An X message is used to transfer data from ISU to GS (ground station) or from GS to ISU. The data is assumed to be binary and each byte can have any value from 0x00 to 0xff. The format of the message is the same regardless of direction of transmission:

\[
\text{Xnnmmddp<data>$cc>}
\]

- **X** = the character X
- **nn** = number of data characters in the message following after nn. The count does not include X, nn, or anything from $ to the end >. The count is in 2 binary bytes with MSB first and LSB second.
- **mm** = serial number of SOLOII. The SN is in 2 binary bytes with MSB first and LSB second.
- **dd** = the dive number in 2 binary bytes with MSB first and LSB second. : Dive number begins at -1 for the start-up, increments to 0 for the test dive, increments +1 for all normal (0xE2) dives.
- **p** = one-byte packet ID index, range 0 to 255. Used to identify multiple X messages within a dive cycle. The data for each dive cycle starts with p=0.
- **<data>** = binary data characters. The length of <data> = nn-5. The contents of the <data> section is described below.
- **$** = a dollar sign delimiter at start of the checksum
- **cc** = the 8 bit byte-wise checksum from X to the byte preceding the $. The 8 bit sum is coded as 2 4bit nibbles. The binary value of a nibble is converted to a visible character by adding 0x30. Thus a value of 0x0 -> 0x30 = character '0', 0x1 -> 0x31 = '1', 0xe -> 0x3e = '>', and 0x0 -> 0x3f = '?'.
- **>** = a > delimiter at end of checksum which also serves as a prompt to GS that the ISU is done transmitting and that the GS may now transmit to ISU.

The remainder of this document describes the format of the <data> portion of the message sent from SOLOII to the ground station (GS). The format of commands from GS sent to SOLOII will be described in another document.

Highlights in document

- Fields that are moved relative to the previous float version are highlighted in cyan
- New fields relative to the previous version are highlighted in yellow

NOTE: For firmware including and later than V1.5: An ice detection algorithm was introduced. However, for the float to be able to use the algorithm, the Seabird CTD must be a SBE41CP-V3.0C. SBE41CP-V3.0A, does not contain the necessary commands. Thus not all floats may have operational ice detection due to hardware.

BUGS:

For firmware 15Feb17, the Voltage and Current are bad within the pump message for 'short' pumps. This includes the pump at 100dabar (code=2), Seeks (code=3), and other seeks during drift and falling (codes 4, 5, 10 and 12).
The <data> section contains information from multiple sensors. Data from successive sensors are separated by a semicolon (‘;’ = 0x3b); the final sensor is terminated by a ‘;’ (immediately preceding the $ delimiter).

```
IDjj<sensor_data>;
ID = one-byte sensor ID code.
jj = Number of bytes for this sensor. The count includes ID, jj, and the trailing ;.
The count is in 2 binary bytes with MSB first and LSB second.
<sensor_data> = binary data characters. The length of <sensor_data> = jj-4 bytes, and its contents are described below for each sensor.
; = delimiter at the end of each sensor’s data.
```

The ID byte is divided into two 4-bit nibbles. The MS nibble identifies the sensor and the second nibble specifies the message number for that sensor. For example, the ID for first Pressure message is 0x10, the second is 0x11, the third 0x12, etc.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>ID byte(hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>00</td>
<td>fix at end of first diagnostic dive at start of mission</td>
</tr>
<tr>
<td>GPS</td>
<td>01</td>
<td>fix at before leaving surface</td>
</tr>
<tr>
<td>GPS</td>
<td>02</td>
<td>fix at end of normal profiling ascent</td>
</tr>
<tr>
<td>GPS</td>
<td>03</td>
<td>fix following mission abort</td>
</tr>
<tr>
<td>GPS</td>
<td>05</td>
<td>fix during BITest</td>
</tr>
<tr>
<td>Pressure</td>
<td>1x</td>
<td>depths of CTD readings</td>
</tr>
<tr>
<td>Temperature</td>
<td>2x</td>
<td>depth series of temperature</td>
</tr>
<tr>
<td>Salinity</td>
<td>3x</td>
<td>depth series of salinity</td>
</tr>
<tr>
<td>Fall Rate</td>
<td>4x</td>
<td>series of time, depth during SOLO II downward profile</td>
</tr>
<tr>
<td>Rise Rate</td>
<td>5x</td>
<td>series of time, depth from drift depth to surface</td>
</tr>
<tr>
<td>Pump Series</td>
<td>6x</td>
<td>pressure, time, voltage, current, vacuum for each pump</td>
</tr>
<tr>
<td>High Resolution Pressure</td>
<td>9x</td>
<td>High Resolution Pressure [x=0-7]</td>
</tr>
<tr>
<td>High Resolution Temperature</td>
<td>ax</td>
<td>High Resolution Temperature [x=0-7]</td>
</tr>
<tr>
<td>High Resolution Salinity</td>
<td>bx</td>
<td>High Resolution Salinity [x=0-7]</td>
</tr>
<tr>
<td>Drift Profile Pressure</td>
<td>9x</td>
<td>Drift profile of Pressure [x=8-F]</td>
</tr>
<tr>
<td>Drift Profile Temperature</td>
<td>ax</td>
<td>Drift profile of Temperature [x=8-F]</td>
</tr>
<tr>
<td>Drift Profile Salinity</td>
<td>bx</td>
<td>Drift profile of Salinity [x=8-F]</td>
</tr>
<tr>
<td>Mission EEPROM</td>
<td>dx</td>
<td>ASCII dump of mission parameters in EEPROM</td>
</tr>
<tr>
<td>Engineering</td>
<td>e0</td>
<td>diagnostic data in first diagnostic dive</td>
</tr>
<tr>
<td>Engineering</td>
<td>e2</td>
<td>engineering data in normal profiling dive</td>
</tr>
<tr>
<td>Engineering</td>
<td>e3</td>
<td>engineering data following mission abort</td>
</tr>
<tr>
<td>Engineering</td>
<td>e5</td>
<td>engineering data BIT test pass</td>
</tr>
<tr>
<td>Engineering</td>
<td>e6</td>
<td>engineering data BIT test failure</td>
</tr>
<tr>
<td>Argo Data</td>
<td>f0</td>
<td>Mission parameter list</td>
</tr>
<tr>
<td>Test pattern</td>
<td>f1</td>
<td>ID reserved, format not yet defined</td>
</tr>
</tbody>
</table>
GPS data (ID=0x00, 0x01, 0x02, 0x03, 0x05)

The LS nibble of the ID indicates in what phase of the mission the fix was taken. The remainder of the data is the same for all mission phases. The length of GPS data is in bytes 1 and 2. GPS fix data starts in byte 3:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mission phase:</td>
</tr>
<tr>
<td></td>
<td>0 = 1st diagnostic dive at the start of a mission</td>
</tr>
<tr>
<td></td>
<td>1 = beginning of normal dive cycle (just before leaving surface)</td>
</tr>
<tr>
<td></td>
<td>2 = end of a normal dive cycle</td>
</tr>
<tr>
<td></td>
<td>3 = following mission abort</td>
</tr>
<tr>
<td></td>
<td>5 = during BITest</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes in the message, 24 = 0x18 with the format as described here</td>
</tr>
<tr>
<td>3</td>
<td>0 if fix is invalid, +2 if longitude is East, -2 if longitude is West</td>
</tr>
<tr>
<td>4-7</td>
<td>Signed latitude degrees * 1e7</td>
</tr>
<tr>
<td>8-11</td>
<td>Signed longitude degrees * 1e7 range (+180 to -180 degrees)</td>
</tr>
<tr>
<td>12-13</td>
<td>GPS week (traditional GPS week =0 to 1023 in LS 10 bits; rollover fix in MS 6 bits)</td>
</tr>
<tr>
<td>14</td>
<td>GPS day of week, 0=Sunday, 6=Saturday</td>
</tr>
<tr>
<td>15</td>
<td>UTC hour</td>
</tr>
<tr>
<td>16</td>
<td>UTC minutes</td>
</tr>
<tr>
<td>17</td>
<td>Time to get fix = (seconds to get fix)/10 , range 0 to 255 = 0 to 2550 seconds</td>
</tr>
<tr>
<td>18</td>
<td>Number of satellites used in fix</td>
</tr>
<tr>
<td>19</td>
<td>Minimum signal level</td>
</tr>
<tr>
<td>20</td>
<td>Average signal level</td>
</tr>
<tr>
<td>21</td>
<td>Maximum signal level</td>
</tr>
<tr>
<td>22</td>
<td>10*Horiz. dilution of precision</td>
</tr>
<tr>
<td>23</td>
<td>; terminator (0x3B)</td>
</tr>
</tbody>
</table>
Pressure data (ID=0x1n)
Temperature data (ID=0x2n)
Salinity data (ID=0x3n)

Profile data from the pressure, temperature, and salinity sensors are all processed in the same way and the message format differs only in the ID code. The SeaBird CTD takes a profile as the SOLOII ascends and stores the values internally. When SOLOII reaches the surface, it takes the data from the CTD and block averages it in depth into PRO_BINS ( =1000) bins.

The size of depth bins can vary with depth. The averaging scheme is determined by 5 parameters: BLOK, PB1, PB2, AV1, and AV2. The smallest bin size is BLOK decibars. Bins 0 thru PB1-1 have a vertical extent of BLOK decibars. Bins PB1 thru PB2-1 are AV1*BLOK decibars tall while bins PB2 thru PRO_BINS-1 are AV2*BLOK decibars. In the special case that PB1 >= PRO_BINS, then all of the bins are BLOK decibars in extent, and the values of PB2, AV1, and AV2 are ignored.

There are two options for packing the Core (bin averaged) profile data. The packing used within the message data stream is indicated via first nibble of the jj variable (see below).

1. Difference Packing (Standard to versions previous to V2.0, Optional in V2.0)

The data series from all channels are processed in the same way and are synchronous with each other. Each depth series is broken into sub-blocks of 25 samples, and a first-differencing method is applied to each sub-block to reduce the number of bytes required to transmit the data. Because the data series will generally be longer than the 189 bytes available in a 9601 SBD message, it is divided into multiple messages. Each message has an integral number of sub-blocks in it. The final sub-block of the time series may have fewer than 25 samples in it. The data message looks like:

\[ \text{ID}j<\text{sub-block 0}><\text{sub-block 1}> \ldots <\text{sub-block m}>; \]

\[ \text{ID} = \text{one-byte sensor ID code and index. The low order hex digit is the message index for this sensor. For example, the pressure messages would have ID's:10,11,12...} \]

\[ j = \text{Profile Packing Format (MS nibble)/Number of bytes for this message (LS 3 nibbles).} \]
\[ \text{Profile Packing Format = 0 for Legacy Diff. (backwards compatible), 1 for Curv.} \]
\[ \text{Number of bytes count includes ID, jj, the data, and the trailing ;.} \]

\[ <\text{sub-block i}> = \text{first-differenced data from the ith sub-block where } i = 1, \ldots, m = \text{number of sub-blocks. If } i < m, \text{ the sub-block will have 25 values in it and will have a total length of 22 bytes. The mth sub-block will have between 1 and 25 values and a length between 3 and 27 bytes.} \]

Suppose a sub-block has the n values \( v[0], v[1], \ldots v[n-1] \). Then this sub-block will be transmitted as:

<table>
<thead>
<tr>
<th>Sub-block Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>one-byte scaling factor ( S ), range = 1 to 255. ( S ) is chosen so that the scaled first-differences fit in one byte, i.e. (</td>
</tr>
<tr>
<td>1</td>
<td>MS byte of ( v[0] )</td>
</tr>
<tr>
<td>2</td>
<td>LS byte of ( v[0] )</td>
</tr>
<tr>
<td>3</td>
<td>LS byte of { ( v[1] - v[0] ) }/S</td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>n+1</td>
<td>LS byte of { ( v[n-1] - v[n] ) }/S</td>
</tr>
</tbody>
</table>

Each sub-block requires \( n+2 \) bytes so the longest sub-block uses 27 bytes. If each sensor has 1000 blocks then it will require 50 sub-blocks, each with 27 bytes. 8 sub-blocks will fit into each message (189/22) so 7 messages are needed per sensor. The total bytes then is \( 50*22 + 7*16 \) which equals 1212. Thus a CTD profile with 1000 blocks can be sent in \( 3*1212 = 3636 \) bytes.
2. Curvature Packing (New to V2.0 and later)

The packing routine is introduced to reduce the volume of transmitted data, primarily by allowing for variation in the bytes allotted for the data. The bytes allotted will be constant within a 16 value sub-block, but will differ between parameters and between sub-blocks of the same parameter.

\[
\text{IDjBBNNVVVDdppppppppp<sub-block 0><sub-block 1>...<sub-block sb>};
\]

- **ID** = one-byte sensor ID code and index. The low order hex digit is the message index for this sensor. For example, the pressure messages would have ID’s: 10, 11, 12... and message index (m) of 0, 1, 2, ...
- **jj** = Profile Packing Format (MS nibble)/Number of bytes for this message (LS 3 nibbles).
- **Profile Packing Format** = 0 for Legacy Diff. (backwards compatible), 1 for Curv.
- Number of bytes count includes ID, jj, the data, and the trailing ;.
- **B** = count of first sub-block number in message, as 1 byte. For message index, m = 0, B = 0, for succeeding messages m > 0, B > 0. The position (n) of the first value recorded in a message (VVV) can be computed as TopIndx = m + B * 16, where m is the message index.
- **NN** = total number of values given in the message as 2 bytes.
- **VVV** = \( v[n=\text{TopIndx}] \) first value as 3 binary bytes. In all messages greater than 1, VVV will be the same value as the last value packed in the previous message. Said another way, there is an overlap of 1 value between messages. This allows an additional check on the validity of the data transmitted.
- **DDD** = \{ \( v[n=\text{TopIndx}+1] - v[n=\text{TopIndx}] \) \} first-differenced, second value as 3 bytes.
- **pppppppppppp** = (12 bytes) packing factors for the sub-block second differences where each 3 bits indicate the dynamic range for each sub-block. The packing factor will be the number of nibbles needed to represent the dynamic range of the variable. For example, if the range is from 7 to -7, then the value can be expressed unambiguously using 1 nibble and the packing factor would be 1. Using 12 bytes for the packing factors, there can be up to 32 sub-blocks, or 512 values if the packing factor is 1 (1 nibble). Unfilled factors are valued at 0.

<sub-block i> = \{ \( v[n+2+i*16] - 2^*v[n+1+i*16] + v[n+i*16] \) \} where \( n=0,...,15 \) and \( i=1,...,sb \) =number of sub-blocks.

Each non-last sub-block (i=1: sb-1) will have 16 values in it and will have a total length of of 8 to 32 bytes. The last sub-block (i=sb) will have between 1 and 16 values and a length between 1 and 32 bytes. Message index m > 1 (example ID=11) overlap the previous message index m-1 by 1 value. Thus the VVV value in message index m will be redundant with the last value from message index m-1. If all sub-blocks are full in message index m, then the message contains values for index \( n = m + B * 16 \) through \( n = 1 + m + (B + sb_m) * 16 \), where \( sb_m \) is the number of sub-blocks in the message m.

Suppose sub-block i has \( n \) values \( v[2+i*16], v[3+i*16],...v[n+2+i*16] \), and the packing factor = 2 Then this sub-block will be transmitted as:

<table>
<thead>
<tr>
<th>Sub-block Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( v[2+i<em>16] - 2^<em>v[1+i</em>16] + v[i</em>16] )</td>
</tr>
<tr>
<td>1</td>
<td>( v[3+i<em>16] - 2^<em>v[2+i</em>16] + v[1+i</em>16] )</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>( v[n+2+i<em>16] - 2^<em>v[n+1+i</em>16] + v[n+i</em>16] )</td>
</tr>
</tbody>
</table>

Within a message, the original values can then be reconstructed by (1) starting with DDD and doing a cumulative sum of the entries for the sub- blocks, and then (2) using these values and starting with VVV doing a second cumulative sum.
Missing Data

The profile series will have gaps in it if there is no valid CTD data in a block. In that case, all of the profile series will be missing the same gap. If a block average contains no valid data, that block is ignored and is not transmitted. For example, suppose the pressure bin size is 1 db and that bin 0 has P=0. Suppose there is no valid data in bin 5. Then the sub-block will contain:

1 0000 01 01 01 01 02 01 ...
  ^  ^  ^  ^  ^  ^  ^
  |  |  |  |  |  |  |
  |  |  |  |  |  |  | +P=0007
  |  |  |  |  |  |  | +P=0006
  |  |  |  |  |  |  +P=0004
  |  |  |  +P=0003
  |  +P=0002
  |  +P=0001
  |  +P=0000

Note that the 6th bin, for which P=5, will be omitted from the pressure, temperature, and salinity messages.

Converting to scientific Units

After the sub blocks have been reassembled into a sequence of observations, the counts are converted to scientific units by:

- dBar = pressure counts *Pgain - Poff
- degC = temperature counts *Tgain - Toff
- psu = salinity counts *Sgain - Soff

The values of Gain/Offset are now sent back within the Argo Metafile message (0xf0) for data decoding purposes allowing a way to determine what Gain/offset is used in a given cycle. The GAIN/OFFSET of Temperature/Salinity/Pressure can be modified via 2-way communication. Modifying these parameters will effect all variables returned.

High Resolution Pressure data (ID=0x9n, n=0:7)
High Resolution Temperature data (ID=0xan, n=0:7)
High Resolution Salinity data (ID=0xbn, n=0:7)

The float can be set to return a high resolution P,T,S profile spanning a subsection of the primary binned profile. Data is packed and decoded similarly to the binned profile (ID=0x1n, 0x2n, 0x3n). The High Resolution profile can return every scan of the CTD (1 Hz) or every other scan (1/2 Hz). The data is limited to 1024 values. [Note: Typical SOLO II controller averaging uses every other CTD scan. However if the High Resolution profile includes every scan, the bin averages will also use every scan. Thus the averaging of the primary binned profile may differ between the subsection with High Resolution data and all other spans. If the Seabird does the binning, this is not an issue (for V2.1 11Dec15 and later). High Resolution profile data is decoupled from BinMod and is set to always use 'curvature packing'.

Drift Pressure timeseries data (ID=0x9n, n=8:f)
Drift Temperature timeseries data (ID=0xan, n=8:f)
Drift Salinity timeseries data (ID=0xbn, n=8:f)

The float can be set to return a timeseries of P,T,S recorded during the drift phase. Data is packed and decoded similarly to the binned profile (ID=0x1n, 0x2n, 0x3n), thus no time information is returned. Time can be estimated from the rise/fall records and the sampling interval of the drift data. The first value is taken a few seconds after drift begins, while the last value is taken a few seconds before drift ends. Thus there should be NsamX+1 values. The data is limited to 1024 values. Drift data is decoupled from BinMod and is set to always use 'curvature packing'.
**Fall Rate data (ID=0x4n)**

As it falls from the surface to its drift depth, SOLOII periodically interrogates the SeaBird for a depth reading. This time series is sent back in this data message.

The data message looks like:

```
IDjj<start_time><time(1),depth(1)> . . . <time(m),depth(m)>
```

- **ID** = one-byte sensor ID code = 0x4n.
- **jj** = Fall Packing Format (MS nibble)/Number of bytes in the message (LS 3 nibbles). Fall Packing Format = 0 for Legacy 4 byte reporting (backwards compatible), 1 for 5 byte reporting. The count includes ID, jj, the data, and the trailing ;.
- **start_time** = SOLO time at start of message (seconds since 1Jan2000) in 4 bytes (MSB first). This will be start of Fall in ID=0x40.
- **time(i)** = seconds since start_time in 2 bytes, i=1, ..., m.
- **code(i)** = Code representing float phase while data value recorded in 1 nibble, i=1, ..., m.

Possible Phase codes values

- START_OF_SINK =1, Buoyancy at 100db =2, SEEK =3, BEGINNING_OF_DRIFT =4, SEEK_DURING_DRIFT =5, BEGINNING_OF_FALL_TO_PROFILE =6, START_OF_RISE =7, END_OF_RISE =8, ICE_TURNAROUND =9, SINKING =10, DRIFTING =11, FALLING_TO_PROFILE =12, RISING =13, SURFACE =14 ;

- **depth(i)** = depth (LSB=Pgain db) at time(i) in 2.5 bytes, i=1, ..., m.
  - dBar = Pgain * depth(i) - Poff
  - depth(i) = 0xffff if the pressure reading is invalid

Each depth observation takes 5 bytes. The first time is taken when the valve is opened to leave the surface. The next two times are when the float passes 50m and 100m. After 100 m, pressures are logged every 30 minutes. Typically we allow for 500 (Falln) minutes for the SOLOII to fall 1000 meters so there will be about 16 more measurements. The last record should be recorded when the float begins its park phase.

Fall Rate data can be found over multiple messages.

The float resets its clock using GPS typically, but can use Iridium time if GPS is unavailable. Any re-epoch of the Iridium system will shift the reported time (after reset by Iridium) by 226492400 seconds.

Maximum of 240 Fall values can be transmitted in a given cycle.
Rise Rate data (ID=0x5n)

The rise rate message is identical in structure to the fall rate message. The rise rate time series begins when the SOLO II opens its valve to descent from the drift depth to the profile depth. It logs a pressure/time record 10 times during its descent to the profile depth (interval = PwaitN/10). At the bottom of dive, whether determined by timing out (exceeding PwaitN) or by reaching the target depth (ZproN), another pressure/time record is logged. At this point, the float pumps for PmpBtm seconds. A pressure/time record is logged every 30 minutes while the float is ascending. An additional Rise Rate record has been introduced in V2.0 which is measured AFTER surface buoyancy pumping.

Rise Rate data can be found over multiple messages. 
Note: There is a bug in the rise record data, when the ice-estimation routine is tripped. The bug leads to there being multiple code=9 (Ice Turnaround), in the rise record. The first code=9 is valid, the rest are not.

Maximum of 240 Fall values can be transmitted in a given cycle.

Pump data (ID=0x6n)

The data message looks like:

\[
\text{ID}j< \text{depth}(1), \text{time}(1), \text{voltage}(1), \text{current}(1), \text{vac0}(1), \text{vac1}(1)> \ldots
\]

\[
< \text{depth}(m), \text{time}(m), \text{voltage}(m), \text{current}(m), \text{vac0}(m), \text{vac1}(m);
\]

\[\text{ID} = \text{one-byte sensor ID code = 0x6n.}\]

\[jj = \text{Pump Packing Format (MS nibble)/Number of bytes in the message (LS 3 nibbles). Pump Packing Format = 0 for Legacy 10 byte packing (backwards compatible), 1 for 11 byte packing. The count includes ID, jj, the data, and the trailing ;.}\]

\[\text{code}(i) = \text{Code representing float phase in 1 nibble, i=1, ..., m (See Fall for values).}\]

\[\text{depth}(i) = \text{depth (LSB=Pgain db) at time}(i) \text{ in 2.5 bytes, i=1, ..., m.}\]

\[d\text{Bar} = \text{Pgain} \times \text{depth}(i) - \text{Poff}\]

\[\text{depth}(i) = \text{0xffff if the pressure reading is invalid}\]

\[\text{time}(i) = \text{seconds the pump ran in 2 bytes (signed)}\]

\[\text{voltage}(i) = \text{average pump battery counts while pumping in 2 bytes (0.01V)}\]

\[\text{current}(i) = \text{average pump current at bottom in 2 bytes, LSB=1ma}\]

\[\text{vac0}(i) = \text{vacuum counts after pump starts in 1 byte}\]

\[\text{vac1}(i) = \text{vacuum counts before pump stops in 1 byte}\]

Pump time series can be found over multiple messages.

The pump time series typically reports the pressure prior to pumping. However in the last pump record (Code=14), the pressure reported is post pumping.

For firmware 15Feb17, the Voltage and Current are bad within the pump message for ‘short’ pumps. This includes the pump at 100dbar (code=2), Seeks (code=3), and other seeks during drift and falling (codes 4, 5, 10 and 12).
Engineering data (ID=0xe0, 0xe2, 0xe3, 0xe5, 0xe6)

The engineering data is used to diagnose SOLOII anomalies. A different format is used in each of the 3 distinct phases of a SOLOII mission. The LS nibble of the ID indicates the phase of the mission.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase:</td>
</tr>
<tr>
<td></td>
<td>0xe0 = 1st diagnostic dive at the start of a mission</td>
</tr>
<tr>
<td></td>
<td>0xe2 = end of a normal dive cycle</td>
</tr>
<tr>
<td></td>
<td>0xe3 = following mission abort</td>
</tr>
<tr>
<td></td>
<td>0xe5 = BITtest</td>
</tr>
<tr>
<td></td>
<td>0xe6 = BITtest failure</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes in the message, depends on mission phase as described below</td>
</tr>
<tr>
<td>3</td>
<td>Depends on mission phase as described below</td>
</tr>
</tbody>
</table>

ID=0xe0, Engineering message in 1st diagnostic dive at start of mission

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xe0</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 80= 0x50</td>
</tr>
<tr>
<td>3</td>
<td>Engineering message version =5</td>
</tr>
<tr>
<td>4</td>
<td>#packets in current session</td>
</tr>
<tr>
<td>5-10</td>
<td>0 (dummy filler)</td>
</tr>
<tr>
<td>11-12</td>
<td>EP -&gt; sattime</td>
</tr>
<tr>
<td>13-14</td>
<td>DP-&gt;Vcpu = CPU battery voltage counts 0.01V</td>
</tr>
<tr>
<td>15-16</td>
<td>DP-&gt;Vpmp = Pump battery counts at surface(0.01V)</td>
</tr>
<tr>
<td>17-18</td>
<td>DP-&gt;Vple = Pump battery counts at end of last pump(0.01V)</td>
</tr>
<tr>
<td>19-20</td>
<td>BTvac = pcase vacuum at beginning of BIT in 0.01 inHg</td>
</tr>
<tr>
<td>21-22</td>
<td>DP-&gt;Air[1] = vac before filling bladder at surface 0.01 inHg</td>
</tr>
<tr>
<td>23-24</td>
<td>DP-&gt;Air[2] = vac after filling bladder at surface 0.01 inHg</td>
</tr>
<tr>
<td>25-26</td>
<td>DP-&gt;ISRID = i.d. of last interrupt</td>
</tr>
<tr>
<td>27-28</td>
<td>DP-&gt;HPavg! = average pump current at bottom, LSB=1ma</td>
</tr>
<tr>
<td>29-30</td>
<td>DP-&gt;HPmax! = maximum pump current at bottom, LSB=1ma</td>
</tr>
<tr>
<td>31-32</td>
<td>Total seconds pumped to surface</td>
</tr>
<tr>
<td>33-34</td>
<td>Seconds pumped at Surface</td>
</tr>
<tr>
<td>35-36</td>
<td>SPRX = Surf press before resetoffset (pertains to prev dive)</td>
</tr>
<tr>
<td>37-38</td>
<td>SPRXL = press after resetoffset (pertains to prev dive)</td>
</tr>
<tr>
<td>39-41</td>
<td>diagP[0] = Press when “in water” sensed</td>
</tr>
<tr>
<td>42-44</td>
<td>diagT[0] = Temp when “in water” sensed</td>
</tr>
<tr>
<td>45-47</td>
<td>diagS[0] = Salinity when “in water” sensed</td>
</tr>
<tr>
<td>48-49</td>
<td>SBnscan = # scans recorded by SBE</td>
</tr>
<tr>
<td></td>
<td>// -1 (0xffff) indicates unable to get scan count from SBE</td>
</tr>
<tr>
<td></td>
<td>// -2 (0xfffe) indicates SBE never started so SBE didn't reset</td>
</tr>
<tr>
<td></td>
<td>// scan count before returning an old value</td>
</tr>
<tr>
<td>50-51</td>
<td>Compacted SBntry,SBstrt,SBstop status (see misspec.h):</td>
</tr>
<tr>
<td></td>
<td>(!!(DP-&gt;SBntry&amp;0xf)&lt;&lt;4)</td>
</tr>
<tr>
<td>52-54</td>
<td>diagP[1] = Shallowest press in profile</td>
</tr>
<tr>
<td>55-57</td>
<td>diagT[1] = Shallowest Temp in profile</td>
</tr>
<tr>
<td>58-60</td>
<td>diagS[1] = Shallowest Salinity in profile</td>
</tr>
<tr>
<td>61-62</td>
<td>BTvac = BIT vacuum in 0.01 inHg</td>
</tr>
<tr>
<td>63-64</td>
<td>BTPcur = BIT motor current OUT, LSB=1mA</td>
</tr>
<tr>
<td>65-66</td>
<td>BTPsec = BIT Pump seconds</td>
</tr>
<tr>
<td>67</td>
<td>BTPvac[0] = BIT Pump vacuum at beginning of test, before pumping</td>
</tr>
<tr>
<td>68</td>
<td>BTPvac[1] = BIT Pump vacuum after pumping</td>
</tr>
<tr>
<td>69-70</td>
<td>BTVple = BIT pump batt 0.01V</td>
</tr>
<tr>
<td>71-72</td>
<td>BTVcpu= BIT CPU batt 0.01V</td>
</tr>
<tr>
<td>73-74</td>
<td>exception flags</td>
</tr>
</tbody>
</table>
vent data; MSB=#0.1 seconds vent motor ran
LSB LLD status before/after vent ran

AbtCd = code for what caused abort_miss

ID=0xe2, Engineering message in normal dive cycle

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xe2</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 102 = 0x66</td>
</tr>
<tr>
<td>3</td>
<td>Engineering message version = 5</td>
</tr>
<tr>
<td>4</td>
<td>#packets sent in current surface session</td>
</tr>
<tr>
<td>5-6</td>
<td>#tries to connect in previous surface session</td>
</tr>
<tr>
<td>7-8</td>
<td>parse_X_reply</td>
</tr>
</tbody>
</table>

7-8: low order byte number of messages: upper byte bit field of errors

9-10 | ATSBD return status in previous surface session |
11-12 | EP->sattime Seconds taken in previous surface session to send all SBD messages |
13-14 | DP->Vcpu = CPU battery voltage counts 0.01V |
15-16 | DP->Vpmp = Pump battery counts at surface(0.01V) |
17-18 | DP->Vple = Pump battery counts at end of last pump(0.01V) |
19-20 | DP->Air[0] = pcase vac during sinking @50db with oil all inside pcase, 0.01 inHg |
21-22 | DP->Air[1] = pcase vac before filling oil bladder at surface 0.01 inHg |
23-24 | DP->Air[2] = pcase vac after filling bladder at surface 0.01 inHg |
25-26 | DP->ISRId = i.d. of last interrupt |
27-28 | DP->HPavg[1] = average pump current at bottom, LSB=1mA |
29-30 | DP->HPmax[1] = maximum pump current at bottom, LSB=1mA |
31-32 | Total seconds pumped to surface |
33-34 | Seconds pumped at Surface |
35-36 | SPRX = Surf press before resetoffset (pertains to prev dive) |
37-38 | SPRXL = press after resetoffset (pertains to prev dive) |
39-41 | diagP[0] = Pressure before pumping for ascent |
42-44 | diagT[0] = Temp before pumping for ascent |
45-47 | diagS[0] = Salinity before pumping for ascent |
48-50 | diagP[1] = Last (shallowest) Pressure scan on ascent |
51-53 | diagT[1] = Last (shallowest) Temperature scan on ascent |
54-56 | diagS[1] = Last (shallowest) Salinity scan on ascent |
57-58 | SBnbad = # bad bins from SBE |
59-60 | SBnsncan = # scans recorded by SBE |

61-62 | Compacted SBntrly,SBstrt,SBsSTOP status (see misspec.h): ((DP->SBntry&0xf)<<4) | ((DP->SBstrt&0x3)<<2) | (DP->SBsSTOP&0x3) |
63-65 | DP->PAVG[0]=average pressure over first half of DRIFT |
66-68 | DP->TAVG[0]=average temperature over first half of DRIFT |
69-71 | DP->SAVG[0]=average salinity over first half of DRIFT |
72-74 | DP->PAVG[1]=average pressure over second half of DRIFT |
75-77 | DP->TAVG[1]=average temperature over second half of DRIFT |
78-80 | DP->SAVG[1]=average salinity over second half of DRIFT |
81-82 | DP->fall_time = seconds from open air valve to end of settle |
83-84 | DP->fall_rate = avg mm/sec while sinking |
85-86 | DP->SeekT = seconds pumped in 1st settle to drift |
87-88 | DP->SeekP = change of depth (signed 0.1 dbar in 1st settle) |
89-90 | exception flags (see table) |
91   | vent data; # 0.1 seconds vent motor ran |
92   | vent data; LLD status before and after vent ran |
93-94 | SBE P offset(*800) |
95-96 | PP->SeekSc; tenths of seconds pumped to target depth |
97-98  Number of Packets sent in previous cycle
99  Ice-detect status [ICE_CHECK_OFF=0; ICE_NOT_FOUND=1;ICE_MIXLAYER=2, ICE_BREAKUP=3];
100  Compacted Binning Mode (upper 5 bits), Subcycle number (lower 3 bits)
    BinMod options:
    0: Binned by SBE, curvature packing
    2: Binned by controller (float), curvature packing
    16: Binned by SBE, difference packing
    18: Binned by controller (float), difference packing

ID=0xe3, Engineering message following mission abort

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xe3</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 30 = 0x1e</td>
</tr>
<tr>
<td>3</td>
<td>Engineering message version = 5</td>
</tr>
<tr>
<td>4</td>
<td>#packets sent in current surface session</td>
</tr>
<tr>
<td>5-6</td>
<td>#tries to connect in last surface session</td>
</tr>
<tr>
<td>7-8</td>
<td>parse_X_reply:</td>
</tr>
<tr>
<td></td>
<td>low order byte number of messages: upper byte bit field of errors</td>
</tr>
<tr>
<td>9-10</td>
<td>ATSBD return status in last surface session</td>
</tr>
<tr>
<td>11-12</td>
<td>Seconds taken in sending last SBD message</td>
</tr>
<tr>
<td>13-14</td>
<td>current CPU battery voltage counts 0.01V</td>
</tr>
<tr>
<td>15-16</td>
<td>current pump battery counts 0.01V</td>
</tr>
<tr>
<td>17-18</td>
<td>DP-&gt;Air[1] = pcase vacuum at beginning of abort 0.01inHg</td>
</tr>
<tr>
<td>19-20</td>
<td>DP-&gt;Air[0] = pcase vacuum at end of last xmit (previous cycle) 0.01 inHg</td>
</tr>
<tr>
<td>23-24</td>
<td>DP-&gt;ISRID = i.d. of last interrupt</td>
</tr>
<tr>
<td>25-26</td>
<td>AbrtCd = code for what caused abort_miss</td>
</tr>
<tr>
<td></td>
<td>0 = no error</td>
</tr>
<tr>
<td></td>
<td>1 = current time is later than RTCabott</td>
</tr>
<tr>
<td></td>
<td>2 = BAD_P (unable to WakeOST)</td>
</tr>
<tr>
<td></td>
<td>3 = BAD_XMITS (unable to send Dive number to SOLO II)</td>
</tr>
<tr>
<td></td>
<td>4 = SHORE_CMD (Iridium ground station commanded to go to abort)</td>
</tr>
<tr>
<td></td>
<td>5 = FINAL_DIVE (FnlDiv was completed. Mission is done)</td>
</tr>
<tr>
<td></td>
<td>6 = BAD_PHASE (Diagnostic dive failed to get GPS fix, pressure never&gt;dBarGo, or unable to send message to Iridium)</td>
</tr>
<tr>
<td></td>
<td>7 = CANT_SURFACE (pressure sensor failure)</td>
</tr>
<tr>
<td></td>
<td>8 = BAD_XMITS (while in abort mode)</td>
</tr>
<tr>
<td></td>
<td>9 = TIMED_OUT_WAITING (time out before diagnostic dive)</td>
</tr>
</tbody>
</table>

27-28  Empty
29  ; terminator

ID=0xe5, Engineering message following BITest

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xe5</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 58 = 0x3a</td>
</tr>
<tr>
<td>3</td>
<td>Engineering message version =5</td>
</tr>
<tr>
<td>4</td>
<td>#packets sent in this surface session</td>
</tr>
<tr>
<td>5-6</td>
<td>SBE P Offset(*800) [Fixed in SBE602 22Jul14, invalid before]</td>
</tr>
<tr>
<td>7-8</td>
<td>CPU battery voltage 0.01 V</td>
</tr>
<tr>
<td>9-10</td>
<td>no load pump battery voltage 0.01 V</td>
</tr>
<tr>
<td>11-12</td>
<td>pump battery voltage counts at end of last pump (0.01V)</td>
</tr>
</tbody>
</table>
13-14  DP->HPavgl = average pump current at bottom, LSB=1ma
15-16  seconds pumped out during test
17    Oil vacuum before filling bladder 0.01inHG
18    Oil vacuum after filling bladder 0.01 inHg
19-20  DP -> Air[0] = Pcase Vacuum at beginning of BIT. (Oil Bladder Empty) 0.01 inHg
21-22  DP → Air[1] = Pcase Vacuum at end of BIT with air bladder inflated. 0.01 inHg
23    Number of tries needed to open valve
24    Number of tries to close valve
25-26  i.d. of last interrupt
27-56  string returned from SBE pt command
57    ; terminator

**ID=0xe6, Engineering message following Failed BITest**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xe6c</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 60 = 0x3c</td>
</tr>
<tr>
<td>3</td>
<td>Engineering message version =5</td>
</tr>
<tr>
<td>4</td>
<td>#packets sent in this surface session</td>
</tr>
<tr>
<td>5-6</td>
<td>BitTest status register</td>
</tr>
<tr>
<td>7-8</td>
<td>SBE P Offset(*800)</td>
</tr>
<tr>
<td>9-10</td>
<td>CPU battery voltage 0.01 V</td>
</tr>
<tr>
<td>11-12</td>
<td>no load pump battery voltage 0.01 V</td>
</tr>
<tr>
<td>13-14</td>
<td>pump battery voltage counts at end of last pump (0.01V)</td>
</tr>
<tr>
<td>15-16</td>
<td>DP-&gt;HPavgl = average pump current at bottom, LSB=1ma</td>
</tr>
<tr>
<td>17-18</td>
<td>seconds pumped out during test</td>
</tr>
<tr>
<td>19</td>
<td>Oil vacuum before filling bladder 0.01inHG</td>
</tr>
<tr>
<td>20</td>
<td>Oil vacuum after filling bladder 0.01 inHg</td>
</tr>
<tr>
<td>21-22</td>
<td>DP -&gt; Air[0] = Pcase Vacuum at beginning of BIT. (Oil Bladder Empty) 0.01 inHg</td>
</tr>
<tr>
<td>23-24</td>
<td>DP → Air[1] = Pcase Vacuum at end of BIT with air bladder inflated. 0.01 inHg</td>
</tr>
<tr>
<td>25</td>
<td>Number of tries needed to open valve</td>
</tr>
<tr>
<td>26</td>
<td>Number of tries to close valve</td>
</tr>
<tr>
<td>27-28</td>
<td>i.d. of last interrupt</td>
</tr>
<tr>
<td>29-58</td>
<td>string returned from SBE pt command</td>
</tr>
<tr>
<td>59</td>
<td>; terminator</td>
</tr>
</tbody>
</table>

**Mission EEPROM dump (ID=0xdn)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xd0,0xd1,0xd2,0xd3 [Possible values 0:d]</td>
</tr>
<tr>
<td>1-2</td>
<td>len=Number of bytes (variable, see below)</td>
</tr>
<tr>
<td>3- (len-2)</td>
<td>ASCII listing of mission parameters</td>
</tr>
<tr>
<td></td>
<td>Each EEPROM parameter has a 6 character name and 5 char value:</td>
</tr>
<tr>
<td></td>
<td>NAMExx=vvvvv</td>
</tr>
<tr>
<td></td>
<td>The = &amp;</td>
</tr>
<tr>
<td></td>
<td>Successive parameters follow without gaps.</td>
</tr>
<tr>
<td>len-1</td>
<td>; terminator at the end of the dump</td>
</tr>
</tbody>
</table>

An example showing only the initial 3 and final 2 elements follows:

PROup = 1| BLOK= 1| PB1= 10[... | XPdly= 0| FstSrf= 0];

The EEPROM dump message is sent only in response to a command "P" from the ground station. It is sent over 4 SBD messages (0xD0=328 bytes, 0xD1=328 bytes, 0xD2=328 bytes, 0xD3=72 bytes).
**Float Echo (ID=0xde)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xde</td>
</tr>
<tr>
<td>1-2</td>
<td>len=Number of bytes (includes ID and ;)</td>
</tr>
<tr>
<td>3- (len-2)</td>
<td>ASCII string</td>
</tr>
<tr>
<td>len-1</td>
<td>; terminator at the end of the echo</td>
</tr>
</tbody>
</table>

Float responds to commands. Within the Echo, the returned ascii string is followed by a ‘;’ if the command was NOT accepted by the float. and if followed by a ‘;‘ then the command was accepted and should change float behavior.

**Argo Data ID=0xf0** Relayed in normal cycles

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xf0</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = 37 = 0x25</td>
</tr>
<tr>
<td>3</td>
<td>Data Version (Minor version in high order nibble, major version in low order)</td>
</tr>
<tr>
<td>4-5</td>
<td>Target profile depth</td>
</tr>
<tr>
<td>6-7</td>
<td>Target parking depth</td>
</tr>
<tr>
<td>8-9</td>
<td>Maximum rise time in minutes</td>
</tr>
<tr>
<td>10-11</td>
<td>Target (maximum) fall to parking depth time in minutes</td>
</tr>
<tr>
<td>12-13</td>
<td>Maximum fall-from-parking-to-profile-depth time in second</td>
</tr>
<tr>
<td>14-15</td>
<td>Target drift time (units=5 minutes; To convert to minutes multiply packed value by 5)</td>
</tr>
<tr>
<td>16</td>
<td>Float version (0 SOLOII)</td>
</tr>
<tr>
<td>17</td>
<td>Target ascent rate while profiling</td>
</tr>
<tr>
<td>18-19</td>
<td>Number of seeks</td>
</tr>
<tr>
<td>20-21</td>
<td>Surface Time</td>
</tr>
<tr>
<td>22-23</td>
<td>Seek Interval in minutes</td>
</tr>
<tr>
<td>24-25</td>
<td>Pressure scaling gain; db=counts/gain-offset</td>
</tr>
<tr>
<td>26-27</td>
<td>Pressure scaling offset</td>
</tr>
<tr>
<td>28-29</td>
<td>Temperature scaling gain; degreesC=counts/gain-offset</td>
</tr>
<tr>
<td>30-31</td>
<td>Temperature scaling offset</td>
</tr>
<tr>
<td>32-33</td>
<td>Salinity scaling gain; PSU=counts/gain-offset</td>
</tr>
<tr>
<td>34-35</td>
<td>Salinity scaling offset</td>
</tr>
<tr>
<td>36</td>
<td>; terminator</td>
</tr>
</tbody>
</table>

The values of Gain/Offset are now sent back within the Argo Metafile message (0xf0) for data decoding purposes allowing a way to determine what Gain/offset is used in a given cycle. The GAIN/OFFSET of Temperature/Salinity/Pressure can be modified via 2-way communication. Modifying these parameters will effect all variables returned.

Float Version= SOLOII/S2A 0; Deep SOLO/S2A 1; ALAMO 3

**Test Data (ID=0xf1)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ID/Mission phase = 0xf1</td>
</tr>
<tr>
<td>1-2</td>
<td>Number of bytes = variable</td>
</tr>
<tr>
<td>3</td>
<td>modulo</td>
</tr>
<tr>
<td>4-n</td>
<td>test data</td>
</tr>
</tbody>
</table>
Exception Flag (Engineering Message) Table  [Value sent by float can be sum from multiple errors]

<table>
<thead>
<tr>
<th>Hex</th>
<th>Value</th>
<th>Description</th>
<th>Mission</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>1</td>
<td>Valve failed to open (no okay return value received)</td>
<td>Mission</td>
</tr>
<tr>
<td>0x0002</td>
<td>2</td>
<td>Valve failed to close (no okay return value received)</td>
<td>Mission</td>
</tr>
<tr>
<td>0x0004</td>
<td>4</td>
<td>Questionable pressure (float times out sink but no pump at 100dbar)</td>
<td>Fall</td>
</tr>
<tr>
<td>0x0008</td>
<td>8</td>
<td>Reset Antenna (toggled, no GPS satellite after 1 minute)</td>
<td>Surface</td>
</tr>
<tr>
<td>0x0010</td>
<td>16</td>
<td>Antenna switch failure (2nd toggle, no GPS satellite after 1 minute)</td>
<td>Surface</td>
</tr>
<tr>
<td>0x0020</td>
<td>32</td>
<td>GPS communication error: No GPS</td>
<td>Surface</td>
</tr>
<tr>
<td>0x0040</td>
<td>64</td>
<td>SBE_BIN_ERROR (no bins are available from SBE)</td>
<td>Rise</td>
</tr>
<tr>
<td>0x0080</td>
<td>128</td>
<td>Float didn’t leave (returned to) the surface (10 min (halved Tlast)</td>
<td>Surface</td>
</tr>
<tr>
<td>0x0100</td>
<td>256</td>
<td>Restarted profile (stalled &gt; 10dbar)</td>
<td>Rise</td>
</tr>
<tr>
<td>0x0200</td>
<td>512</td>
<td>BAD_PRES (3 or more bad pressure readings from SBE)</td>
<td>Rise</td>
</tr>
<tr>
<td>0x0400</td>
<td>1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0800</td>
<td>2048</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1000</td>
<td>4096</td>
<td>Valve failure during Sink phase of mission</td>
<td></td>
</tr>
<tr>
<td>0x2000</td>
<td>8192</td>
<td>Valve failure during Ascend phase of mission</td>
<td></td>
</tr>
<tr>
<td>0x4000</td>
<td>16384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8000</td>
<td>32768</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Surface Interval: Normal versus Fast

The CONFIG FastSrf decides the structure/order of surface interval events

If FstSrf=1 float logic follows as such...

- Create drift messages.
- Turn on GPS.
- Start CTD binning
- Pump MnSfp
- finish GPS fix.
- If no fix in 1/2 GPSsec,
  - Pump (MxSfP-MnSfP)
  - Get GPS fix.
- Wait for CTD binning to end.
- Read CTD binned profile
- Create profile messages.
- Read Raw data
- Create raw data messages
- Start CTD pressure zeroing
- Create rise, fall, pump, and engineering messages.
- Send the data.
- Finish CTD pressure zeroing
- Start next cycle with a GPS fix.

NOTE: In FstSrf mode, the final rise pressure value (0x050 msg) will take place AFTER the first GPS fix. This is a modification from the traditional ordering of the surface events.

If FstSrf = 0 (same as all previous float versions) float logic follows as such...

- Create drift messages.
- Pump for MxSIP secs.
- Start CTD binning
- Get GPS fix.
- Wait for CTD binning to end.
- Read CTD binned profile
- Create profile messages.
- Read Raw data
- Create raw data messages
- Create rise, fall, pump, and engineering messages.
- Send the data.
- Tell CTD to zero the pressure.
- Wait 2 minutes for CTD
- Start next cycle with a GPS fix.