have some knowledge about your computers' network connections.

4.2.2 Settings

The real time interface sub menu can be found as one of the options in the external menu in Sea Scan PC. To configure the real time interface select the Settings menu option.



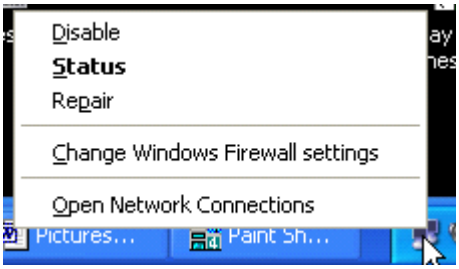


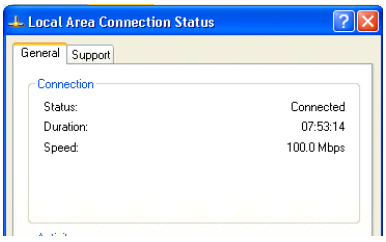
After clicking the Settings menu option, the Real Time Settings dialog window will appear. This window contains all of the configuration settings for the real time interface.

Click the OK or Cancel button when finished to close the Real Time Settings window.

The first step in setting up a real time interface is to configure the network Connection. There basically are two options when configuring the network Connection. These are Off and On. The Off option is labeled. The On option is not labeled however it is implied by the activation of the IP Address and Port entry boxes. When the Off option is selected the IP Address and Port entry boxes will be inaccessible and is indicated by them being grayed out. When the On option is selected you will be able to click in either of the entry boxes and type.

The IP Address is the address of the computer that will be receiving the collected data. This typically is 4 numbers between 1 and 255 separated by periods or dots. You can find the IP address of the receiving computer by right clicking on the network icon next to the clock on the computer that will be receiving the data as shown in the picture to the right. This is applicable to the Windows XP only.





Then choose the Status option. This will show the Local Area Network Connection Status window. Next click on the Support tab.

This will display the Connection Status information. This will show the IP Address that the computer is currently configured to. This is the number you will type into the Sea Scan PC real time IP Address entry box. Click the Close button in the Local Area Network Connection Status window when finished.

The Port entry box needs to be filled out with the number of the port that the receiving application will be listening on. For Sonar Wiz.MAP the port number is typically 6500. For other applications please refer to their product documentation for the Port number.

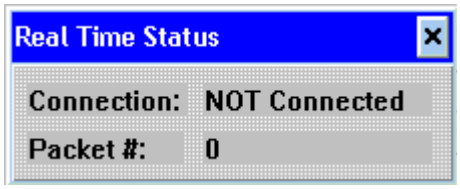
The next step in setting up a real time connection is choosing the correct interface type. For Sonar Wiz.MAP please choose the Sonar Wiz.MAP option. For all other applications please choose the Sonar Data Stream option.

Finally start the application on the receiving computer and click the Connect button in the Real Time Settings window. The status of the connection will be displayed to the right of the Connect button. If the receiving application is running and the Connection settings were filled out correctly, the status should now read Connected.

You may now click the OK button to keep the changes you have made or click the Cancel button to throw away the changes you have made to the settings.

4.2.3 Status

The real time interface in addition to having a Settings window also has a Status window. This window displays the current status of the real time interface. It will display the connection status as well as the number of the current data package that is being sent over the real time connection. Notice in the example to the right that the status window indicates that the connection is NOT Connected. Other possible phrases include Connected and Reconnecting.



4.3 Testing With Fake Data

In order to test your program you will need to operate the Sea Scan PC software in training mode. Sea Scan PC will then generate fake side scan sonar data, navigation data, and fathometer data. By default, Sea Scan PC always starts in active mode, the normal operating mode. Select the Mode menu item in the Options menu to change to the training mode. A pop-up menu will appear. The current mode, either active or training mode, is marked with a check mark. Select the menu item Training from the Mode pop-up menu. The Sea Scan PC will be switched to training mode. When in training mode, the word ---- *TRAINING* ---- is displayed on either side of the application name in the caption bar.

Any data in the image buffer is cleared, information from the external devices is cleared and the Sea Scan plotter is reset. Thus, any data that exists will be lost. If you do not want to lose any existing data, or any real information from the external devices before switching to training mode, save the sonar image file first.

The display parameters, such as the color scale and data scrolling direction, are not changed. However, the Sea Scan plotter parameters are modified. The bounds for both the current and survey plotters are changed so the simulated vessel's track will be properly displayed.

4.4 Contact Information

For more information, questions, or helpful additions/corrections to the sonar data stream format, please contact Marine Sonic Technology, Ltd at:

Christopher Favreau
P.O. Box 730
5508 George Washington Memorial Highway
White Marsh, VA, 23183-0730

Phone: (804)693-9602
Fax: (804)693-6785
Email: cfavreau@marinesonic.com
WWW: http://www.marinesonic.com