MSTIFF

MARINE SONIC TECHNOLOGY, LTD.
IMAGE DATA FILE FORMAT

AUGUST 2011

Marine Sonic Technology, Ltd.
5508 George Washington Memorial Highway
P.O. Box 730, White Marsh, VA 23183-0730

COPYRIGHT NOTICE

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

TECHNICAL SUPPORT

Technical support is available via voice, fax, or Internet.

Voice : 804-693-9602
800-447-4804
Fax : 804-693-6785
Web : www.marinesonic.com

(This page left intentionally blank)

Table of Contents

1 Overview

By default the sonar image data file uses the *.mst* extension. The use of this specific extension facilitates the search of MSTL Sea Scan PC data files by the Sea Scan PC Review application.

This document outlines the basic data file format for the MSTIFF data files. MSTIFF has been designed to be a basic data format for the Marine Sonic Technology, Ltd. sonar data image files. A high priority was given to structuring the data in such a way to minimize any difficulties for future additions. This format allows for additions and enhancements to the data file as the capabilities of the Sea Scan PC increase. MSTIFF was designed to be an extensible data format. The MSTIFF format is modeled after the TIFF (Tagged Interchange File Format) Specification Revision 5.0 © Aldus Corporation 1990. Although TIFFs allow for customization of the format, MSTL decided it was better to use the basic structure and create our own MSTL specific tags instead of trying to fit all of our proprietary information into the TIFF.

2 Structure

In the MSTIFF, individual fields are identified with a unique tag. This allows the data format to be extensible, allowing particular fields to be present or absent from the file.

An MSTIFF file begins with an 8-byte "image file header" that points to an "image file directory." The "image file directory" contains information about the sonar and navigational data, as well as pointers to the actual sonar and navigational data.

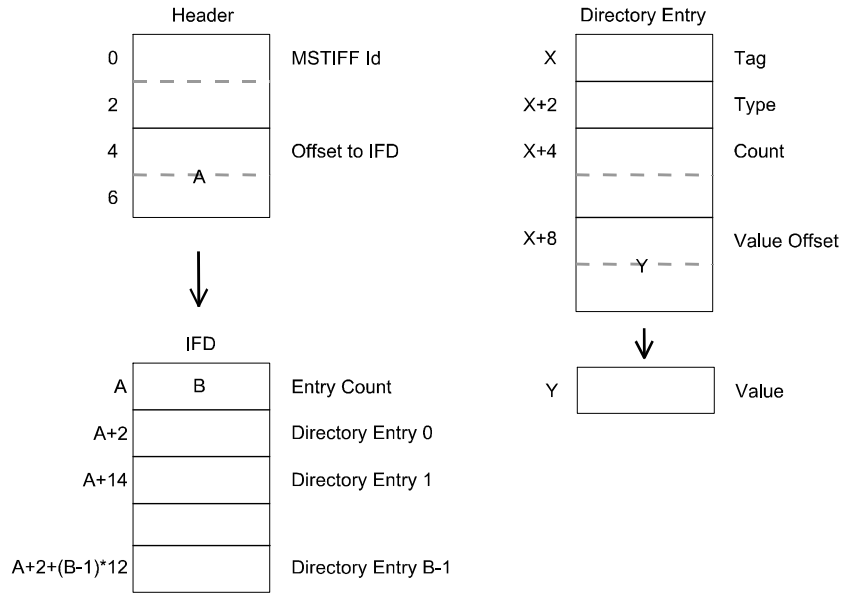

Figure 1 MSTIFF Structure

2.1 Image File Header

An MSTIFF file begins with an 8-byte image file header, containing the following information:

Bytes 0-3: **MSTIFF Identifier**
The first long word of the file is the MSTIFF identifier. This number in hexadecimal format is `4C 54 53 4D`. These four values will appear as ASCII '`MSTL`' in a binary file. An MSTIFF file does not have a real version/revision number. This was a conscious decision in keeping with the philosophy incorporated by the designers of TIFF.

Bytes 4-7: **Image File Directory Offset**
This long word contains the offset (in bytes) of the Image File Directory (IFD). The directory may be at any location in the file after the header. *In particular, an IFD may follow the actual data it describes. When reading an MSTIFF, you must simply follow the pointers, wherever they may lead.*

2.2 Image File Directory

The Image File Directory (IFD) consists of a 2-byte count of the number of entries. This is followed by the sequence of 12-byte field entries

Each 12-byte IFD entry has the following format:

Bytes 0-1: **Tag**
The individual Tags are described in the next section.

Bytes 2-3: **Type**
The data types and their associated lengths are described as follows:

1= BYTE	8-bit unsigned integer
2= ASCII	8-bit bytes for ASCII; last byte must be null (C-string)
3= SHORT	16-bit (2-byte) unsigned integer
4= LONG	32-bit (4-byte) unsigned integer
5= STRUCT	C Structure (defined by Tag)

Bytes 4-7: **Count**
The Count is specified in terms of the data Type, not the total number of bytes. For example a single 16-bit word (SHORT) has a Count of 1, not 2.

The value of the Count part of an ASCII field entry includes the last null byte. An ASCII string with an ending null byte is the standard definition for a C-type string. Note that there is no "count byte" in the ASCII string such as there is for Pascal-type strings. The Count part of the field takes care of this.

Bytes 8-11: **Value Offset**
The Value Offset is the file offset (in bytes) of the **Value** of the field. *This file offset may point to anywhere in the file.*

If the Value fits into 4 bytes, then the Value Offset is interpreted to contain the Value instead of the pointing to the Value. If the Value is less than 4 bytes, it is left justified within the 4-byte Value Offset (i.e. stored in the lower-numbered bytes). Examine the Type and Count fields to determine if the Value fits into 4 bytes.

3 Fields

This section describes the fields defined in the latest MSTIFF version. The fields are sorted by the tag identifier. The documentation for each field contains the following information:

Field Name

Tag: unique identifier for this field
Type: field type as described above
N: the expected number of values
Default: default value if there is one
Comments: description about the field
Dependency: dependent fields

If you are reading the MSTIFF and a specific field does not exist, you must assume the default value.

3.1 Annotation

Tag: 273 (111h)
Type: STRUCT
N: Value from AnnotationCount field

Comments: This field contains the annotation structures. The structure declaration in the C programming language for the Annotation Record (ar) is as follows:

```
struct AnnotationRec {      // ar
        char        strAnno[128];     // annotation string
        float       fLatitude;        // latitude of annotation
        float       fLongitude;       // longitude of annotation
        short       iX;               // horizontal position in Screen Buffer
        short       iLine;            // vertical line position in Screen Buffer
        short       iLinesToEnd;      // number of lines to end of Screen Buffer
};
typedef struct AnnotationRec *AnnotationRecPtr, **AnnotationRecHdl;
```

Each ar contains six fields.

1. Annotation string [128 byte C string]

   This field contains the description string for the annotation as a 128-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 128 characters.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the annotation. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Horizontal Position [short]

   This field contains the horizontal drawing position of the annotation in the screen buffer.

5. Vertical Position [short]

   This field contains the vertical drawing position of the annotation in the screen buffer.

6. Number Lines to End [short]

   This field contains the number of lines from the annotations vertical position to the end of the screen buffer. This value was used in the data collection application to clear annotations that had scrolled off the display screen.

Dependency: AnnotationCount, ScrollDirection

3.2 AnnotationCount

Tag: 272 (110h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of Annotation Records that are included in the data file.

Dependency: None

3.3 Annotation2

Tag: 279 (117h)
Type: STRUCT
N: Value from Annotation2Count field

Comments: This field contains the annotation structures. The structure declaration in the C programming language for the version 2 Annotation Record (ar2) is as follows:

```
struct AnnotationRec2 {       // ar2
        DWORD          dwSysTime;     // system time stamp
        unsigned long  ulTime;        // JulianTime: seconds since midnight
        unsigned long  ulDate;        // JulianDate: YYDDD

        char           strAnno[128];  // annotation string
        float          fLatitude;     // latitude of annotation
        float          fLongitude;    // longitude of annotation
        short          iX;            // horizontal position in Screen Buffer
        short          iLine;         // vertical line position in Screen Buffer
        short          iLinesToEnd;   // number of lines to end of Screen Buffer
};
typedef struct AnnotationRec2 *AnnotationRec2Ptr, **AnnotationRec2Hdl;
```

Each ar2 contains nine fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line that is associated with the annotation..

2. Julian Time [unsigned long]

   This field contains the Julian time for the annotation. This is the number of seconds since midnight of the sonar data line that is associated with the annotation.

3. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the annotation. The format of the Julian Date is described in detail later in this section.

4. Annotation string [128 byte C string]

   This field contains the description string for the annotation as a 128-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 128 characters.

5-6. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the annotation. The L/L are stored in a proprietary format. This format is described in detail later in this section.

7. Horizontal Position [short]

   This field contains the horizontal drawing position of the annotation in the screen buffer.

8. Vertical Position [short]

   This field contains the vertical drawing position of the annotation in the screen buffer.

9. Number Lines to End [short]

   This field contains the number of lines from the annotations vertical position to the end of the screen buffer. This value was used in the data collection application to clear annotations that had scrolled off the display screen.

Dependency: Annotation2Count, ScrollDirection

3.4 Annotation2Count

Tag: 278 (116h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 2 Annotation Records that are included in the data file.

Dependency: None

3.5 Annotation3

Tag: 281 (119h)
Type: STRUCT
N: Value from Annotation3Count field

Comments: This field contains the annotation structures. The structure declaration in the C programming language for the version 3 Annotation Record (ar3) is as follows:

```
struct AnnotationRec3 {        // ar3
        annotationType atType;         // Annotation type

        DWORD          dwSysTime;      // system time stamp
        unsigned long  ulTime;         // JulianTime: seconds since midnight
        unsigned long  ulDate;         // JulianDate: YYDDD

        char           strAnno[128];   // annotation string
        float          fLatitude;      // latitude of annotation
        float          fLongitude;     // longitude of annotation
        short          iX;             // horizontal position in Screen Buffer
        short          iLine;          // vertical line position in Screen Buffer
        short          iLinesToEnd;    // number of lines to end of Screen Buffer
};
typedef struct AnnotationRec3 *AnnotationRec3Ptr, **AnnotationRec3Hdl;
```

Each ar3 contains ten fields.

1. Annotation type [integer typedef as annotation Type]

   This field contains the annotation type. This value may be either:

$0001	Image Annotation
$0002	Event Annotation

2. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line that is associated with the annotation..

3. Julian Time [unsigned long]

   This field contains the Julian time for the annotation. This is the number of seconds since midnight of the sonar data line that is associated with the annotation.

4. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the annotation. The format of the Julian Date is described in detail later in this section.

5. Annotation string [128 byte C string]

   This field contains the description string for the annotation as a 128-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 128 characters.

6-7. Latitude and Longitude [float] in proprietary format

These fields contain the latitude and longitude (L/L) of the annotation. The L/L are stored in a proprietary format. This format is described in detail later in this section.

8. Horizontal Position [short]

This field contains the horizontal drawing position of the annotation in the screen buffer.

9. Vertical Position [short]

This field contains the vertical drawing position of the annotation in the screen buffer.

10. Number Lines to End [short]

This field contains the number of lines from the annotations vertical position to the end of the screen buffer. This value was used in the data collection application to clear annotations that had scrolled off the display screen.

Dependency: Annotation3Count, ScrollDirection

3.6 Annotation3Count

Tag: 280 (118h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 3 Annotation Records that are included in the data file.

Dependency: None

3.7 BinsPerChannel

Tag: 260 (104h)
Type: SHORT
N: 1
Default: 512

Comments: Number of sample bins per channels. Number of horizontal columns in the sonar data.

Dependency: None

3.8 BitsPerBin

Tag: 258 (102h)
Type: SHORT
N: 1
Default: 8 (with 2 MSB set to 0)

Comments: This tag defines the resolution of the individual samples of the sonar data. Each "bin" is a sample of the sonar data. The number of bits per bin is the sampling resolution. Each "bin" is a 6-bit intensity value, stored as 8-bit data with the 2 MSB set to 0.

Dependency: None

3.9 Compression

Tag: 254 (FEh)
Type: SHORT
N: 1
Default: 1

Comments: Data compression technique. Each of the data compression techniques uses the PKWare Compression Library. The PKWare Compression Library uses the *Sliding Window Dictionary* method of data compression. The larger the "search window" the greater the data compression but the longer the time required.

1= NONE
2= LOW (1024 byte window)
3= MEDIUM (2048 byte window)
4= HIGH (4096 byte window)

Dependency: None

3.10 CondensedImage

Tag: 255 (FFh)
Type: BYTE
N: BITMAP structure and 102 x 200 bitmap

Comments: The CondensedImage field is a complex multi-component field. The field first contains a Borland C++ BITMAP structure. The Borland C++ BITMAP structure defines the height, width, color format, and bit values of a logical bitmap. Note that the bmBits pointer in the BITMAP structure is not a valid pointer.

The Borland C++ BITMAP structure is defined as follows:

```
typedef struct tagBITMAP {          // bm
        short                       bmType;
        short                       bmWidth;
        short                       bmHeight;
        short                       bmWidthBytes;
        BYTE                        bmPlanes;
        BYTE                        bmBitsPixel;
        void FAR*                   bmBits;
} BITMAP;
```

The CondensedImage field then contains the bitmap of the Condensed image as defined by the preceding BITMAP structure.

Dependency: Compression

3.11 CreatorVersion

Tag: 301 (12Dh)
Type: STRUCT
N: 1

Comments: This field contains the version information about the application that created the data file. The structure declaration in the C programming language for the Creator Version record (cvr) is follows:

```
struct CreatorVersionRec {              // cvr
        char                            cApp;
        short                           sMajorRev;
        short                           sMinorRec;
        short                           sBetaRev;
};
typedef struct CreatorVersionRec *CreatorVersionRecPtr, **CreatorVersionRecHdl;
```

Each cvr contains four fields.

1. Creator Application [char]

   This field contains a char that represents the creator application.

   'R' Sea Scan PC Review
   'A' Sea Scan PC (Data Acquisition)

2. Major Rev [short]

   This field contains the major version number.

3. Minor Rev [short]

   This field contains the minor version number.

4. Beta Rev [short]

   This field contains the beta version number. This value will be 0 for a release version.

Dependency: None

3.12 Description

Tag: 256 (100h)
Type: ASCII

Comments: Description of the sonar data file. For example, you may want to attach a general comment to the sonar data that is not appropriate in the annotations.

Dependency: None

3.13 Fathometer

Tag: 287 (11Fh)
Type: STRUCT
N: Value from FathometerCount field

Comments: This field contains the fathometer records. The structure declaration in the C programming language for the Fathometer Record (fr) is as follows:

```
struct FathometerRec {          // fr
        DWORD       dwTime;         // system Timestamp
        float       fWaterDepth;    // Water Depth
};
typedef struct FathometerRec * FathometerRecPtr, ** FathometerRecHdl;
```

Each fr contains two fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming fathometer information. This is the system time that the fathometer information was received via the serial port from the external device.

2. Water Depth [float]

   This field contains the water depth in meters.

Dependency: FathometerCount

3.14 Fathometer2

Tag: 296 (128h)
Type: STRUCT
N: Value from FathometerCount field

Comments: This field contains the fathometer records. The structure declaration in the C programming language for the version 2 Fathometer Record (fr2) is as follows:

```
struct FathometerRec2 {          // fr2
                DWORD            dwTime;          // system Timestamp
                float            fWaterDepth;     // Water Depth
                float            fAltitude;       // Towfish Altitude
};
typedef struct FathometerRec2 * FathometerRec2Ptr, ** FathometerRec2Hdl;
```

Each fr2 contains three fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming fathometer information. This is the system time that the fathometer information was received via the serial port from the external device.

2. Water Depth [float]

   This field contains the total water depth in meters. This is measured from the water line to the floor.

3. Towfish Altitude [float]

   This field contains the towfish altitude in meters. This is measured from the transducer to the floor.

Dependency: FathometerCount

3.15 FathometerCount

Tag: 286 (11Eh)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of Fathometer Records that are included in the data file.

Dependency: None

3.16 History

Tag: 257 (101h)
Type: ASCII

Comments: Modification history of the sonar data file. Records any modifications that have been made to the original sonar data file.

Dependency: None

3.17 LeftChannel

Tag: 263 (107h)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the 6-bit sonar data (0-63) for the Left Channel only. The Left channel data of the sonar record lines is stored contiguously in the file as a single segment.

*Please refer to **5.4***

Range Code *(p. 90) for more information on the storage scheme for the different range codes.*

Dependency: Compression, BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo

3.18 LeftChannel2

Tag: 299 (12Bh)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the 8-bit sonar data (0-255) for the Left Channel only. The Left channel data of the sonar record lines is stored contiguously in the file as a single segment.

*Please refer to **5.4***

Range Code *(p. 90) for more information on the storage scheme for the different range codes.*

Dependency: Compression, BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo2

3.19 LeftChannelOdd

Tag: 310 (136h)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the odd fields for 8-bit sonar data (0-255) for the left over data when sampling at 1024 bytes per channel for the Left Channel only. The Left channel odd data of the sonar record lines is stored contiguously in the file as a single segment.

Dependency: BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo2

3.20 Magnetometer

Tag: 289 (121h)
Type: STRUCT
N: Value from MagnetometerCount field

Comments: This field contains the magnetometer records. The structure declaration in the C programming language for the Magnetometer Record (magr) is as follows:

```
struct MagnetometerRec {        // magr
                float           fReading;       // Reading
                DWORD           dwTime;         // system Timestamp
                unsigned short  uiAttr;         // reading attributes
};
typedef struct MagnetometerRec * MagnetometerRecPtr, ** MagnetometerRecHdl;
```

Each magr contains three fields.

1. Magnetometer Reading [float]

   This field contains the magnetometer reading in gammas.

2. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming fathometer information. This is the system time that the fathometer information was received via the serial port from the external device.

3. Reading Attributes [unsigned short]

   This field contains the magnetometer reading attributes. This value may be either:

$0000	None
$0001	Marker
$0002	Annotation

Dependency: MagnetometerCount

3.21 MagnetometerCount

Tag: 288 (120h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of Magnetometer Records that are included in the data file.

Dependency: None

3.22 MagnetometerParms

Tag: 291 (123h)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the Magnetometer Parameters Record (mpr) is as follows:

```
struct MagParmsRec {            // mpr
                magMode         mm;                 // Magnetometer Mode
                short           iScale1;            // line scale 1
                short           iScale2;            // line scale 2
                BOOL            bCentreAmbient;     // Center around ambient flag
                float           fAmbient;           // Ambient gamma
                BOOL            bSynchWithSonar;    // Synchronize with sonar data flag
};
typedef struct MagParmsRec *MagParmsRecPtr, **MagParmsRecHdl;
```

1. Magnetometer Mode [magMode]

   This field contains an enum value indicating the magnetometer mode. The magMode enum has been defined as follows:

   ```
   enum magMode {MAG_OFF, MAG_856, MAG_866, MAG_876, MAG_880};
   ```

2-3. Scale values [short]

   These fields indicate the gamma scale used in the Magnetometer display window during the data collection.

4. Center ambient flag [integer typedef as Boolean]

   This Boolean flag indicates that the magnetometer readings are to be centered on the ambient field for display in the Magnetometer display window.

5. Ambient [float]

   This field contains the ambient magnetic field in gammas.

6. Synchronize data with sonar [integer typedef as Boolean]

   This Boolean flag indicates that the magnetometer readings are to be synchronized in time with the sonar data for display in the Magnetometer display window.

Dependency: None

3.23 MagnetometerParms2

Tag: 303 (12Fh)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the version 2 Magnetometer Parameters Record (mprv2) is as follows:

```
struct MagParmsRecV2 {          // mpr v2
        magMode     mm;         // Magnetometer Mode
        float       fScale1;    // red line scale
        float       fScale2;    // blue line scale
        BOOL        bCentreAmbient;   // Center around ambient flag
        float       fAmbient;   // Ambient gamma
        BOOL        bSynchWithSonar;  // Synchronize with sonar data flag
};
typedef struct MagParmsRecV2 *MagParmsRecV2Ptr, **MagParmsRecV2Hdl;
```

1. Magnetometer Mode [magMode]

   This field contains an enum value indicating the magnetometer mode. The magMode enum has been defined as follows:

   ```
   enum magMode {MAG_OFF, MAG_856, MAG_866, MAG_876, MAG_880};
   ```

2-3. Scale values [float]

   These fields indicate the gamma scale used in the Magnetometer display window during the data collection.

4. Center ambient flag [integer typedef as Boolean]

   This Boolean flag indicates that the magnetometer readings are to be centered on the ambient field for display in the Magnetometer display window.

5. Ambient [float]

   This field contains the ambient magnetic field in gammas.

6. Synchronize data with sonar [integer typedef as Boolean]

   This Boolean flag indicates that the magnetometer readings are to be synchronized in time with the sonar data for display in the Magnetometer display window.

Dependency: None

3.24 Marker

Tag: 269 (10Dh)
Type: STRUCT
N: Value from MarkerCount field

Comments: This field contains the marker records. The structure declaration in the C programming language for the Marker Record (mr) is as follows:

```
struct MarkerRec {              // mr
        float       fLatitude;      // Latitude
        float       fLongitude;     // Longitude
        markerType  mtType;         // Marker type
        char        strId[32];      // Identifier string
};
typedef struct MarkerRec *MarkerRecPtr, **MarkerRecHdl;
```

Each mr contains four fields.

1-2. Latitude and Longitude [float] in proprietary format

These fields contain the latitude and longitude (L/L) of the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

3. Marker type [integer typedef as markerType]

This field contains the marker type. This value may be either:

$0001	Waypoint marker (Green Cross)
$0002	Nadir marker (Purple Square)
$0004	Estimated marker (Red Square)

4. Identifier string [32 byte C string]

This field contains the identifier string for the marker as a 32-byte C string. The identifier string, including the required '\0' terminator for a C string, cannot be more than 32 characters.

Dependency: MarkerCount

3.25 MarkerCount

Tag: 268 (10Ch)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of Marker Records that are included in the data file.

Dependency: None

3.26 Marker2

Tag: 277 (115h)
Type: STRUCT
N: Value from Marker2Count field

Comments: This field contains the marker records. The structure declaration in the C programming language for the version 2 Marker Record (mr2) is as follows:

```
struct MarkerRec2 {          // mr2
        markerType    mtType;            // Marker type
        unsigned short uiId;             // unique identifier
        char          strDataFile[145];  // pathname to data file
        char          strDescriptor[32]; // Description string

        DWORD         dwSysTime;         // system time stamp
        unsigned long ulTime;            // JulianTime: seconds since midnight
        unsigned long ulDate;            // JulianDate: YYDDD

        float         fTargetHeight;     // Target Attributes: Height
        float         fTargetSlantRange; //                    Slant Range
        float         fTargetRange;      //                    Range
        float         fTargetLat;        //                    Latitude
        float         fTargetLong;       //                    Longitude

        float         fTowfishHeight;    // Towfish Attributes:    Height
        float         fTowfishLat;       //                        Latitude
        float         fTowfishLong;      //                        Longitude
};
typedef struct MarkerRec2 *MarkerRec2Ptr, **MarkerRec2Hdl;
```

Each mr2 contains fifteen fields.

1. Marker type [integer typedef as markerType]

   This field contains the marker type. This value may be either:

$0001	Waypoint marker (Green Cross)
$0002	Nadir marker (Purple Square)
$0004	Estimated marker (Red Square)

2. Identifier [unsigned short]

   This field contains the unique identifier for the marker. This identifier is assigned at the time the marker is created and may be changed by the reader.

3. Data filename string [145 byte C string]

   This field contains the pathname (including the filename) of the file that contains the sonar data for the marker as a 145-byte C string. The pathname string, including the required '\0' terminator for a C string, cannot be more than 145 characters.

4. Description string [32 byte C string]

   This field contains the description string for the marker as a 32-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 32 characters.

5. Timestamp [long word typedef as DWORD]

This field contains the timestamp for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the system time as invalid.

6. Julian Time [unsigned long]

   This field contains the Julian time for the marker. This is the number of seconds since midnight of the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the time as invalid.

7. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the date as invalid. The format of the Julian Date is described in detail later in this section.

8. Target Height [float]

   This field contains the height of the target associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

9. Target Slant Range [float]

   This field contains the slant range of the target associated with the marker in meters. This field is set to 0.0 to mark the slant range as invalid.

10. Target Range [float]

    This field contains the range of the target associated with the marker in meters. This field is set to 0.0 to mark the range as invalid.

11-12. Target Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the target associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

13. Towfish Height [float]

    This field contains the height of the towfish for the sonar data line associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

14-15. Towfish Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the towfish for the sonar data line associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

Dependency: Marker2Count

3.27 Marker2Count

Tag: 276 (114h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 2 Marker Records that are included in the data file.

Dependency: None

3.28 Marker3

Tag: 284 (11Ch)
Type: STRUCT
N: Value from Marker3Count field

Comments: This field contains the marker records. The structure declaration in the C programming language for the version 3 Marker Record (mr3) is as follows:

```
struct MarkerRec3 {          // mr3
        markerType      mtType;             // Marker type
        unsigned short  uiId;               // unique identifier
        char            strDataFile[145];   // pathname to data file
        char            strDescriptor[32];  // Description string

        DWORD           dwSysTime;          // system time stamp
        unsigned long   ulTime;             // JulianTime: seconds since midnight
        unsigned long   ulDate;             // JulianDate: YYDDD

        float           fTargetHeight;      // Target Attributes:       Height
        float           fTargetSlantRange;  //                  Slant Range
        float           fTargetRange;       //                  Range
        float           fTargetLat;         //                  Latitude
        float           fTargetLong;        //                  Longitude

        float           fTowfishHeight;     // Towfish Attributes:      Height
        float           fTowfishLat;        //                  Latitude
        float           fTowfishLong;       //                  Longitude

        float           fWaterDepth;        // Water Depth
};
typedef struct MarkerRec3 *MarkerRec3Ptr, **MarkerRec3Hdl;
```

Each mr3 contains sixteen fields.

1. Marker type [integer typedef as markerType]

   This field contains the marker type. This value may be either:

$0001	Waypoint marker (Green Cross)
$0002	Nadir marker (Purple Square)
$0004	Estimated marker (Red Square)
$0008	Magnetometer marker (Red square and black ‘M’)

2. Identifier [unsigned short]

   This field contains the unique identifier for the marker. This identifier is assigned at the time the marker is created and may be changed by the reader.

3. Data filename string [145 byte C string]

   This field contains the pathname (including the filename) of the file that contains the sonar data for the marker as a 145-byte C string. The pathname string, including the required ‘\0’ terminator for a C string, cannot be more than 145 characters.

4. Description string [32 byte C string]

   This field contains the description string for the marker as a 32-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 32 characters.

5. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the system time as invalid.

6. Julian Time [unsigned long]

   This field contains the Julian time for the marker. This is the number of seconds since midnight of the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the time as invalid.

7. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the date as invalid. The format of the Julian Date is described in detail later in this section.

8. Target Height [float]

   This field contains the height of the target associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

9. Target Slant Range [float]

   This field contains the slant range of the target associated with the marker in meters. This field is set to 0.0 to mark the slant range as invalid.

10. Target Range [float]

    This field contains the range of the target associated with the marker in meters. This field is set to 0.0 to mark the range as invalid.

11-12. Target Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the target associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

13. Towfish Height [float]

    This field contains the height of the towfish for the sonar data line associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

14-15. Towfish Latitude and Longitude [float] in proprietary format

These fields contain the latitude and longitude (L/L) of the towfish for the sonar data line associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

16. Water Depth [float]

    This field contains the water depth for the sonar data line associated with the marker in meters. This field is set to –1.0 to mark the depth as invalid.

Dependency: Marker3Count

3.29 Marker3Count

Tag: 283 (11Bh)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 3 Marker Records that are included in the data file.

Dependency: None

3.30 Marker4

Tag: 295 (127h)
Type: STRUCT
N: Value from Marker4Count field

Comments: This field contains the marker records. The structure declaration in the C programming language for the version 4 Marker Record (mr4) is as follows:

```
struct MarkerRec4 {        // mr4
        markerType      mtType;             // Marker type
        unsigned short  uiId;               // unique identifier
        char            strDataFile[145];   // pathname to data file
        char            strDescriptor[32];  // Description string

        DWORD           dwSysTime;          // system time stamp
        unsigned long   ulTime;             // JulianTime: seconds since midnight
        unsigned long   ulDate;             // JulianDate: YYDDD

        float           fTargetHeight;      // Target Attributes:  Height
        float           fTargetSlantRange;  //                     Slant Range
        float           fTargetRange;       //                     Range
        float           fTargetLat;         //                     Latitude
        float           fTargetLong;        //                     Longitude

        float           fTowfishAltitude;   // Towfish Attributes:   Altitude

        float           fNavLat;            // Navigation:         Latitude
        float           fNavLong;           //                     Longitude

        BOOL            bLaybackIsOn;       // Layback:            enable flag
        float           fLaybackX;          //                     X (lateral )layback
        float           fLaybackY;          //                     Y (axial) layback

        float           fWaterDepth;        // Water Depth
};
typedef struct MarkerRec4 *MarkerRec4Ptr, **MarkerRec4Hdl;
```

Each mr4 contains nineteen fields.

1. Marker type [integer typedef as markerType]

   This field contains the marker type. This value may be either:

$0001	Waypoint marker (Green Cross)
$0002	Nadir marker (Purple Square)
$0004	Estimated marker (Red Square)
$0008	Magnetometer marker (Red square and black ‘M’)

2. Identifier [unsigned short]

   This field contains the unique identifier for the marker. This identifier is assigned at the time the marker is created and may be changed by the reader.

3. Data filename string [145 byte C string]

   This field contains the pathname (including the filename) of the file that contains the sonar data for the marker as a 145-byte C string. The pathname string, including the required ‘\0’ terminator for a C string, cannot be more than 145 characters.

4. Description string [32 byte C string]

   This field contains the description string for the marker as a 32-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 32 characters.

5. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the system time as invalid.

6. Julian Time [unsigned long]

   This field contains the Julian time for the marker. This is the number of seconds since midnight of the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the time as invalid.

7. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the date as invalid. The format of the Julian Date is described in detail later in this section.

8. Target Height [float]

   This field contains the height of the target associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

9. Target Slant Range [float]

   This field contains the slant range of the target associated with the marker in meters. This field is set to 0.0 to mark the slant range as invalid.

10. Target Range [float]

    This field contains the range of the target associated with the marker in meters. This field is set to 0.0 to mark the range as invalid.

11-12. Target Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the target associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

13. Towfish Altitude [float]

    This field contains the altitude of the towfish for the sonar data line associated with the marker in meters. This field is set to a negative value of -1.0 to mark the altitude as invalid.

14-15. Survey Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the survey vessel for the sonar data line associated with the marker.

The L/L are stored in a proprietary format. This format is described in detail later in this section.

16. Layback is Enabled[integer typedef as Boolean]

    This field contains the Boolean flag indicating the layback state. If the flag is 0, the layback was disabled at this position fix. Otherwise, the layback was enabled at this position fix.

17. Layback – X (Lateral) [float]

    This field defines the lateral layback of the transducer from the null point of the survey vessel. A negative and positive value for the X distance sets the layback to the left and right respectively of the null point of the survey vessel. The layback is measured in meters.

18. Layback – Y (Axial) [float]

    This field defines the axial layback of the transducer from the null point of the survey vessel. A negative and positive value for the Y distance sets the layback to the fore and aft respectively of the null point of the survey vessel. The layback is measured in meters.

19. Water Depth [float]

    This field contains the water depth for the sonar data line associated with the marker in meters. This field is set to a negative value of -1.0 to mark the depth as invalid.

Dependency: Marker4Count

3.31 Marker4Count

Tag: 294 (126h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 4 Marker Records that are included in the data file.

Dependency: None

3.32 Marker5

Tag: 307 (133h)
Type: STRUCT
N: Value from Marker5Count field

Comments: This field contains the marker records. The structure declaration in the C programming language for the version 5 Marker Record (mr5) is as follows:

```
struct MarkerRec5 {        // mr5
        markerType      mtType;             // Marker type
        unsigned short  uiId;               // unique identifier
        char            strDataFile[145];   // pathname to data file
        char            strDescriptor[32];  // Description string

        DWORD           dwSysTime;          // system time stamp
        unsigned long   ulTime;             // JulianTime: seconds since midnight
        unsigned long   ulDate;             // JulianDate: YYDDD

        float           fTargetHeight;      // Target Attributes:  Height
        float           fTargetSlantRange;  //                     Slant Range
        float           fTargetRange;       //                     Range
        float           fTargetLat;         //                     Latitude
        float           fTargetLong;        //                     Longitude

        float           fTowfishAltitude;   // Towfish Attributes:   Altitude

        float           fNavLat;            // Navigation:         Latitude
        float           fNavLong;           //                     Longitude

        BOOL            bLaybackIsOn;       // Layback:            enable flag
        float           fLaybackX;          //                     X (lateral )layback
        float           fLaybackY;          //                     Y (axial) layback
        float           fLaybackLat;        //                     Latitude
        float           fLaybackLong;       //                     Longitude

        float           fWaterDepth;        // Water Depth
};
typedef struct MarkerRec5 *MarkerRec5Ptr, **MarkerRec5Hdl;
```

Each mr5 contains twenty-one fields.

1. Marker type [integer typedef as markerType]

   This field contains the marker type. This value may be either:

$0001	Waypoint marker (Green Cross)
$0002	Nadir marker (Purple Square)
$0004	Estimated marker (Red Square)
$0008	Magnetometer marker (Red square and black ‘M’)

2. Identifier [unsigned short]

   This field contains the unique identifier for the marker. This identifier is assigned at the time the marker is created and may be changed by the reader.

3. Data filename string [145 byte C string]

   This field contains the pathname (including the filename) of the file that contains the sonar data for the marker as a 145-byte C

string. The pathname string, including the required '\0' terminator for a C string, cannot be more than 145 characters.

4. Description string [32 byte C string]

   This field contains the description string for the marker as a 32-byte C string. The description string, including the required '\0' terminator for a C string, cannot be more than 32 characters.

5. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the system time as invalid.

6. Julian Time [unsigned long]

   This field contains the Julian time for the marker. This is the number of seconds since midnight of the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the time as invalid.

7. Julian Date [unsigned long]

   This field contains the Julian date for the sonar data line that is associated with the marker. If the marker does not have an associated sonar data line, this field is set to 0 to mark the date as invalid. The format of the Julian Date is described in detail later in this section.

8. Target Height [float]

   This field contains the height of the target associated with the marker in meters. This field is set to –1.0 to mark the height as invalid.

9. Target Slant Range [float]

   This field contains the slant range of the target associated with the marker in meters. This field is set to 0.0 to mark the slant range as invalid.

10. Target Range [float]

    This field contains the range of the target associated with the marker in meters. This field is set to 0.0 to mark the range as invalid.

11-12. Target Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the target associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

13. Towfish Altitude [float]

    This field contains the altitude of the towfish for the sonar data line associated with the marker in meters. This field is set to a negative value of -1.0 to mark the altitude as invalid.

14-15. Survey Latitude and Longitude [float] in proprietary format

These fields contain the latitude and longitude (L/L) of the survey vessel for the sonar data line associated with the marker. The L/L are stored in a proprietary format. This format is described in detail later in this section.

16. Layback is Enabled[integer typedef as Boolean]

    This field contains the Boolean flag indicating the layback state. If the flag is 0, the layback was disabled at this position fix. Otherwise, the layback was enabled at this position fix.

17. Layback – X (Lateral) [float]

    This field defines the lateral layback of the transducer from the null point of the survey vessel. A negative and positive value for the X distance sets the layback to the left and right respectively of the null point of the survey vessel. The layback is measured in meters.

18. Layback – Y (Axial) [float]

    This field defines the axial layback of the transducer from the null point of the survey vessel. A negative and positive value for the Y distance sets the layback to the fore and aft respectively of the null point of the survey vessel. The layback is measured in meters.

19-20. Layback Latitude and Longitude [float] in proprietary format

    These fields contain the latitude and longitude (L/L) of the layback of the transducer from the null point of the survey vessel. The L/L are stored in a proprietary format. This format is described in detail later in this section.

21. Water Depth [float]

    This field contains the water depth for the sonar data line associated with the marker in meters. This field is set to a negative value of -1.0 to mark the depth as invalid.

Dependency: Marker4Count

3.33 Marker5Count

Tag: 306 (132h)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of version 5 Marker Records that are included in the data file.

Dependency: None

3.34 NavInfo

Tag: 267 (10Bh)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the Navigation Record (nr) is as follows:

```
struct NavRec {                 // nr
        DWORD       dwSysTime;          // system timestamp
        float       fLatitude;          // Latitude (proprietary format)
        float       fLongitude;         // Longitude (proprietary format)
        float       fSOGKnots;          // speed-over-ground in knots
        float       fCOG;               // course-over-ground (true)
        float       fTD1;               // Loran-C time delay 1 (if avail)
        float       fTD2;               // Loran-C time delay 2 (if avail)
        float       fPortStartPtLat;    // Port swath coverage
        float       fPortStartPtLong;
        float       fPortEndPtLat;
        float       fPortEndPtLong;
        float       fStbdStartPtLat;    // Starboard swath coverage
        float       fStbdStartPtLong;
        float       fStbdEndPtLat;
        float       fStbdEndPtLong;
        BOOL        bSonarIsActive;     // flag indicating sonar is active
};
typedef struct NavRec *NavRecPtr, **NavRecHdl;
```

Each nr contains sixteen fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

   This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8-11. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

12-15. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

16. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

Dependency: NavInfoCount

3.35 NavInfo2

Tag: 275 (113h)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the version 2 Navigation Record (nr2) is as follows:

```
struct NavRec2 {                // nr2
        DWORD       dwSysTime;          // system timestamp
        float       fLatitude;          // Latitude (proprietary format)
        float       fLongitude;         // Longitude (proprietary format)
        float       fSOGKnots;          // speed-over-ground in knots
        float       fCOG;               // course-over-ground (true)
        float       fTD1;               // Loran-C time delay 1 (if avail)
        float       fTD2;               // Loran-C time delay 2 (if avail)
        float       fDepth;             // Added since NavInfo
        float       fAltitude;          // Added since NavInfo
        float       fHeading;           // Added since NavInfo
        float       fPortStartPtLat;    // Port swath coverage
        float       fPortStartPtLong;
        float       fPortEndPtLat;
        float       fPortEndPtLong;
        float       fStbdStartPtLat;    // Starboard swath coverage
        float       fStbdStartPtLong;
        float       fStbdEndPtLat;
        float       fStbdEndPtLong;
        BOOL        bSonarIsActive;     // flag indicating sonar is active
};
typedef struct NavRec2 *NavRec2Ptr, **NavRec2Hdl;
```

Each nr2 contains nineteen fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

   This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8. Towfish Depth [float]

This field contains the towfish depth. If the depth was not available this field is set to 99999.9 to mark the depth as invalid.

9. Towfish Altitude [float]

This field contains the towfish altitude. If the altitude was not available this field is set to 99999.9 to mark the altitude as invalid.

10. Towfish Heading [float]

This field contains the towfish heading. If the heading was not available this field is set to 99999.9 to mark the heading as invalid.

11-14. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

15-18. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

19. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

Dependency: NavInfoCount

3.36 NavInfo3

Tag: 282 (11Ah)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the version 3 Navigation Record (nr3) is as follows:

```
struct NavRec3 {          // nr3
        DWORD     dwSysTime;          // system timestamp
        float     fLatitude;          // Latitude (proprietary format)
        float     fLongitude;         // Longitude (proprietary format)
        float     fSOGKnots;          // speed-over-ground in knots
        float     fCOG;               // course-over-ground (true)
        float     fTD1;               // Loran-C time delay 1 (if avail)
        float     fTD2;               // Loran-C time delay 2 (if avail)
        float     fPortStartPtLat;    // Port swath coverage
        float     fPortStartPtLong;
        float     fPortEndPtLat;
        float     fPortEndPtLong;
        float     fStbdStartPtLat;    // Starboard swath coverage
        float     fStbdStartPtLong;
        float     fStbdEndPtLat;
        float     fStbdEndPtLong;
        BOOL      bSonarIsActive;     // flag indicating sonar is active
};
typedef struct NavRec3 *NavRec3Ptr, **NavRec3Hdl;
```

Each nr3 contains sixteen fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

   This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8-11. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

12-15. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

16. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

Dependency: NavInfoCount

3.37 NavInfo4

Tag: 293 (125h)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the version 4 Navigation Record (nr4) is as follows:

```
struct NavRec4 {          // nr4
        DWORD     dwSysTime;          // system timestamp
        float     fLatitude;          // Latitude (proprietary format)
        float     fLongitude;         // Longitude (proprietary format)
        float     fSOGKnots;          // speed-over-ground in knots
        float     fCOG;               // course-over-ground (true)
        float     fTD1;               // Loran-C time delay 1 (if avail)
        float     fTD2;               // Loran-C time delay 2 (if avail)
        float     fPortStartPtLat;    // Port swath coverage
        float     fPortStartPtLong;
        float     fPortEndPtLat;
        float     fPortEndPtLong;
        float     fStbdStartPtLat;    // Starboard swath coverage
        float     fStbdStartPtLong;
        float     fStbdEndPtLat;
        float     fStbdEndPtLong;
        BOOL      bSonarIsActive;     // flag indicating sonar is active
        float     fLaybackX;          // Lateral Layback (port/stbd)
        float     fLaybackY;          // Axial layback (fore/aft)
        BOOL      bLaybackEnabled;    // flag indicating layback enabled
};
typedef struct NavRec4 *NavRec4Ptr, **NavRec4Hdl;
```

Each nr4 contains nineteen fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

   This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8-11. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

12-15. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

16. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

17. Layback – X (Lateral) [float]

This field defines the lateral layback of the transducer from the null point of the survey vessel. A negative and positive value for the X distance sets the layback to the left and right respectively of the null point of the survey vessel. The layback is measured in meters.

18. Layback – Y (Axial) [float]

This field defines the axial layback of the transducer from the null point of the survey vessel. A negative and positive value for the Y distance sets the layback to the fore and aft respectively of the null point of the survey vessel. The layback is measured in meters.

19. Layback is Enabled[integer typedef as Boolean]

This field contains the Boolean flag indicating the layback state. If the flag is 0, the layback was disabled at this position fix. Otherwise, the layback was enabled at this position fix.

Dependency: NavInfoCount

3.38 NavInfo5

Tag: 297 (129h)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the version 5 Navigation Record (nr5) is as follows:

```
struct NavRec5 {          // nr5
        DWORD     dwSysTime;          // system timestamp
        float     fLatitude;          // Latitude (proprietary format)
        float     fLongitude;         // Longitude (proprietary format)
        float     fSOGKnots;          // speed-over-ground in knots
        float     fCOG;               // course-over-ground (true)
        float     fTD1;               // Loran-C time delay 1 (if avail)
        float     fTD2;               // Loran-C time delay 2 (if avail)
        float     fHeading;           // towfish heading in degrees
        float     fPortStartPtLat;    // Port swath coverage
        float     fPortStartPtLong;
        float     fPortEndPtLat;
        float     fPortEndPtLong;
        float     fStbdStartPtLat;    // Starboard swath coverage
        float     fStbdStartPtLong;
        float     fStbdEndPtLat;
        float     fStbdEndPtLong;
        BOOL      bSonarIsActive;     // flag indicating sonar is active
        float     fLaybackX;          // Lateral Layback (port/stbd)
        float     fLaybackY;          // Axial layback (fore/aft)
        BOOL      bLaybackEnabled;    // flag indicating layback enabled
};
typedef struct NavRec5 *NavRec5Ptr, **NavRec5Hdl;
```

Each nr5 contains twenty fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

   This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8. Towfish Heading [float]

This field contains the towfish heading. If the heading was not available this field is set to 99999.9 to mark the heading as invalid.

9-12. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

13-16. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

17. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

18. Layback – X (Lateral) [float]

This field defines the lateral layback of the transducer from the null point of the survey vessel. A negative and positive value for the X distance sets the layback to the left and right respectively of the null point of the survey vessel. The layback is measured in meters.

19. Layback – Y (Axial) [float]

This field defines the axial layback of the transducer from the null point of the survey vessel. A negative and positive value for the Y distance sets the layback to the fore and aft respectively of the null point of the survey vessel. The layback is measured in meters.

20. Layback is Enabled[integer typedef as Boolean]

This field contains the Boolean flag indicating the layback state. If the flag is 0, the layback was disabled at this position fix. Otherwise, the layback was enabled at this position fix.

Dependency: NavInfoCount

3.39 NavInfo6

Tag: 308 (134h)
Type: STRUCT
N: Value from NavInfoCount field

Comments: This field contains the Navigational Information records. Each record contains the navigational information at an associated system time.

The structure declaration in the C programming language for the version 6 Navigation Record (nr6) is as follows:

```
struct NavRec6 {          // nr6
        DWORD     dwSysTime;          // system timestamp
        float     fLatitude;          // Latitude (proprietary format)
        float     fLongitude;         // Longitude (proprietary format)
        float     fSOGKnots;          // speed-over-ground in knots
        float     fCOG;               // course-over-ground (true)
        float     fTD1;               // Loran-C time delay 1 (if avail)
        float     fTD2;               // Loran-C time delay 2 (if avail)
        float     fHeading;           // towfish heading in degrees
        float     fPortStartPtLat;    // Port swath coverage
        float     fPortStartPtLong;
        float     fPortEndPtLat;
        float     fPortEndPtLong;
        float     fStbdStartPtLat;    // Starboard swath coverage
        float     fStbdStartPtLong;
        float     fStbdEndPtLat;
        float     fStbdEndPtLong;
        BOOL      bSonarIsActive;     // flag indicating sonar is active
        float     fLaybackX;          // Lateral Layback (port/stbd)
        float     fLaybackY;          // Axial layback (fore/aft)
        BOOL      bLaybackEnabled;    // flag indicating layback enabled
        float     fInterNavDistance;  // distance in meters from previous fix
};
typedef struct NavRec6 *NavRec6Ptr, **NavRec6Hdl;
```

Each nr6 contains twenty-one fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the incoming navigational information. This is the system time that the navigational information was received via the serial port from the external navigational device.

2-3. Latitude and Longitude [float] in proprietary format

   These fields contain the latitude and longitude (L/L) of the position fix from the external navigational device. The L/L are stored in a proprietary format. This format is described in detail later in this section.

4. Speed-Over-Ground [float]

   This field contains the speed-over-ground (SOG) in knots. The SOG has a range of 0.0 to 9.9 knots.

5. Course-Over-Ground [float]

This field contains the true course-over-ground (COG). The COG has a range of 0.0 to 359.9 degrees. 0.0 degrees is due north and 90.0 degrees is due east.

6-7. Time delays [float]

These fields contain the primary and secondary time delays used by a Loran-C to compute the L/L. These values are only valid if the external navigational device used for navigational information was a Loran-C. Otherwise these two fields are set to 99999.9 to mark the time delays as invalid.

8. Towfish Heading [float]

This field contains the towfish heading. If the heading was not available this field is set to 99999.9 to mark the heading as invalid.

9-12. Port Swath Coverage [float]

These fields define the area covered by the port swath in L/L coordinates. These values are pre-defined in L/L for quick drawing by the Sea Scan PC plotter. The Start point defines the beginning of the range and the End point defines the end of the range. Note that the start point may be some distance from the Towfish location due to a range delay.

13-16. Starboard Swath Coverage [float]

These fields define the area covered by the starboard swath in L/L coordinates.

17. Sonar is Active [integer typedef as Boolean]

This field contains the Boolean flag indicating the sonar state. If the flag is 0, the sonar was turned OFF at this position fix. Otherwise, sonar was turned ON at this position fix.

18-19. Layback – X (Lateral) & Y (Axial) [float]

These fields define the lateral and axial layback of the transducer from the null point of the survey vessel. A negative and positive value for the X distance sets the layback to the left and right respectively of the null point of the survey vessel. Similarly, a negative and positive value for the Y distance sets the layback to the fore and aft respectively of the null point of the survey vessel. The layback is measured in meters.

20. Layback is Enabled[integer typedef as Boolean]

This field contains the Boolean flag indicating the layback state. If the flag is 0, the layback was disabled at this position fix. Otherwise, the layback was enabled at this position fix.

21. Inter-Navigation Fix Distance [float]

This field defines the distance from the previous navigation fixe in meters.

Dependency: NavInfoCount

3.40 NavInfoCount

Tag: 266 (10Ah)
Type: SHORT
N: 1
Default: 0

Comments: This field contains the number of Navigational Information Records that are included in the data file.

Dependency: None

3.41 NavInterpolationTimeout

Tag: 304 (130h)
Type: LONG
N: 1
Default: 10000

Comments: This field contains the timeout limit for the navigation interpolation algorithm in milliseconds. The interpolation algorithm will interpolate between consecutive navigation data points if the time difference between the two points is less than the timeout value.

Dependency: None

3.42 PingNavInfo

Tag: 274 (112h)
Type: STRUCT
N: SonarLines

Comments: This field contains the navigational information for the nadir of each sonar record line. This field is not provided in real-time by the Sea Scan PC. It must be calculated in post-processing by correlating the Navigational Information records with the sonar record lines using the system timestamp. *This is a redundant field, but it is provided to allow fast lookups for the navigational information at the nadir of each ping.* It is useful for applications that need to search the sonar data based on ping nadir location and do not want to correlate the navigational information and sonar data each time.

Use the same structure used for the Navigation Record as described in the MSTIFF NavInfo field. However, fTD1 and fTD2 must be set to 99999.9 to mark the time delays as invalid. In addition, the bDrawSwath field must be set to TRUE.

Dependency: SonarLines

3.43 RightChannel

Tag: 264 (108h)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the 6-bit sonar data (0-63) for the Right Channel only. The Right channel data of the sonar record lines is stored contiguously in the file as a single segment.

*Please refer to **5.4***

Range Code *(p. 90) for more information on the storage scheme for the different range codes.*

Dependency: Compression, BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo

3.44 RightChannel2

Tag: 300 (12Ch)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the 8-bit sonar data (0-255) for the Right Channel only. The Right channel data of the sonar record lines is stored contiguously in the file as a single segment.

*Please refer to **5.4***

Range Code *(p. 90) for more information on the storage scheme for the different range codes.*

Dependency: Compression, BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo2

3.45 RightChannelOdd

Tag: 310 (136h)
Type: BYTE
N: SonarLines x BinsPerChannel

Comments: This field contains the odd fields for 8-bit sonar data (0-255) for the left over data when sampling at 1024 bytes per channel for the Right Channel only. The Right channel odd data of the sonar record lines is stored contiguously in the file as a single segment.

Dependency: BitsPerBin, SonarLines, BinsPerChannel, SonarDataInfo2

3.46 ScrollDirection

Tag: 261 (105h)
Type: SHORT
N: 1
Default: 0

Comments: Data scroll direction.
0= Data scrolls UP, thus Left Channel on right side of screen.
1= Data scrolls DOWN, thus Left Channel on left side of screen.

Dependency: None

3.47 SonarDataInfo

Tag: 265 (109h)
Type: STRUCT
N: SonarLines

Comments: This field contains the sonar data information for each line in the sonar record. Each sonar data line has an associated sdir structure that defines the sonar data. The structure declaration in the C programming language for the Sonar Data Information Record (sdir) is as follows:

```
struct SonarDataInfo1Rec {      // sdir1
                DWORD       dwSysTime;      // system timestamp
                rngValue    rvRangeCode;    // range code
                short       iRangeDelay;    // range delay in bins
                short       iAltitude;      // altitude in bins
};
typedef struct SonarDataInfo1Rec *SonarDataInfo1RecPtr, **SonarDataInfo1RecHdl;
```

Each sdir1 contains four fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line. This is the system time that the ping signal was sent to the Towfish.

2. Range code [integer typedef as rngValue]

   This field contains the range code for the sonar data line. The range code is described in detail later in this section.

3. Range delay [integer]

   This field contains the range delay, recorded in bins as opposed to meters. To convert the range delay from bins to meters:

$$RangeDelay[meter] = RangeDelay[bin] * \frac{Range[meters]}{BinsPerChannel}$$

4. Altitude [integer]

   This field contains the altitude, recorded in bins. Similar to the range delay calculation, to convert the altitude from bins to meters:

$$Altitude[meter] = Altitude[bin] * \frac{Range[meters]}{BinsPerChannel}$$

Dependency: SonarLines

3.48 SonarDataInfo2

Tag: 292 (124h)
Type: STRUCT
N: SonarLines

Comments: This field contains the sonar data information for each line in the sonar record. Each sonar data line has an associated sdir structure that defines the sonar data. The structure declaration in the C programming language for the Sonar Data Information Record (sdir) is as follows:

```
struct SonarDataInfo2Rec {      // sdir2
                DWORD       dwSysTime;      // system timestamp
                rngValue    rvRangeCode;    // range code
                frequency   fActive;        // frequency
                short       iRangeDelay;    // range delay in bins
                short       iAltitude;      // altitude in bins
};
typedef struct SonarDataInfo2Rec *SonarDataInfo2RecPtr, **SonarDataInfo2RecHdl;
```

Each sdir2 contains five fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line. This is the system time that the ping signal was sent to the Towfish.

2. Range code [integer typedef as rngValue]

   This field contains the range code for the sonar data line. The range code is described in detail later in this section.

3. Frequency [integer typedef as frequency]

   This field contains an enum value indicating the frequency mode. The frequency enum has been defined as follows:

   \enum MSTIFF_FREQUENCY {FREQ_150, FREQ_300, FREQ_600, FREQ_1200, FREQ_UNKNOWN, FREQ_900, FREQ_2400, FREQ_1800};

4. Range delay [integer]

   This field contains the range delay, recorded in bins as opposed to meters. To convert the range delay from bins to meters:

$$RangeDelay[meter] = RangeDelay[bin] * \frac{Range[meters]}{BinsPerChannel}$$

5. Altitude [integer]

   This field contains the altitude, recorded in bins. Similar to the range delay calculation, to convert the altitude from bins to meters:

$$Altitude[meter] = Altitude[bin] * \frac{Range[meters]}{BinsPerChannel}$$

Dependency: SonarLines

3.49 SonarDataInfo3

Tag: 298 (12Ah)
Type: STRUCT
N: SonarLines

Comments: This field contains the sonar data information for each line in the sonar record. Each sonar data line has an associated sdir structure that defines the sonar data. The structure declaration in the C programming language for the Sonar Data Information Record (sdir) is as follows:

```
struct SonarDataInfo3Rec {      // sdir3
                DWORD       dwSysTime;      // system timestamp
                rngValue    rvRangeCode;    // range code
                frequency   fActive;        // frequency
                short       iRangeDelay;    // range delay in bins
                short       iAltitude;      // altitude in bins
                LRGainRec   lrgr;           // gain settings
};
typedef struct SonarDataInfo3Rec *SonarDataInfo3RecPtr, **SonarDataInfo3RecHdl;
```

Each sdir contains six fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp for the sonar data line. This is the system time that the ping signal was sent to the Towfish.

2. Range code [integer typedef as rngValue]

   This field contains the range code for the sonar data line. The range code is described in detail later in this section.

3. Frequency [integer typedef as frequency]

   This field contains an enum value indicating the frequency mode. The frequency enum has been defined as follows:

   ```
   enum MSTIFF_FREQUENCY {FREQ_150, FREQ_300, FREQ_600, FREQ_1200, FREQ_UNKNOWN, FREQ_900, FREQ_2400, FREQ_1800};
   ```

4. Range delay [integer]

   This field contains the range delay, recorded in bins as opposed to meters. To convert the range delay from bins to meters:

   $$RangeDelay[meter] = RangeDelay[bin] * \frac{Range[meters]}{BinsPerChannel}$$

5. Altitude [integer]

   This field contains the altitude, recorded in bins. Similar to the range delay calculation, to convert the altitude from bins to meters:

   $$Altitude[meter] = Altitude[bin] * \frac{Range[meters]}{BinsPerChannel}$$

6. Gain [struct]

This field contains the gain settings for the left and right channels. The time-gain compensation curve that is applied to the raw sonar data is derived from this information. Each gain setting is an 8-bit value stored as a 16-bit short. The eight separate gain settings are applied at the following ranges. The gain structures are defined as follows:

```
struct GainRec {        // gr
        short           iGain[NUM_GAINPTS];         // gain
};
typedef struct GainRec *GainRecPtr, **GainRecHdl;

struct LRGainRec {      // lrgr
        GainRec         grLeft;                     // Left Channel gain
        GainRec         grRight;                    // Right Channel gain
};
typedef struct LRGainRec *LRGainRecPtr, **LRGainRecHdl;
```

Please see section **5.5** for the definition and code samples in C for conversion of the gain data.

Dependency: SonarLines

3.50 SonarLines

Tag: 259 (103h)
Type: SHORT
N: 1
Default: 1000

Comments: Number of sonar lines stored in this data file. Number of vertical rows in the sonar data.

Dependency: None

3.51 SurveyPlotterImage

Tag: 271 (10Fh)
Type: BYTE
N: 300 x 200 bitmap

Comments: This field contains the bitmap of the Survey Plotter. This is only a bitmap of the Survey plotter. You must read the SurveyPlotterParms or SurveyPlotterParms2 field to determine the boundary coordinates.

Dependency: Compression

3.52 SurveyPlotterParms

Tag: 270 (10Eh)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the Plotter Parameters Record (ppr) is as follows:

```
struct PlotterParmsRec {        // ppr
                float           fLL_Lat;        // Lower left Latitude
                float           fLL_Long;       // Lower left Longitude
                float           fUR_Lat;        // Upper Right Latitude
                float           fUR_Long;       // Upper Right Longitude
                float           fRng_Lat;       // Latitude range
                float           fRng_Long;      // Longitude range
                char            strLL_Lat[15];  // Lower left Latitude string
                char            strLL_Long[15]; // Lower left Longitude string
                char            strUR_Lat[15];  // Upper right  Latitude string
                char            strUR_Long[15]; // Upper right  Longitude string
                BOOL            bDrawSwath;     // Draw swath flag
                BOOL            bDrawLines;     // Draw interconnecting lines flag
                float           fPrevLatitude;  // Previous Latitude
                float           fPrevLongitude; // Previous Longitude
                BOOL            bCrossDateLine; // Plotter parameters cross IDL
};
typedef struct PlotterParmsRec *PlotterParmsRecPtr, **PlotterParmsRecHdl;
```

Although you will only require the first four fields to define the Survey Plotter, all 15 fields of each ppr will be described.

1-2. Lower Left Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the lower left (southwest) corner of the plotter. These are the minimum latitude and longitude values for the plotter. The L/L are stored in a proprietary format. This format is described in detail later in this section.

3-4. Upper Right Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the upper right (northeast) corner of the plotter. These are the maximum L/L values for the plotter.

5-6. Latitude and Longitude Range [float]

These fields contain the range from the lower left (southwest) corner to the upper right (northeast) corner in latitude and longitude (L/L) values. These fields are not required to define the coordinates of the plotter.

7-10. Lat/Long strings [15 byte C string]

These fields contain the latitude and longitude strings for the lower left and upper right corners of the plotter. The individual strings, including the required '\0' terminator for a C string, cannot be more than 15 characters. These fields are not required to define the coordinates of the plotter.

11-12. Draw Swath and Lines [integer typedef as Boolean]

These fields contain Boolean flags to set the display parameters. If the respective values are non-zero the plotter will draw the swath lines for the sonar data on the plotter and the interconnecting lines between neighboring position fixes. These fields are not required to define the coordinates of the plotter.

13-14. Previous Lat/Long [float]

These are reserved fields that do not contain any information but are used when drawing the plotters. These fields are not required to define the coordinates of the plotter.

15. Cross Date Line [integer typedef as Boolean]

This field contains a Boolean flag to alert the plotter drawing routine that the plotter parameters cross the international date line. Thus, contrary to convention, the leftmost (west) longitude will be greater than the rightmost (east) longitude.

Dependency: None

3.53 SurveyPlotterParms2

Tag: 290 (122h)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the version 2 Plotter Parameters Record (ppr2) is as follows:

```
struct PlotterParmsRec2 {       // ppr2
                float           fLL_Lat;            // Lower left Latitude
                float           fLL_Long;           // Lower left Longitude
                float           fUR_Lat;            // Upper Right Latitude
                float           fUR_Long;           // Upper Right Longitude
                float           fRng_Lat;           // Latitude range
                float           fRng_Long;          // Longitude range
                char            strLL_Lat[15];      // Lower left Latitude string
                char            strLL_Long[15];     // Lower left Longitude string
                char            strUR_Lat[15];      // Upper right  Latitude string
                char            strUR_Long[15];     // Upper right  Longitude string
                BOOL            bDrawSwath;         // Draw swath flag
                BOOL            bDrawLines;         // Draw interconnecting lines flag
                BOOL            bDrawDepth;         // Draw depth flag
                float           fMaxWaterDepth;     // Maximum Water Depth
                float           fPrevLatitude;      // Previous Latitude
                float           fPrevLongitude;     // Previous Longitude
                BOOL            bCrossDateLine;     // Plotter parameters cross IDL
};
typedef struct PlotterParmsRec2 *PlotterParmsRec2Ptr, **PlotterParmsRec2Hdl;
```

Although you will only require the first four fields to define the Survey Plotter, all seventeen fields of each ppr2 will be described.

1-2. Lower Left Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the lower left (southwest) corner of the plotter. These are the minimum latitude and longitude values for the plotter. The L/L are stored in a proprietary format. This format is described in detail later in this section.

3-4. Upper Right Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the upper right (northeast) corner of the plotter. These are the maximum L/L values for the plotter.

5-6. Latitude and Longitude Range [float]

These fields contain the range from the lower left (southwest) corner to the upper right (northeast) corner in latitude and longitude (L/L) values. These fields are not required to define the coordinates of the plotter.

7-10. Lat/Long strings [15 byte C string]

These fields contain the latitude and longitude strings for the lower left and upper right corners of the plotter. The individual strings, including the required '\0' terminator for a C string, cannot be more than 15 characters. These fields are not required to define the coordinates of the plotter.

11-13. Draw Swath, Lines and Depth [integer typedef as Boolean]

These fields contain Boolean flags to set the display parameters. If the respective values are non-zero the plotter will draw the swath lines for the sonar data on the plotter, the interconnecting lines between neighboring position fixes and the depth using a standard depth scale. These fields are not required to define the coordinates of the plotter.

14. Maximum Water Depth [float]

This field contains the maximum water depth. This value is used to scale the depth display using the standard depth color scale.

15-16. Previous Lat/Long [float]

These are reserved fields that do not contain any information but are used when drawing the plotters. These fields are not required to define the coordinates of the plotter.

17. Cross Date Line [integer typedef as Boolean]

This field contains a Boolean flag to alert the plotter drawing routine that the plotter parameters cross the international date line. Thus, contrary to convention, the leftmost (west) longitude will be greater than the rightmost (east) longitude.

Dependency: None

3.54 SurveyPlotterParms3

Tag: 302 (12Eh)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the version 3 Plotter Parameters Record (ppr3) is as follows:

```
struct PlotterParmsRec3 {       // ppr3
                float           fLL_Lat;            // Lower left Latitude
                float           fLL_Long;           // Lower left Longitude
                float           fUR_Lat;            // Upper Right Latitude
                float           fUR_Long;           // Upper Right Longitude
                float           fRng_Lat;           // Latitude range
                float           fRng_Long;          // Longitude range
                char            strLL_Lat[15];      // Lower left Latitude string
                char            strLL_Long[15];     // Lower left Longitude string
                char            strUR_Lat[15];      // Upper right  Latitude string
                char            strUR_Long[15];     // Upper right  Longitude string
                BOOL            bDrawSwath;         // Draw swath flag
                BOOL            bDrawLines;         // Draw interconnecting lines flag
                BOOL            bDrawDepth;         // Draw depth flag
                float           fMaxWaterDepth;     // Maximum Water Depth
                float           fPrevLatitude;      // Previous Latitude
                float           fPrevLongitude;     // Previous Longitude
                BOOL            bCrossDateLine;     // Plotter parameters cross IDL
                BOOL            bDrawToPlotter;     // Draw info to Plotter flag
};
typedef struct PlotterParmsRec3 *PlotterParmsRec3Ptr, **PlotterParmsRec3Hdl;
```

Although you will only require the first four fields to define the Survey Plotter, all eighteen fields of each ppr3 will be described.

1-2. Lower Left Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the lower left (southwest) corner of the plotter. These are the minimum latitude and longitude values for the plotter. The L/L are stored in a proprietary format. This format is described in detail later in this section.

3-4. Upper Right Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the upper right (northeast) corner of the plotter. These are the maximum L/L values for the plotter.

5-6. Latitude and Longitude Range [float]

These fields contain the range from the lower left (southwest) corner to the upper right (northeast) corner in latitude and longitude (L/L) values. These fields are not required to define the coordinates of the plotter.

7-10. Lat/Long strings [15 byte C string]

These fields contain the latitude and longitude strings for the lower left and upper right corners of the plotter. The individual strings, including the required '\0' terminator for a C string,

cannot be more than 15 characters. These fields are not required to define the coordinates of the plotter.

11-13. Draw Swath, Lines and Depth [integer typedef as Boolean]

These fields contain Boolean flags to set the display parameters. If the respective values are non-zero the plotter will draw the swath lines for the sonar data on the plotter, the interconnecting lines between neighboring position fixes and the depth using a standard depth scale. These fields are not required to define the coordinates of the plotter.

14. Maximum Water Depth [float]

This field contains the maximum water depth. This value is used to scale the depth display using the standard depth color scale.

15-16. Previous Lat/Long [float]

These are reserved fields that do not contain any information but are used when drawing the plotters. These fields are not required to define the coordinates of the plotter.

17. Cross Date Line [integer typedef as Boolean]

This field contains a Boolean flag to alert the plotter drawing routine that the plotter parameters cross the international date line. Thus, contrary to convention, the leftmost (west) longitude will be greater than the rightmost (east) longitude.

18. Draw to Plotter [integer typedef as Boolean]

This field contains a Boolean flag to indicate the drawing persistency for the plotter. If the value is non-zero the navigational information (depending on the Draw Swath and Lines flags) will be drawn on the plotter. This field is used during the real-time data collection and is not required to define the coordinates of the plotter.

Dependency: None

3.55 SurveyPlotterParms4

Tag: 305 (131h)
Type: STRUCT
N: 1

Comments: The structure declaration in the C++ programming language for the version 4 Plotter Parameters Record (ppr4) is as follows:

```
struct PlotterParmsRec4 {       // ppr4
        float       fLL_Lat;            // Lower left Latitude
        float       fLL_Long;           // Lower left Longitude
        float       fUR_Lat;            // Upper Right Latitude
        float       fUR_Long;           // Upper Right Longitude
        float       fRng_Lat;           // Latitude range
        float       fRng_Long;          // Longitude range
        char        strLL_Lat[32];      // Lower left Latitude string
        char        strLL_Long[32];     // Lower left Longitude string
        char        strUR_Lat[32];      // Upper right  Latitude string
        char        strUR_Long[32];     // Upper right  Longitude string
        BOOL        bDrawSwath;         // Draw swath flag
        BOOL        bDrawLines;         // Draw interconnecting lines flag
        BOOL        bDrawDepth;         // Draw depth flag
        float       fMaxWaterDepth;     // Maximum Water Depth
        BOOL        bDrawCoast;         // Draw WVSFULL.DAT coastline
        BOOL        bDrawGrid;          // Draw Lat/Long grid lines
        float       fPrevLatitude;      // Previous Latitude
        float       fPrevLongitude;     // Previous Longitude
        BOOL        bCrossDateLine;     // Plotter parameters cross IDL
        BOOL        bDrawToPlotter;     // Draw info to Plotter flag
};
typedef struct PlotterParmsRec4 *PlotterParmsRec4Ptr, **PlotterParmsRec4Hdl;
```

Although you will only require the first four fields to define the Survey Plotter, all twenty fields are described as follows:

1-2. Lower Left Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the lower left (southwest) corner of the plotter. These are the minimum latitude and longitude values for the plotter. The L/L are stored in a proprietary format. This format is described in detail later in this section.

3-4. Upper Right Latitude and Longitude [float]

These fields contain the latitude and longitude (L/L) of the upper right (northeast) corner of the plotter. These are the maximum L/L values for the plotter.

5-6. Latitude and Longitude Range [float]

These fields contain the range from the lower left (southwest) corner to the upper right (northeast) corner in latitude and longitude (L/L) values. These fields are not required to define the coordinates of the plotter.

7-10. Lat/Long strings [32 byte C string]

These fields contain the latitude and longitude strings for the lower left and upper right corners of the plotter. The individual strings, including the required '\0' terminator for a C string,

cannot be more than 32 characters. These fields are not required to define the coordinates of the plotter.

11-13. Draw Swath, Lines and Depth [integer typedef as Boolean]

These fields contain Boolean flags to set the display parameters. If the respective values are non-zero the plotter will draw the swath lines for the sonar data on the plotter, the interconnecting lines between neighboring position fixes and the depth using a standard depth scale. These fields are not required to define the coordinates of the plotter.

14. Maximum Water Depth [float]

This field contains the maximum water depth. This value is used to scale the depth display using the standard depth color scale.

15-16. Draw Coast and Grid [integer typedef as Boolean]

These fields contain Boolean flags to set the background display parameters. If the respective values are non-zero the plotter will draw the coastline from the WVSFULL.DAT coastline file and the Lat/Long grid lines on the plotter. These fields are not required to define the coordinates of the plotter.

17-18. Previous Lat/Long [float]

These are reserved fields that do not contain any information but are used when drawing the plotters. These fields are not required to define the coordinates of the plotter.

19. Cross Date Line [integer typedef as Boolean]

This field contains a Boolean flag to alert the plotter drawing routine that the plotter parameters cross the international date line. Thus, contrary to convention, the leftmost (west) longitude will be greater than the rightmost (east) longitude.

20. Draw to Plotter [integer typedef as Boolean]

This field contains a Boolean flag to indicate the drawing persistency for the plotter. If the value is non-zero the navigational information (depending on the Draw Swath and Lines flags) will be drawn on the plotter. This field is used during the real-time data collection and is not required to define the coordinates of the plotter.

Dependency: None

3.56 TimeCorrelation

Tag: 262 (106h)
Type: STRUCT
N: 1

Comments: The Sea Scan PC applications make extensive use of the system time as a timestamp to maintain relative timing between the incoming sonar data lines and the navigational information. The system time is the time elapsed in milliseconds since Windows was last started. You can correlate the system time with local time by looking at the TimeCorrelation field. The system time and the Local time (as a Borland C++ tm structure) were recorded at the same time (within a millisecond). This data will allow you to determine a local time for any system time in the active data file.

This field contains the time correlation structure. The structure declaration in the C programming language for the TimeCorrelation Record (tcr) is as follows:

```
struct TimeCorrelationRec {      // tcr
                DWORD       dwSystime;      // system time
                struct tm   tmTime;         // Borland tm structure for time
};
typedef struct TimeCorrelationRec * TimeCorrelationRecPtr, ** TimeCorrelationRecHdl;
```

Each tcr contains two fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp, which is the time elapsed in milliseconds since Windows was last started.

2. Borland C++ tm structure [128 byte C string]

   This field contains the Borland C++ tm structure that was recorded at the same time as the Timestamp.

   The Borland C++ tm structure is defined as follows:

```
struct tm {
        short   tm_sec;         // Seconds
        short   tm_min;         // Minutes
        short   tm_hour;        // Hour (0--23)
        short   tm_mday;        // Day of month (1--31)
        short   tm_mon;         // Month (0--11)
        short   tm_year;        // Year (calendar year minus 1900)
        short   tm_wday;        // Weekday (0--6; Sunday = 0)
        short   tm_yday;        // Day of year (0--365)
        short   tm_isdst;       // 0 if daylight savings time is not in effect)
};
```

Dependency: None

3.57 TVGType

Tag: 311 (137h)
Type: LONG
N: 1

Comments: The Sea Scan PC applications stores type of TVG hardware that it is controlling in the MSTIFF file so the TVG curve can accurately be reconstructed. The type of TVG hardware is encoded as a bit. The following bits are currently being used:

TVGTYPE_NORMAL = 0x00000000
TVGTYPE_HIGHRES = 0x00000001
TVGTYPE_HIGHSPEED = 0x00000002

These bits can be used in any combination. For example, Sea Scan PC hardware that has both High Resolution TVG and High Speed TVG will be stored as 0x00000003.

Please see section **5.5** for the definition and code samples in C for conversion of the TVG data.

Dependency: None

3.58 Y2KTimeCorrelation

Tag: 285 (11Dh)
Type: STRUCT
N: 1

Comments: The Sea Scan PC applications make extensive use of the system time as a timestamp to maintain relative timing between the incoming sonar data lines and the navigational information. The system time is the time elapsed in milliseconds since Windows was last started. You can correlate the system time with federal government Year 2000 compliant date/ time by looking at the Y2KTimeCorrelation field. The system time and the Year 2000 compliant time were recorded at the same time (within a millisecond). This data will allow you to determine a valid Year 2000 compliant time for any system time in the active data file.

This field contains the Year 2000 time correlation structure. The structure declaration in the C programming language for the Y2KTimeCorrelation Record (y2ktcr) is as follows:

```
struct Y2KTimeCorrelationRec {// y2ktcr
                DWORD          dwSystime;          // system time
                unsigned long  ulDate;             // Y2K Date
                unsigned long  ulTime;             // Y2K Time
};
typedef struct Y2KTimeCorrelationRec *Y2KTimeCorrelationRecPtr, **Y2KTimeCorrelationRecHdl;
```

Each y2ktcr contains three fields.

1. Timestamp [long word typedef as DWORD]

   This field contains the timestamp, which is the time elapsed in milliseconds since Windows was last started.

2. Year 2000 compliant date [unsigned long]

   This field contains the federal government Y2K-compliant date that was recorded at the same time as the Timestamp. The format of the Y2K Date is described in detail later in this section.

3. Year 2000 compliant time [unsigned long]

   This field contains the federal government Y2K-compliant time that was recorded at the same time as the Timestamp. The format of the Y2K Time is described in detail later in this section.

Dependency: None

4 Fields Sorted by Number

4.1 Current Fields

Field	*Decimal*	*Hex*	*Type*	*Number of Values*
Compression	254	FE	SHORT	1
CondensedImage	255	FF	BYTE	
Description	256	100	ASCII	
History	257	101	ASCII	
BitsPerBin	258	102	SHORT	1
SonarLines	259	103	SHORT	1
BinsPerChannel	260	104	SHORT	1
ScrollDirection	261	105	SHORT	1
TimeCorrelation	262	106	STRUCT	1
NavInfoCount	266	10A	SHORT	1
SurveyPlotterImage	271	10F	BYTE	300 x 200
PingNavInfo	274	112	STRUCT	SonarLines
Annotation3Count	280	118	SHORT	1
Annotation3	281	119	STRUCT	Annotation3Count
Y2KtimeCorrelation	285	11D	STRUCT	1
FathometerCount	286	11E	SHORT	1
MagnetometerCount	288	120	SHORT	1
Magnetometer	289	121	STRUCT	MagnetometerCount
Fathometer2	296	128	STRUCT	FathometerCount
SonarDataInfo3	298	12A	STRUCT	SonarLines
LeftChannel2	299	12B	BYTE	SonarLines x BinsPerChannel
RightChannel2	300	12C	BYTE	SonarLines x BinsPerChannel
CreatorVersion	301	12D	STRUCT	1
MagnetometerParms2	303	12F	STRUCT	1
NavInterpolationTimeout	304	130	LONG	1
SurveyPlotterParms4	305	131	STRUCT	1
Marker5Count	306	132	SHORT	1
Marker5	307	133	STRUCT	Marker5Count
NavInfo6	308	134	STRUCT	NavInfoCount
TVGType	311	137	STRUCT	1

4.2 Out of Date Fields

Field	*Decimal*	*Hex*	*Type*	*Number of Values*
LeftChannel	263	107	BYTE	SonarLines x BinsPerChannel
RightChannel	264	108	BYTE	SonarLines x BinsPerChannel
SonarDataInfo	265	109	STRUCT	SonarLines
NavInfo	267	10B	STRUCT	NavInfoCount
MarkerCount	268	10C	SHORT	1
Marker	269	10D	STRUCT	MarkerCount
SurveyPlotterParms	270	10E	STRUCT	1
AnnotationCount	272	110	SHORT	1
Annotation	273	111	STRUCT	AnnotationCount
NavInfo2	275	113	STRUCT	NavInfoCount
Marker2Count	276	114	SHORT	1
Marker2	277	115	STRUCT	Marker2Count
Annotation2Count	278	116	SHORT	1
Annotation2	279	117	STRUCT	Annotation2Count
NavInfo3	282	11A	STRUCT	NavInfoCount
Marker3Count	283	11B	SHORT	1
Marker3	284	11C	STRUCT	Marker3Count
Fathometer	287	11F	STRUCT	FathometerCount
SurveyPlotterParms2	290	122	STRUCT	1
MagnetometerParms	291	123	STRUCT	1
SonarDataInfo2	292	124	STRUCT	SonarLines
NavInfo4	293	125	STRUCT	NavInfoCount
Marker4Count	294	126	SHORT	1
Marker4	295	127	STRUCT	Marker4Count
NavInfo5	297	129	STRUCT	NavInfoCount
SurveyPlotterParms3	302	12E	STRUCT	1

5 Definitions

5.1 Julian Date and Time

The Julian Date is expressed as a five digit unsigned long. The first two digits (ten thousands and thousands) are the last two digits of the year (i.e. 1998 becomes 98xxx). The last three digits are the day of the year using a 1-based notation (i.e. February 1 is xx032). To convert the date to Julian Date:

JulianDate= (Year – 1900) * 1000 + DayOfYear

i.e. February 1, 1998 ⇒ Julian Date = (1998 – 1900) * 1000 + 32
= 98 * 1000 + 32
= 98032

The Julian Time is expressed as an unsigned long. The time is the number of seconds that have past since midnight. To convert the time to Julian Time:

JulianTime= (((Hour * 60) + Minutes) * 60) + Seconds

i.e. 2:32:45 p.m. ⇒ Julian Time = (((14 * 60) + 32) * 60) + 45
= 52365

5.2 Year 2000 Compliant Date

The U.S. Federal Government has mandated a Year 2000 (Y2K) compliance criterion. This stipulates that all dates must use a YYYYMMDD format, with no exceptions. The Y2K Date is expressed as an eight digit unsigned long. The first four digits are the four digit year (i.e. 1998 becomes 1998xxxx). The next two digits are the month of the year using a 1-based notation (i.e. February is xxxx02xx. The last two digits are the day of the month using a 1-based notation (i.e. the 7th is xxxxxx07. To convert the date to Y2K Date:

Y2KDate= Year * 10000 + Month * 100 + DayOfMonth

i.e. February 1, 1998 ⇒ Y2K Date = 1998 * 10000 + 2 * 100 + 1
= 19980000 + 200 + 1
= 19980201

5.3 Latitude and Longitude Values

The latitude and longitude (L/L) values are stored as floats. The degrees and decimal minute values have been converted into decimal minutes. For example, 1° is 60.0 decimal minutes. The decimal minutes are made positive for the North and East hemispheres and negative for the South and West hemispheres.

To convert the float value to degrees and decimal minutes for Latitude:

$$LatDegrees = MSTLat / 60.0$$
$$LatMinutes = MSTLat - (LatDegrees * 60.0)$$
$$\text{if } MSTLat \geq 0 \Rightarrow \text{N Hemisphere}$$
$$\text{otherwise} \Rightarrow \text{S Hemisphere}$$

To convert the float value to degrees and decimal minutes for Longitude:

$$LongDegrees = MSTLong / 60.0$$
$$LongMinutes = MSTLong - (LongDegrees * 60.0)$$
$$\text{if } MSTLong \geq 0 \Rightarrow \text{E Hemisphere}$$
$$\text{otherwise} \Rightarrow \text{W Hemisphere}$$

5.4 Range Code

The range code defines the channel and range with which the sonar data was collected. This code is divided into two sections:

Channel Mode

The channel mode designates which of the sonar channels were selected for the sonar data line. The operator may chose to look at Both channels, the Left channel only at double resolution or the Right channel only also at double resolution.

The seventh and eighth bits of the 8-bit range code determine the channel mode. The channel mode bits may be separated from the other components of the range code by masking the range code with the hexadecimal value $C0.

$$ChannelMode = RangeCode \wedge \$C0$$

Thus the channel mode is defined as:

Channel Mode Bits	Channel Mode
01xx xxxx	Left Channel Only
10xx xxxx	Right Channel Only
00xx xxxx or 11xx xxxx	Both Channels

The data is stored in slightly different ways depending on the channel mode.

Both Channels

The data from the left channel is stored sequentially in the left channel buffer. For example, the n^{th} data bin from the Interface board for the left channel is stored as the n^{th} element in the left channel array for the current sonar line. Similarly, the n^{th} data bin from the Interface board for the right channel is stored as the n^{th} element in the right channel array for the current sonar line. There are *BinsPerChannel* data bins for each channel stored for each sonar data line, thus *2*BinsPerChannel* data bins in total for each sonar data line.

Left Channel Only

The double resolution data from the left channel is stored sequentially in both the left and right channel buffers. For example, the $(2*n)^{th}$ data bin from the Interface board for the left channel is stored as the n^{th} element in the left channel data storage array for the current sonar line. Then, the $(2*n+1)^{th}$ data bin from the Interface board for the left channel is stored as the n^{th} element in the right channel data storage array for the current sonar line. There are *2*BinsPerChannel* data bins for the left channel stored for each sonar data line.

Right Channel Only

Similar to the Left Channel Only mode, the double resolution data from the right channel is stored sequentially in both the right and left channel buffers. For example, the $(2*n)^{th}$ data bin from the Interface board for the right channel is stored as the n^{th} element in the right channel data storage array for the current sonar line. Then, the $(2*n+1)^{th}$ data bin from the Interface board for the right channel is stored as the n^{th} element in the left channel data storage array for the current sonar line. There are *2*BinsPerChannel* data bins for the right channel stored for each sonar data line.

Range Mode

The range mode designates the range of the two sonar channels. The range is totally independent of the channel mode.

The first four bits of the 8-bit range code determine the range mode. The range mode bits may be separated from the other components of the range code by masking the range code with the hexadecimal value $0F.

$$RangeMode = RangeCode \wedge \$0F$$

Thus the range mode is defined as:

Range Mode Bits	Range Mode Decimal	Range [meters]
0001	1	5
0010	2	10
0011	3	20
0100	4	50
0101	5	75
0110	6	100
0111	7	150
1000	8	200
1001	9	300
1010	10	500
1011	11	30
1100	12	40

5.5 Time Varying Gain

Time varying gain is controlled in the Sea Scan PC system via a combination of software and hardware. A total of 8 gain control sliders are provided for each channel, one for the right channel and one for the left channel. Each of these sliders controls a voltage ramp that is sent to an amplifier that ultimately increases its gain over time. The table below is a list (in compiler #define statements) of Normal Gain ranges.

#define RANGE_GAINPT0	2.0
#define RANGE_GAINPT1	4.0
#define RANGE_GAINPT2	8.0
#define RANGE_GAINPT3	16.0
#define RANGE_GAINPT4	32.0
#define RANGE_GAINPT5	64.0
#define RANGE_GAINPT6	128.0
#define RANGE_GAINPT7	1024.0

Each of these gain points represents the range at which a particular slider has control. Each slider can never be less than the previous slider. There can never be a decrease in the gain ramp. When the gain hardware is setup in High Speed mode the ranges are decreased by a factor of 4. The following ranges are the result:

#define RANGE_GAINPT0	0.5
#define RANGE_GAINPT1	1.0
#define RANGE_GAINPT2	2.0
#define RANGE_GAINPT3	4.0
#define RANGE_GAINPT4	8.0
#define RANGE_GAINPT5	16.0
#define RANGE_GAINPT6	32.0
#define RANGE_GAINPT6	256.0

For the Normal TVG and High Speed hardware each slider position is converted into a slope value that describes the ramp that the slider controls. There is one exception. The first slider (slider 0) does not control a ramp it actually sets the initial value of where the gain curve start.

For example:

To convert the slider 3 from its absolute position to slope form for the Normal TVG hardware one would use this equation:

$$Slope_3 = (Absolute_3 – Absolute_2) / (RANGE\_GAINPT_3 – RANGEGAINPT_2)$$

Where $Slope_3$ is the resultant slope value, $Absolute_3$ is the absolute position value of slider 3, and $Absolute_2$ is the absolute position value of slider 2.

For the High Resolution TVG hardware each slider position is not converted into slope value. The hardware does the slope conversion, the difference between the previous slider and the next slider is used instead.

For example:

To convert the slider 3 from its absolute position to a difference value for the High Resolution TVG hardware one would use this equation:

$$Difference_3 = (Absolute_3 – Absolute_2)$$

Where $Difference_3$ is the resultant difference value, $Absolute_3$ is the absolute position value of slider 3, and $Absolute_2$ is the absolute position value of slider 2.