DEVIL XBT

and XSV

Data Formats

TURO TECHNOLOGY PTY LTD

Turo XBT and XSV Data Formats

Disclaimer

Although Turo Technology Pty Ltd (Turo) has taken all care in preparing this document, Turo makes no explicit or implied warranty with regards to the information contained herein and will not be liable for any damage or claim arising out of the information.

www.turo.com.au

Revision History

Turo XBT and XSV Data Formats

Date	Revision	Description
27 May 2014	R16	1. The beginning of the "time" variable is clarified in Explanatory Notes on Parameters
26 May 2014	R15	1. Explanatory Notes on Parameters describes "time 2. New exported meta data file "*.met".
10 Sep 2013	R14	3. Added meta data file information.).
3 Oct 2011	R13	4. Expanded information in netCDF description 5. New file, CALC (sound speed).
20 Apr 2009	R12	1. Table 3.5.2 CSIRO Format foot note, number of bytes for ship call sign now 9. 2. Section 3.5 Transmitted Data Format (TxData) explanation of maximum number of Temperature Depth points.
8 Feb 2009	R11	1. Updated netCDF format.
4 Sept 2008	R10	1. Iridium header byte definition
28 June 2008	R9	1. Corrected CSIRO Format call sign bytes.
25 June 2008	R8	1. Iridium SBD packet length changed.
16 June 2008	R7	1. ASCII mode in "Packing Iridium Transmissions
27 Apr 2008	R6	1. ASCII mode in "Packing Iridium Transmissions
5 Mar 2008	R5	1.
22 Feb 2008	R4	1. Amended JJVV file description and added .xbt format and added CSIRO transmission format
17 Jan 2008	R3	1. Added reassembling and TxData (transmit message) format.
31 May 2007	R3	1. Added definitions for columns of ascii formatted data.
28 May 2007	R2	1. Added links to netCDF and WOCE version 3.0 format.
17 May 2007	R1	1. Original document

CONTENTS

1 Devil Files and Formats

A number of different files are created. The files and formats used are:

- netCDF file
- 2 types of ascii text file, “CSIRO” format and csv format
- an ascii text file containing meta data
- an ascii text file of a hexadecimal representation of a JJVV bathymessage
- an ascii text file in “Sage XBT” format
- 2 type of packaging formats for satellite transmission, depending on the transmitter being used
- an ascii text file for sound speed in “CALC” format

The underlying data is stored in netCDF format.

Data from every XBT drop and test drop is saved in netCDF format and JJVV formats. Data for every XSV drop and XBT drop where calculated sound speed is enabled, is saved in netCDF and CALC formats.

Depending on the installation MODE and the configuration setup, other of the files listed above may also be automatically generated.

2 Files

2.1 Explanatory Notes on Parameters

2.1.1 Time

The time that is recorded against each sample may be either the *sampletime* or the *processtime* :

1) *sampletime* - The time, in seconds, that the sample was acquired, beginning at 0.00 for the first sample. The first sample is typically taken at 1 sample period after hitting the surface. This *sampletime* is determined by the sample interval, set by the hardware and is exactly 0.1 seconds.

2) *processtime* - The time, in seconds, that the computer got around to processing that sample. The time begins at 0.00 for the first sample to be processed. This *processtime* will be at approximately the sample rate with perhaps the first 10 samples having the same time, and occasionally 2 or more samples may be processed at the same time. This *processtime* should NOT be confused with the sample acquisition times which are exactly at 0.1 second intervals.

2.2 netCDF File

The netCDF file follows the WOCE version 3.0 format as described in:

http://woce.nodc.noaa.gov/wdiu/utils/netcdf/woce_conventions/woce_netcdf_format.htm.

A complete description of netCDF files can be found at

http://www.unidata.ucar.edu/software/netcdf.

Tools for reading these files are summarised in:

http://www.bodc.ac.uk/data/online_delivery/international_sea_level/woce_netcdf.html.

This data format has many advantages, the most important of which is that it is self-describing. This means that software packages can directly read the data and determine their structure, the variables' names and essential metadata such as the units.

The following is an example of a file describing the format:

```
netcdf drop019 {
dimensions:
        time = 1 ;
        depth = 1577 ;
        latitude = 1 ;
        longitude = 1 ;
variables:
        int time(time) ;
                time:long_name = "time" ;
                time:units = "seconds since 2008-01-01 00:00:00" ;
                time:data_min = 14083562 ;
                time:data_max = 14083562 ;
```

```
                time:_FillValue = -1 ;
        int woce_date(time) ;
                woce_date:long_name = "WOCE date" ;
                woce_date:units = "yyyymmdd UTC" ;
                woce_date:data_min = 20080612 ;
                woce_date:data_max = 20080612 ;
                woce_date:_FillValue = -1 ;
        int woce_time(time) ;
                woce_time:long_name = "WOCE time" ;
                woce_time:units = "hhmmss UTC" ;
                woce_time:data_min = 602 ;
                woce_time:data_max = 602 ;
                woce_time:_FillValue = -1 ;
        float depth(depth) ;
                depth:long_name = "depth" ;
                depth:units = "meters" ;
                depth:positive = "up" ;
                depth:data_min = 0.67f ;
                depth:data_max = 999.21f ;
                depth:_FillValue = NaNf ;
        double latitude(latitude) ;
                latitude:long_name = "latitude" ;
                latitude:units = "degrees_N" ;
                latitude:data_min = -9.37 ;
                latitude:data_max = -9.37 ;
                latitude:_FillValue = NaN ;
        double longitude(longitude) ;
                longitude:long_name = "longitude" ;
                longitude:units = "degrees_E" ;
                longitude:data_min = 132.436666666667 ;
                longitude:data_max = 132.436666666667 ;
                longitude:_FillValue = NaN ;
        double sampleTime(time, depth, latitude, longitude) ;
                sampleTime:long_name = "sample time" ;
                sampleTime:units = "milliseconds since 2008-01-01 00:00:00" ;
                sampleTime:data_min = 14083562281. ;
                sampleTime:data_max = 14083719890. ;
                sampleTime:_FillValue = NaN ;
        float temperature(time, depth, latitude, longitude) ;
                temperature:long_name = "temperature" ;
                temperature:units = "degree C" ;
                temperature:data_min = 19.061f ;
                temperature:data_max = 27.472f ;
                temperature:_FillValue = NaNf ;
        float resistance(time, depth, latitude, longitude) ;
                resistance:long_name = "resistance" ;
                resistance:units = "ohms" ;
                resistance:data_min = 4461.f ;
                resistance:data_max = 6517.f ;
                resistance:_FillValue = NaNf ;
        float procTemperature(time, depth, latitude, longitude) ;
                procTemperature:long_name = "processed temperature" ;
                procTemperature:units = "degree C" ;
                procTemperature:data_min = 19.061f ;
                procTemperature:data_max = 27.472f ;
                procTemperature:_FillValue = NaNf ;
        byte sampleQC(time, depth, latitude, longitude) ;
                sampleQC:long_name = "sample QC" ;
                sampleQC:units = "" ;
                sampleQC:data_min = 0b ;
                sampleQC:data_max = 0b ;
                sampleQC:_FillValue = -51b ;

// global attributes:
                :WOCE_VERSION = "3.0" ;
                :Conventions = "COARDS/WOCE" ;
                :Data_Type = "XBT" ;
                :Ship = "Wana Bhum" ;
                :CallSign = "HSB3403" ;
                :Voyage = "618/1118" ;
                :LineNo = "PX02" ;
                :Operator = "1st Officer         " ;
                :SoftwareVersion = "5.18" ;
                :CRC = "2a91bf39" ;
                :Type = "DeepBlue" ;
                :Code = "052" ;
                :InterfaceType = "Devil" ;
                :HardwareVersion = "4.4.1" ;
                :HardwareSerialNo = "45" ;
                :FirmwareVersion = "3.1" ;
```

```
                :HardwareCalibration = "Cal1 = 17998.2, Cal2 = 3900.5, Cal Date = 13:46 26/11/2007" ;
                :SerialNo = "904181" ;
                :TestCanister = "no" ;
                :Scale = "0.9991" ;
                :Offset = "10.0" ;
                :DropNo = "19" ;
                :WaterDepth = "NaN" ;
                :XBT_SST = "27.456" ;
                :XBT_SST_DEPTH = "NaN" ;
                :SeaState = " " ;
                :Operations = " " ;
                :PreDropComments = "MOD SEA,SPD 17.2 KTS,AIR TEMP 27 C" ;
                :PostDropComments = "\n",
                        "QC failure: Failed climatology test\n",
                        "" ;
}
```

Most of these variables are self-explanatory, however the following comments should be made.

The "temperature" variable is the temperature that the Devil/Quoll hardware reports. The "procTemperature" variable is the temperature after any processing due to the QC, ie smoothing, spike removal, etc. In the majority of cases these will be the same.

The "sampleQC" is a flag which indicates the status of each data sample and as a guide as to what has been done to get from "temperature" to "procTemperature". The values of the flags are defined in the attached documents. Note that since netCDF doesn't have an unsigned byte data class, the byte values are stored as signed values, ie if the value in the netCDF is negative you need to add 256 to convert to unsigned.

The “sampleTime” refers to either the *sampletime* or the *processtime* – see 2.1 Explanatory Notes on Parameters.

2.2.1 QC flags

Quality control flags are only available in the netCDF file. The following paragraphs define the flags.

All quality control flags are to be unsigned integer numbers in the range 0 to 255. Where appropriate, the flags are to be stored as unsigned byte length values. Each byte length QC flag is subdivided into 3 fields. These fields are defined as follows:

Data State (bits 6 & 7)

The data state describes the overall status of the data without concern about the type of error, and the type of correction process performed on the data, if any. If the QC is unknown, the person loading data must determine the data state, i.e. unknown QC does not necessarily imply no QC.

Data State	Numeric value	Description
0	0	Data is good
1	64	Data is suspect
2	128	Data is bad
3	192	No QC

Operation type (bits 4 & 5)

The operation type describes the type of operation performed on the data to enable it to be classified with the given data state.

Operation	Numeric value	Description
0	0	No operation – data used as is.
1	16	Data has been interpolated to replace bad values.
2	32	Data has been averaged or otherwise filtered.
3	48	Data has been manually adjusted.

Error type (bits 0 & 3)

The error type describes the type of data error detected which resulted in the given data state and subsequent operation on the data.

Error type	Numeric value	Description
0	0	No error – data is good, or if no QC, error is unknown.
1	1	Hardware error.
2	2	Software error.
3	3	Operator error.
4	4	Error flagged by hardware.
5	5	Error flagged by processor.
6	6	Analytical error.
7	7	Recording anomaly, e.g. transcription error.
8	8	Data stream corrupted, e.g. communications fault.
9	9	Data out of range.
10	10	Second difference out of range, e.g. data spikes.
11	11	Preliminary processing (calibration) only.
12	12	Unprocessed (uncalibrated) or processing error.
13	13	No data – data missing for unknown reason.
14	14	Timing error.
15	15	User defined – user must provide adequate description.

Numeric interpretation:

The complete flag for a given data element is the sum of the numeric values of the 3 fields. To unpack a flag, the user can either use a lookup table, or perform the following manipulations:

Arithmetic method	Bit manipulation method
To unpack a flag:	To unpack a flag:
state = int(flag / 64)	state = flag >> 6
op = int((flag – state * 64) / 16)	op = (flag & 0x30) >> 4
error = flag – state * 64 – op * 16	error = flag & 0x0f
To pack a flag:	*To pack a flag:*
flag = state * 64 + op * 16 + error	flag = (state << 6) & (op << 4) & error

On some systems and file formats, eg. netCDF, it is not possible to store unsigned byte values. In this case, flags greater than 127 are stored as negative numbers. To convert them to unsigned integers, add 256.

If a user is only interested in the state flag, the following can be used to interpret flags:

State	Unsigned Byte	Signed Byte
Good	$0 \le flag \le 63$	$0 \le flag \le 63$
Suspect	$64 \le flag \le 127$	$64 \le flag \le 127$
Bad	$128 \le flag \le 191$	$-128 \le flag \le -65$
No QC	$192 \le flag \le 255$	$-64 \le flag \le -1$

2.3 Exported Files

The underlying data in the netCDF files can be exported in ACII format for quick viewing or use in spreadsheets, etc. Exported files can be generated manually by operator intervention using the Export facility or they can be generated automatically after each drop or test (using the Configuration settings).

Select either the CSIRO Text or the CSV format for the profile listing or Metadata format for the meta data.

2.3.1 CSIRO Text Format

For CSIRO Text Format of netCDF file "drop008.nc" the example output "drop8.asc" is:

```
HShip  Lollipop
HCruise  SOTIV
HLineNo
HDrop number  8
HLatitude  49:00.00S
HLongitude  179:10.00E
HBottom depth unknown
HProbe type  DeepBlue
HHardware version  3.40.1.0
HHardware serial no.  031
HFirmware version  2.05
HHardware calibration  Cal1 = 17987, Cal2 = 3896, Cal Date = 08:46 02/05/2007
S Probe launched,10-May-2007,13:09:30
DDate  10-May-2007
DTime (UTC)  13:09:30
DLast header record
D    0.000,   0.67, 4703.500, 26.40
D    0.110,   1.34, 4703.400, 26.40
D    0.219,   2.01, 4702.900, 26.40
D    0.329,   2.68, 4703.200, 26.40
D    0.438,   3.34, 4703.600, 26.40
D    0.548,   4.01, 4703.200, 26.40
D    0.657,   4.68, 4703.200, 26.40
D    0.766,   5.35, 4703.300, 26.40
D    0.876,   6.02, 4703.400, 26.40
D    0.985,   6.69, 4703.200, 26.40
```

The lines beginning with '`H`' are header lines containing metadata.

The line beginning with '`S`' gives the launch date and time.

The lines beginning with '`D`' contain recorded data.

Each line of profile data (after the '`D`') contains:

> time in seconds, depth in metres, thermistor resistance in ohms, temperature in deg C

The "time" may be either the *sampletime* or the *processtime* – see 2.1 Explanatory Notes on Parameters.

2.3.2 CSV Format

For CSV of netCDF file “drop008.nc” the example output file “drop008.csv: is

```
Time,Depth,Resistance,Temperature
    0.000,   0.67, 4703.500, 26.40
    0.110,   1.34, 4703.400, 26.40
    0.219,   2.01, 4702.900, 26.40
    0.329,   2.68, 4703.200, 26.40
    0.438,   3.34, 4703.600, 26.40
    0.548,   4.01, 4703.200, 26.40
    0.657,   4.68, 4703.200, 26.40
    0.766,   5.35, 4703.300, 26.40
    0.876,   6.02, 4703.400, 26.40
    0.985,   6.69, 4703.200, 26.40
    1.095,   7.36, 4703.400, 26.40
    1.095,   8.03, 4703.500, 26.40
    1.204,   8.69, 4703.600, 26.40
    1.313,   9.36, 4702.400, 26.41
    1.423,  10.03, 4703.100, 26.40
```

The data in each column is as per the definition in the CSIRO Text Format.

As for the CSIRO Text Format the “time” may be either the *sampletime* or the *processtime* – see 2.1 Explanatory Notes on Parameters.

2.3.3 Meta Data

Meta data for any drop or test can be exported. For “drop003.nc” the following is an example of the meta data file “drop008.met”:

```
Data type: XBT
Ship: CHAIKA
Call sign: RG457
Voyage: 140212
Line no:
Operator: APop
Software version: 8.0.8
Release version: Test Version: 4.06.03
CRC: a4ffcb42
Probe type: DeepBlue
Probe code: 052
Temperature coefficients: -2.22,0.01145102,-1.0144E-6,1.791067E-10
Depth coefficients: 6.691,0.00225
Interface type: QuollUSB
Interface code: 72
Hardware version: 6.2.1
Hardware serial no.: 237
Firmware version: 7.06.00
Hardware calibration: Cal1 = 15993.6, Cal2 = 3899.0, Cal Date = 13:31 08/10/2013
Serial no.: 1165305
Test canister: no
Scale: 1.00015
Offset: -0.2
Drop no.: 3
Water depth: NaN
XBT SST: 16.81
XBT SST depth: 4.01
Sea state:
Operations:
Pre drop comments:
Post drop comments:
Drop time: 12/02/2014 00:21 +0000
Drop position: 43 13.800S 148 12.000E
```

2.4 JJVV File

The JJVV format is a binary format specified in

http://www.meds-sdmm.dfo-mpo.gc.ca/isdm-gdsi/gts-smt/codes/new-nouv-bath-eng.htm

In every installation MODE an ascii hexadecimal representation of a JJVV message is generated with a file extension of ".jjv". The following is an example:

```
JJVV 05028 2101/ 339000 139001 88888 05271 04264 13264 99901 12264
13240 13182 14127 15110 15127 16141 25141 25129 26094 27051 27022
28015 29017 99902 01017 SHIP=
```

The temperature depth points are obtained from the XBT Message used for transmission. The method used to obtain these points is either selected in the menu item Configuration->Transmitter Message or in the case of SECURE Mode the method used is CSIRO Fixed Tolerance using the Douglas Peucker Decimator (fixed tolerance).

The JJVV file is not transmitted, it is only generated and saved in the same directory as the netCDF files.

2.5 Sage XBT File

In some installation MODEs an ascii text file with extension ".xbt" is generated, in other MODEs it is possible to export in this format. This text file has the following format:

```
Sage Input File
07/02/2008
13:41
C:\XBT 080207\FEB08\drop001.nc
45 00 S
148 00 E
222
21
Depth Temp
0 23.6
47 23.7
49 23.8
59 23.6
62 23.4
67 23.0
76 22.5
78 22.4
80 22.3
86 21.9
99 21.4
103 21.2
105 21.0
125 20.7
132 20.5
141 20.3
175 19.6
203 19.2
204 19.1
205 19.0
222 18.3
```

Where the lines are:

Line 1	heading = "`Sage Input File`"
Line 2	date
Line 3	time
Line 4	the full path and file name to the netCDF file
Line 5	latitude
Line 6	longitude
Line 7	maximum depth of this profile in metres
Line 8	number of Depth Temperature points (metres, deg C) in this file
Line 9	heading "`Depth Temp`"
Line 10	first Depth Temperature pair (has to be a depth = 0 metres)
Line 11	second Depth Temperature pair eg "`1 25.01`" = 1 metre, 25.01°C
Lines >11	remainder of the Depth Temperature pairs

2.6 CALC File (sound speed)

Sound speed can be exported to an ascii file.

The file format for sound speed:

CALC, xxxx, 23/01/10, 0, meters
TURO Sound Velocity Calculator S/N:vvvv
DATE:yyddd TIME:hhmm
DEPTH OFFSET (M): 0.0
DEPTH (M) VELOCITY (M/S) TEMP (C)
0.4 1441.925 1.735
1.6 1442.023 1.753
….
….
0 0 0

Where:

Line 1: CALC, sn, 23/01/10, 0, meters

sn is the sensor serial number.

23/01/10 is the date this file is generated.

0 indicates that the depth increment is not uniform, otherwise this is the increment in "metres"

Line 2: TURO Sound Velocity Calculator S/N:vvvv

vvvv is the sound speed calculator software version number.

Line 3: DATE:yyddd TIME:hhmm

yyddd is the year and Julian day of the time of the measurement.

hhmm or hhmmss is the time of the start of this drop.

Line 4: DEPTH OFFSET (M): 00

gives the depth offset at sea level in metres.

Line 5: DEPTH (M) VELOCITY (M/S) TEMP (C)

gives the format of the data lines and units used.

Lines 6+: 0.4 1441.925 -1.735

is a space separated data line containing:

0.4	= depth (metres)
1441.925	= sound speed (m/s)
-1.735	= temperature (degC)

The numbers are separated by a space, and each line is terminated with a linefeed (LF). The numbers shall all include a decimal point (xxx.x xxxx.xxx xx.xxxLF). The data are terminated by a line with three zeros with two spaces between the zeros (0 0 0)

The filename of this file is the netCDF filename (eg "drop2") with the extension svp (eg "drop2.svp")

3 Satellite Transmission Formats

When data is transmitted by satellite the message may be broken into several packets – for example 4 packets of 32 byte Argos messages or several packets of 335 byte Iridium SBD messages (the number of bytes may change, see page 14, Recovering Iridium Transmissions:). These packets have to be reassembled into the one contiguous message. In other documents this is referred to as the TxData.

This poses restrictions on the size of the transmit message (TxData), depending on the particular transmitter. The following is implemented in the software:

ARGOS: maximum of 116 bytes

IRIDIUM: maximum of 8000 bytes

Support for the ARGOS satellite system is a legacy support that will discontinue in the future.

3.1 Packaging Argos Transmissions

The `TxData[T]` array is separated into consecutive 29 byte packets, each packet is preceded by a 3 byte header to produce a 32 byte packet. The length of `TxData` can be up to 116 bytes. If `TxData` is longer, then the array is truncated to use only the first 116 bytes, which are then packaged for transmission.

Maximum Argos lengths of 32 bytes are used. Four of these transmissions are used to send up to 116 bytes of data. If the data (`TxData[T]`) is less than 116 bytes, the transmission data is padded out with zeroes (0x00) to make up 116 bytes.

The consecutive transmissions are cycled through four 32 byte packets.

Header Definition

bits 0 to 15:	crc	this is a 16 bit cyclic redundancy check of the following 30 bytes (bits 16 thru 255)
bits 16 to 21:	sn	binary number, initially random, then incremented for subsequent `TxData`
bits 22 to 23:	txnum	binary number, starting at 0, then incremented for subsequent 32 byte packets of the current `TxData`.

CRC Definition

CRC-CCITT polynomial $X^{16} + X^{12} + X^5 + 1$

Width : 16
Poly : 1021
Init : FFFF
RefIn : False
RefOut : False
XorOut : 0000

Reference information from: Ross N. Williams, "A Painless Guide to CRC Error Detection Algorithms", 19 August 1993.

Transmission Byte Order

The following illustrates the packaging a message of 80 bytes (`TxData[80]`) for **Argos**.

assume `T = 80` (length of `TxData`)
assume `sn binary = 001000`

ARGOS BYTE	ARGOS MESSAGE /PACKET		DATA
1st-2nd	message 1	=	CRC
3rd	message 1	=	0x20
4th-32nd	packet 1	=	TxData[0 thru 28]
1st-2nd	message 2	=	CRC
3rd	message 2	=	0x21
4th-32nd	packet 2	=	TxData[29 thru 57]
1st-2nd	message 3	=	CRC
3rd	message 3	=	0x22
4th-25th	packet 3	=	TxData[58 thru 79]
26th-32nd	packet 3	=	0x00
1st-2nd	message 4	=	CRC
3rd	message 4	=	0x23
4th-32nd	packet 4	=	0x00

3.2 Recovering Argos Transmissions:

Recovering the data involves acquiring a collection of Argos messages. Each message should be 32 bytes long. The data is reassembled in the steps:

1. Check that the Argos messages have come from the same Argos ID.

2. A TxData is broken up into 4 packets, each packet is in a single Argos message. The first 3 bytes of each Argos message consists of a 2 byte CRC (1st & 2nd bytes) and a sn_txnum number (3rd byte). The four Argos messages that contain the four consecutive packets making up a single TxData have consecutive sn_txnum (3rd byte) values beginning with XXXX XX00B. Identify these four Argos messages.

3. Check each message for errors by calculating the CRC and comparing it with the CRC in the first 2 bytes of the message. If the calculated CRC is different from that in the message, there are errors in that message.

4. Assemble the full TxData by concatenating the 4th through 32nd bytes of each consecutive message (116 bytes in total). This is the raw TxData from the Devil XBT system.

3.3 Packaging Iridium Transmissions

The Iridium transmission uses the Short Burst Data mode. The Transmitter supported is the NAL 9601-D. This unit only supports 340 bytes transmit (Mobile Originate – MO) and 270 bytes receive (Mobile Terminate MT).

The TxData message can be up to 8000 bytes. TxData is divided into 335 byte parcels. Each parcel has a 5 byte header and then it is transmitted as a single SBD packet.

SBD Packets

The 5 byte header has the following definition:

BYTE
1st-2nd = Sequence Number (unsigned int)
3rd = Parcel Number (SBD number) (unsigned char)
4th = Total number of Parcels (SBD's) (unsigned char)
5th = not used

If CONTConfigure has the "ASCII" parameter in the command, then no header is implemented and the data begins at the first byte.

When the header is implement the remainder of the SBD packet contains up to 335 bytes of TxData. Otherwise when the header is not implemented the SBD packet contains up to 340 bytes of TxData.

The Sequence Number begins at a random number the first time the program is used after the program is installed and then incremented for every TxData.

The Parcel Number begins at 1 for every new TxData and is incremented for each SBD packet.

NOTE: The number of bytes in an SBD (MT or MO) may vary in future as Iridium and/or NAL change the technology.

3.4 Recovering Iridium Transmissions:

There are 2 ways to sort the order of the SBD packets.

1. Each SBD packet arrives as an attachment to an email. The filename of the attachment includes a sequence number allocated by the 9600-D transmitter and incremented for each subsequent packet.

2. The other way is to use the 5 byte header of the data which is present when the "ASCII" parameter in the command CONTConfigure was not used. For a particular message (TxData), every packet has the same Sequence Number. So the packets with the same sequence number have the Parcel Number incremented (beginning at 1) for each successive parcel. The parcels are then concatenated, leaving out the 5 byte header. Thus reassembled, it is the raw TxData from the Devil XBT system.

3.5 Transmitted Data Format (TxData)

Once reassembled the message (TxData) format is the same irrespective of the transmitter used.

There are 2 formats: BOM format and CSIRO format. The CSIRO format has all the parameters that the BOM format has plus several extra features. The extras features are:

- Number of Temperature Depth points is limited to 63 in the BOM format and has a maximum of 16383 in the CSIRO format.
- The CSIRO format has provision for a 9 character ship call sign. BOM format has no provision for a call sign.

The practical affect of these differences is that it is possible to recreate a JJVV file from the CSIRO format, but not possible from a BOM format because of the ship call sign, and it is possible to have much longer messages. Note that to have longer messages it is necessary that the transmitter being used can support longer messages.

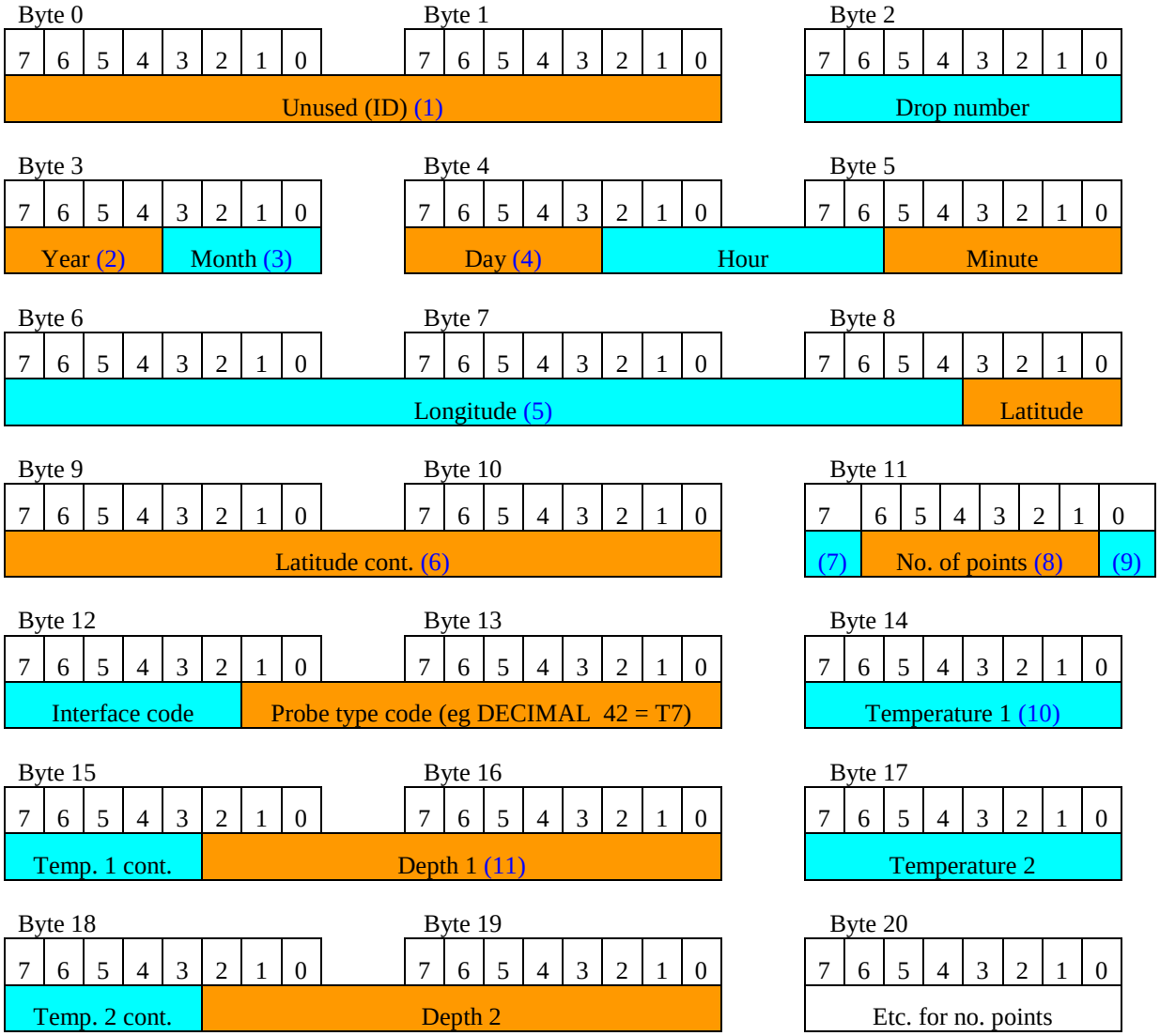
Note that the maximum number of Temperature Depth points mentioned here (63 and 168383) are defined for these two formats (BOM and CSIRO). But when they are packaged for transmission, another restriction is set by the particular transmitter used (Argos or Iridium – see section 2.6 CALC File (sound speed)) that further limits the number of Temperature Depth points – these are:

- 34 points for BOM format using Argos or Iridium
- 31 points for CSIRO format using Argos
- 2262 points for CSIRO format using Iridium (in the future, if there is demand, this will be changed to accommodate up to 16383 points)

It is recommended that ALL new applications use the CSIRO format and not the BOM format.

3.5.1 BOM Format

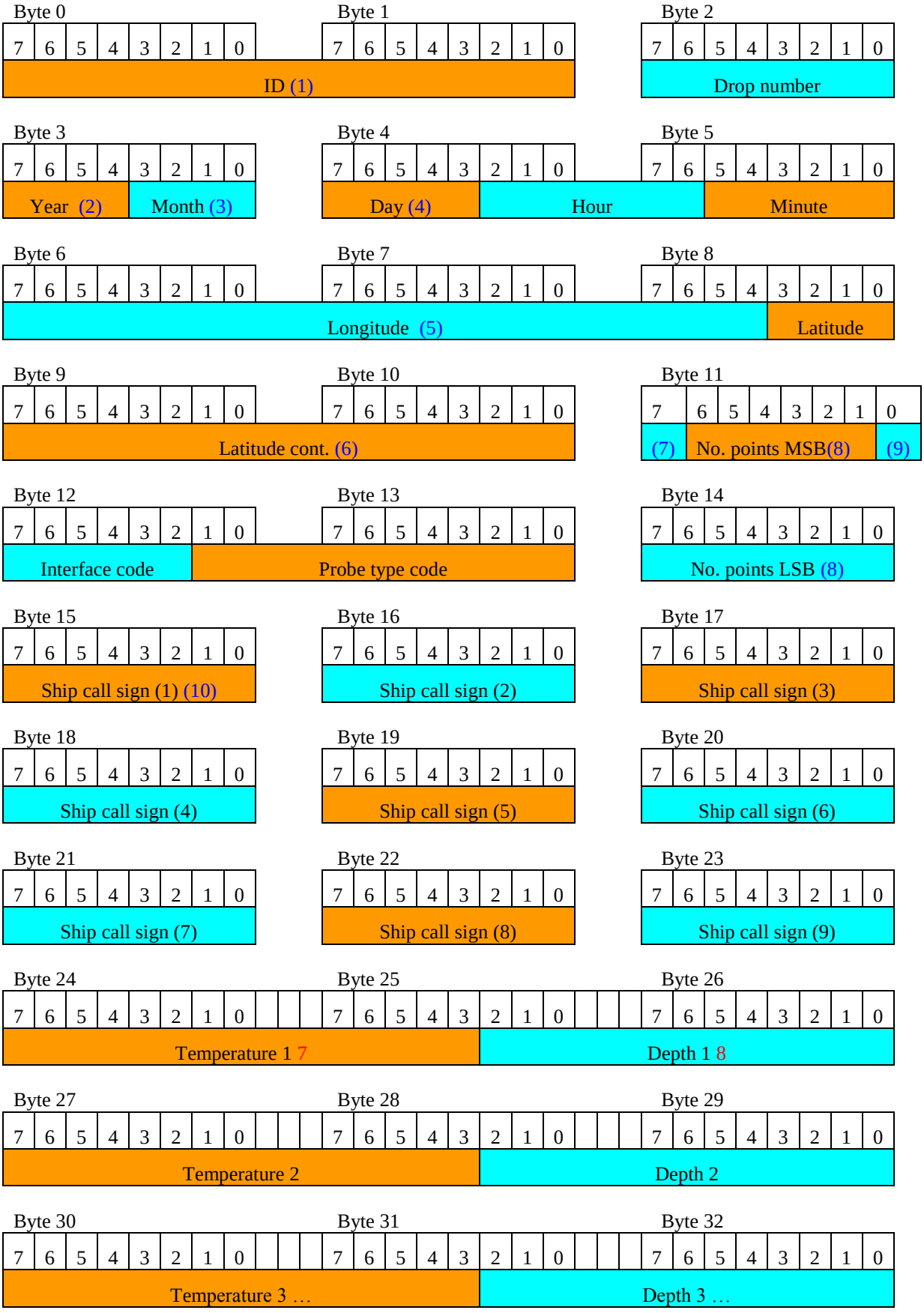
The following specifies the reassembled TxData when in BOM format:



1. Message type ID. Ascii ‘B2’ if fixed length (34 points) or ‘B3’ for fixed tolerance (maximum of 63 points)
2. Year is stored as modulo 16 (eg 2008/16 = 125 with 8 remainder, 8 = year)
3. Month number = 0 -> 11 = Jan -> Dec
4. Day number = 1 -> 28, or 29 or 30 or 31 (0 = invalid)
5. Longitude * 2900 (eg HEX27989 = DECIMAL 162185/2900 = 55.92 = 55deg.55min 55sec E)
6. (Latitude + 90) * 2900 (eg HEX42CF = DECIMAL 271311/2900 = 93.55 (-90) = 3deg.33min 33sec N)
7. If set, this bit indicates that data is to be sent to the GTS.
8. Number of Temperature/Depth points here (max 63)
9. Most significant bit of interface type code
10. (Temperature (ºC) + 3) * 200 (eg HEX 18F6, DECIMAL 6390 = 6390/300 - 3 = 28.95)
11. Depth * 2 (eg HEX 7B, DECIMAL 123 = 123/2 = 61.5)

3.5.2 CSIRO Format

The following specifies the reassembled TxData when in CSIRO format:



1. Message type ID. Ascii ‘C2’ if fixed length (eg 34 points) or ‘C3’ for fixed tolerance (maximum of 63 points)
2. Year is stored as modulo 16 (eg 2008/16 = 125 with 8 remainder, 8 = year)
3. Month number = 0 -> 11 = Jan -> Dec
4. Day number = 1 -> 28, or 29 or 30 or 31 (0 = invalid)
5. Longitude * 2900 (eg HEX27989 = DECIMAL 162185/2900 = 55.92 = 55deg.55min 55sec E)
6. (Latitude + 90) * 2900 (eg HEX42CF = DECIMAL 271311/2900 = 93.55 (-90) = 3deg.33min 33sec N)
7. If set, this bit indicates that data is to be sent to the GTS.
8. Number of Temperature/Depth points here (max 16383)
9. Most significant bit of interface type code
10. Ship call sign in ASCII characters (max 9 characters)
11. (Temperature (ºC) + 3) * 200 (eg HEX 18F6, DECIMAL 6390 = 6390/300 - 3 = 28.95)
12. Depth * 2 (eg HEX 7B, DECIMAL 123 = 123/2 = 61.5)