Navico (Lowrance, Simrad, B&G)
Sonar Log File Format

Herbert Oppmann

herby@memotech.franken.de

http://www.memotech.franken.de/FileFormats/

2018-01-01

Content

Navico Sonar Log File Format

Filename extensions *.slg (format 1, sonar only), *.sl2 (format 2, sonar and structure) and *.sl3 (format 3, includes ForwardScan)

This documentation is based on own research and the sources listed in the references section.

Text highlighted like this is where further investigation is needed.

Basic data types

All values are serialized in little-endian byte order (least significant byte first).

Type	Length	Description
byte	1	8 bit unsigned integer (range 0 .. 255)
ushort	2	16 bit unsigned integer (range 0 .. 65535)
uint	4	32 bit unsigned integer (range 0 .. 4294967295)
short	2	16 bit signed integer (range -32768 .. 32767)
int	4	32 bit signed integer (range -2147483648 .. 2147483647)
float	4	32 bit floating point value according to [8].

Mercator meter to WGS84 conversion (in C#):

```
private const double PolarEarthRadius = 6356752.3142;
private const double RadiansToDegrees = 180.0 / Math.PI;

public static double Longitude(int Easting)
{
   return (Easting / PolarEarthRadius) * RadiansToDegrees;
}

public static double Latitude(int Northing)
{
    return ((2.0 * Math.Atan(Math.Exp(Northing / PolarEarthRadius))) - (Math.PI / 2.0)) * RadiansToDegrees;
 }
```

General file structure

Header
List of Frames (no gaps) The last frame sometimes has incomplete sounding / bounce data.

Header

Offset	Type	Content
+0	ushort	Format 1 = *.slg, 2 = *.sl2, 3 = *.sl3
+2	ushort	Version 0 = ex HDS 7, 1 = ex Elite 4 CHIP
+4	ushort	BlockSize, see table below.
+6	byte	Debug. 0= off, <>0 on If set, enables the ChannelTypes Debug Digital and Debug Noise.

+7	byte	? =0
Only if Format =0:		
+8	ushort	Unknown2, =0

BlockSize seen values:

Value	Meaning
0x0064 (100)	?
0x07B2 (1920)	Downscan
0x0992 (2450)	? Only in Format 1
0x0C80 (3200)	Sidescan

Frame for Format 1

See source code embedded in [6] and file addon\src\main\java\org\vaadin\sonarwidget\data\LowranceSonar.java in [7].

The header has variable size. The fields marked with “cond.” are only present when the Flags indicate “valid”.

Offset	Type	Content
+0	ushort	Flags, see table below
cond.	float	LowerLimit Lower depth limit of the transducer in use
cond.	float	WaterDepth in feet
cond.	float	Water temperature in °Celsius
cond.	float	WaterSpeed Speed through the water from the paddlewheel speed sensor. Actual water speed (or GPS if sensor not present) in knots.
cond.	uint	Easting in mercator meters
cond.	uint	Northing in mercator meters
cond.	float	Surface depth in feet
cond.	float	TopOfBottom depth in feet
cond.	float	Temperature 2 in °Celsius
cond.	float	Temperature 3 in °Celsius
cond.	uint	TimeOffset in ms from the start of the log
cond.	float	Speed (GPS) Speed from GPS in knots.
cond.	float	Track in radians
cond.	float	Altitude in feet above sea level
	ushort	PacketSize. Seen value: 2400
	byte[n]	Sounding / bounce data (no specific structure, just a list of values) A little bit shorter than PacketSize. Filled with 0x00 bytes until BlockSize bytes are full.

Flag bits for Format 1:

Bit offset / mask	Meaning if set
0 0x0001	Altitude invalid (if TimeOffset valid and/or Speed and track valid)
1 0x0002	not seen
2 0x0004	? used
3 0x0008	not seen

Bit offset / mask	Meaning if set
4 0x0010	Water temperature valid
5 0x0020	Temp2 valid
6 0x0040	Temp3 valid
7 0x0080	Water speed valid
8 0x0100	Position valid
9 0x0200	Depth **in**valid
10 0x0400	Surface depth valid
11 0x0800	TopOfBottom depth valid
12 0x1000	ColumnIs50kHz
13 0x2000	TimeOffset valid
14 0x4000	Speed and track valid
15 0x8000	unused

Altitude valid is a bit strange: When TimeOffset valid and / or Speed and track valid is set, Altitude is valid, unless bit 0 is set, in which case it is invalid?!

Frame for Format 2

The size of the frame header is 144 bytes.

General info on "Last *type* channel frame offset in file": The current frame already counts, i.e. if the current frame is of *type*, the value is the same as "Frame offset in file". If there was no *type* frame yet, the value is 0.

Offset	Type	Content
+0	uint	Frame offset in file In first frame = 8.
+4	uint	Last primary channel frame offset in file
+8	uint	Last secondary channel frame offset in file
+12	uint	Last downscan channel frame offset in file
+16	uint	Last left sidescan channel frame offset in file
+20	uint	Last right sidescan channel frame offset in file
+24	uint	Last composite sidescan channel frame offset in file
+28	ushort	FrameSize This is the actual size of the frame in bytes. It is 94 or 144 more than the BlockSize given in the header(?)
+30	ushort	PreviousFrameSize Size of the previous frame (FrameIndex-1) in bytes. In first frame =0.
+32	ushort	ChannelType, see table below.
+34	ushort	PacketSize Size of the sounding/bounce data in bytes. It should be 144 less than FrameSize.
+36	uint	FrameIndex Starts with 0 and counts up.
+40	float	UpperLimit in feet Upper depth limit of the transducer in use
+44	float	LowerLimit in feet Lower depth limit of the transducer in use
+48	ushort	Seen values: 0, 12, 20, 21, 27, 50, 130, 520, 4956, 5000, 5050, 5100, 5150, 5200, ...
+50	byte	Seen values: 0, 4, 5, 8, 13, 25, 32, 40, 48, 56, 64, ...
+51	byte	Seen values: 0, 19, 41, 43, 46, 47, 48, 49, 50, 79, 80, ...

+52	byte	Seen value: 0, 1
+53	byte	Frequency, see table below.
+54	ushort	Seen values: 0, 41, 48, 49, 51, 53, 100, 767, 793, 795, 806, 854, ...
+56	uint	Values seen: 0, 1, 8, 160, 232, 257, 513, 514 Flags or separate bytes
+60	int	CreationDateTime. A value of -1 means not set. Otherwise it is a POSIX time, see [9].
+64	float	WaterDepth in feet
+68	float	KeelDepth in feet
+72	byte	Seen values: 0x02, 0x13, 0x29, 0x33, 0x36, 0x39, 0x3F, 0x46, 0x47, 0x48, 0x49, 0x4C, 0x62, 0x64 or 0x66
+73	byte	Seen values: 0x00, 0xBD, or 0xC0
+74	ushort	Seen values: 0x0000, 0x1305, 0x2805, 0x6205, 0x6505
+76	ushort	Seen values: 0, 0x0100, 0x1E13, 0x1E28, 0x1E62 or 0x1E65
+78	ushort	Seen values: 0, 1, 1000
+80	float	Seen value: -1.0, ..
+84	float	Seen value: -1.0, ..
+88	float	Seen value: -1.0, ..
+92	float	Seen value: -1.0
+96	byte	Seen values: 0, 1, 2
+97	byte	Seen values: 0, 1, 2
+98	byte	Seen values: 0, 1
+99	byte	Seen values: 0, 1
+100	float	Speed (GPS) Speed from GPS in knots.
+104	float	Water temperature in °Celsius
+108	int	Easting in mercator meters
+112	int	Northing in mercator meters
+116	float	WaterSpeed Speed through the water from the paddlewheel speed sensor. Actual water speed (or GPS if sensor not present) in knots.
+120	float	Track (course over ground) in radians
+124	float	Altitude in feet above sea level
+128	float	Heading in radians
+132	ushort	Flags, see table below
+134	ushort	Seen value: 0
+136	byte	Seen values: 1, 3
+137	byte	Seen values: 0, 1, 4, and 6
+138	ushort	Seen values: 0x000A, 0x010A, 0x010F, 0x0114, 0x0214 Could be two separate bytes. Note the low byte is multiple of 5 (10,15,20).
+140	uint	TimeOffset in ms from the start of the log
+144	PacketSize x byte	Sounding / bounce data (no specific structure, just a list of values)

Flag bits for Format 2:

Bit offset / mask	Meaning if set
0 0x0001	Unknown, not seen
1 0x0002	GPS speed valid
2 0x0004	Water temperature valid
3 0x0008	Unknown, but seen
4 0x0010	Position valid

Bit offset / mask	Meaning if set
5 0x0020	Unknown, but seen
6 0x0040	Water speed valid
7 0x0080	Track valid
8 0x0100	Heading valid
9 0x0200	Altitude valid
10-15	Unknown, not seen

Depth is always valid.

Frame for Format 3

The size of the frame header is 168 bytes.

Offset	Type	Content
+0	uint	Frame offset in file In first frame = 8.
+4	uint	Seen values: 10
+8	ushort	FrameSize This is the actual size of the frame in bytes.
+10	ushort	PreviousFrameSize Size of the previous frame (FrameIndex-1) in bytes. In first frame =0.
+12	ushort	ChannelType, see table below.
+14	ushort	Seen values: 0
+16	uint	FrameIndex Starts with 0 and counts up.
+20	float	UpperLimit in feet
+24	float	LowerLimit in feet
+28	...	Seen bytes: 00 00 05 00 00 00 00 00 30 00 00 00
+40	uint	CreationDateTime. A value of -1 means not set. Otherwise it is a POSIX time, see [9].
+44	ushort	PacketSize Size of the sounding/bounce data in bytes. It should be 168 less than FrameSize.
+46	ushort	Seen values: 0
+48	float	WaterDepth in feet
+52	byte	Frequency, see table below.
+53	...	Seen bytes: 00 00 00 44 B8 0F F8 E5 1E E8 03 The first four could be float 512.0
+64	float	Seen values: -1.0, 21.74922, 22.45391, ...
+68	float	Seen values: -1.0, 21.62109, ...
+72	float	Seen value: -1.0, 21.0125, ...
+76	float	Seen value: -1.0
+80	uint	Seen value: 0, 1, 0x00000101
+84	float	SpeedGPS Speed from GPS in knots.
+88	float	Water temperature in °Celsius
+92	int	Easting in mercator meters
+96	int	Northing in mercator meters
+100	uint	Seen value: 0
+104	float	Track (course over ground) in radians
+108	float	Altitude in feet above sea level
+112	float	Heading in radians
+116	uint	Seen values: 0x000003B8 (952)

+120	byte	Seen value: 1
+121	byte	Seen values: 0, 1, 4, and 6 (maybe 0=Depth invalid?)
+122	ushort	Seen values: 0x010A, 0x0114, 0x0214 Could be two separate bytes. Note the low byte is multiple of 5 (10,20).
+124	uint	TimeOffset in ms
+128	uint	Last primary channel frame offset in file
+132	uint	Last secondary channel frame offset in file
+136	uint	Last downscan channel frame offset in file
+140	uint	Last left sidescan channel frame offset in file
+144	uint	Last right sidescan channel frame offset in file
+148	uint	Last composite sidescan channel frame offset in file
+152	3 x uint	Seen values: 0
+164	uint	Last 3D scan channel frame offset in file
+168		

Sounding / bounce data if ChannelType = 9 (3D):

Offset	Type	Content
+0	uint	Size of header in byte = 76
+4	uint	S1 = Size of Section 1 in byte
+8	uint	S2 = Size of Section 2 in byte
+12	uint	S3 =Size of Section 3 in byte = S3a + S3b
+16	uint	S3a = Size of Subsection 3a in byte
+20	uint	S3b = Size of Subsection 3b in byte
+24	uint[7]	=0
+52	float	?
+56	uint[5]	=0
+76	byte[S1]	Section 1 (see below)
+76 + S1	byte[S2]	Section 2 (see below)
+76 + S1 + S2	byte[S3a]	Section 3a (see below)
+76 + S1 + S2 + S3a	byte[S3b]	Section 3b (see below)
+76 + S1 + S2 + S3a + S3b	0-3 byte	Alignment to 4-byte boundary, =0

The sum

- Size of Frame header (168) +
- Size of Sounding data header (76) +
- Size of Section 1 +
- Size of Section 2 +
- Size of Section 3 +

- alignment bytes

must be the FrameSize.

Section 1 and 2

Offset	Type	Content
(S1 or S2)/8 times pair of float:		
	float	Integer number, ascending (but not continuous), starting with 0
	float	Value

Section 3a and 3b

Offset	Type	Content
	uint	? The first two uints and the next two uints are similar. Coordinates?
	uint	?
	uint	?
	uint	?
	uint[(S3a or S3b-16-5) / 4]	? uint or float?
	4 byte	?
	1 byte	? Seen value: 2

ChannelType values:

Value	Meaning
0	Primary (Traditional Sonar)
1	Secondary (Traditional Sonar)
2	DSI (Downscan imaging)
3	Left (Sidescan)
4	Right (Sidescan)
5	Composite (Sidescan)
9	3D (only in Format 3)
10	Debug Digital (only if Debug on)
11	Debug Noise (only if Debug on)
other	invalid

Frequency values:

Value	Meaning
0	200 kHz
1	50 kHz
2	83 kHz
3	455 kHz
4	800 kHz
5	38 kHz
6	28 kHz
7	130-210 kHz
8	90-150 kHz
9	40-60 kHz

Value	Meaning
10	25-45 kHz
other	treated as 200 kHz

References

Used sources of information

[1] OpenStreetMap Wiki http://wiki.openstreetmap.org/wiki/SL2

[2] See Sea - Java Libraries for Maritime Navigation http://sourceforge.net/projects/seesea/

[3] Module to read .sl2 files https://github.com/kmpm/node-sl2format

[4] SL2 library for Go https://github.com/delitrem/sonarlog

[5] Small C# API for working with sonar log files https://github.com/risty/SonarLogApi

[6] Geotech forum thread “Lowrance MCC saved data structure” http://www.geotech1.com/forums/showthread.php?11159-Lowrance-MCC-saved-data-structure

[7] Vaadin add-on for viewing sonar logs https://github.com/capeisti/SonarWidget

Standards and specifications

[8] ISO/IEC/IEEE 60559:2011 Information technology – Microprocessor Systems – Floating-Point arithmetic (which is the successor of IEEE Std 754-2008)

[9] https://en.wikipedia.org/wiki/Unix_time

Sources of sample files

[10] http://www.lochnessinvestigation.com/castlecruisessonar.htm

[11] http://www.lowrance.com/de-DE/Software-Updates/ within firmware update ZIP files as samples

[12] ftp://software.lowrance.com/Simulators/Lowrance-Inland-Simulator.sl3