Format of SOLOII X messages: Argo Version Manual/Decoder V0.7
latest update: 09 Apr 2021
[For ROM RBR 602 V0.7; IDG SN 8851]

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:

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 delimitor 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 0xf -> 0x3f = '?'.
>	=	a > delimitor 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 (relative to previous version V2.6)
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

BUGS:

The <data> section contains information from multiple sensors. Data from successive sensors are separated by a semicolon ('**;**' = 0x 3b); the final sensor is terminated by a '**;**' (immediately preceding the **$** delimitor).

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.
;	=	delimitor 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.

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

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:

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

Pressure data (ID=0x1n)
Temperature data (ID=0x2n)
Salinity data (ID=0x3n)
TEMP_CNDC (ID=0xan, n=0-7)

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 and later)

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:

IDjj<sub-block 0><sub-block 1> . . . <sub-block m>**;**

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…
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 ;.
<sub-block i>	=	first-differenced data from the ith sub=block where i=1,..,m =number of sub-blocks. If i<m, 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],...v[n-1]. Then this sub-block will be transmitted as:

Sub-block Byte	**Contents**
0	one-byte scaling factor S, range = 1 to 255. S is chosen so that the scaled first-differences fit in one byte, i.e. |diff| <= 127.
1	MS byte of v[0]
2	LS byte of v[0]
3	LS byte of { v[1] - v[0] }/S
4	LS byte of { v[2] - v[1] }/S
. . .	
n+1	LS byte of { v[n-1] - v[n] }/S

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 alloted for the data. The bytes alloted will be constant within a 16 value sub-block, but will differ between parameters and between sub-blocks of the same parameter.

IDjjBNNVVVDDDpppppppppppp<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=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=TopIndx+1] - v[n=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 be 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:

Sub-block Byte	Contents
0	v[2+i*16] – 2*v[1+i*16] + v[i*16]
1	v[3+i*16] – 2*v[2+i*16] + v[1+i*16]
....	
n	v[n+2+i*16] – 2*v[n+1+i*16] + v[n+i*16]

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 communcation.
Modifying these parameters will effect all variables returned.

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'.

<u>Fall Rate data (ID=0x4n)</u>

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 (**FallIn**) 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.

Maximum of 240 Fall values can be transmitted in a given cycle (Fixed?)

Pump data (ID=0x6n)

The data message looks like:

IDjj< depth(1),time(1),voltage(1),current(1),vac0(1),vac1(1)> . . .
< depth(m),time(m),voltage(m),current(m),vac0(m),vac1(m)**;**

ID	=	one-byte sensor ID code = 0x6n.
jj	=	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 **;.**
code(i)	=	Code representing float phase in 1 nibble, i=1, …, m (See Fall for values).
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
time(i)	=	seconds the pump ran in 2 bytes (signed)
voltagei)	=	average pump battery counts while pumping in 2 bytes (0.01V)
current(i)	=	average pump current at bottom in 2 bytes, LSB=1ma
vac0(i)	=	vacuum counts after pump starts in 1 byte
vac1(i)	=	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.

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.

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

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

Byte	Contents
0	ID/Mission phase = 0xe0
1-2	Number of bytes = 82= 0x52
3	Engineering message version =6
4	#packets in current session
5-10	0 (dummy filler)
11-12	EP -> sattime
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	BTvac = pcase vacuum at beginning of BIT in 0.01 inHg
21-22	DP->Air[1] = vac before filling bladder at surface 0.01 inHg
23-24	DP->Air[2] = vac after filling bladder at surface 0.01 inHg
25-26	DP->ISRID = i.d. of last interrupt
27-28	DP->HPavgI = average pump current at bottom, LSB=1ma
29-30	DP->HPmaxI = 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] = Press when "in water" sensed
42-44	diagT[0] = Temp when "in water" sensed
45-47	diagS[0] = Salinity when "in water" sensed
48-49	SBnscan = # scans recorded by SBE // -1 (0xffff) indicates unable to get scan count from SBE // -2 (0xfffe) indicates SBE never started so SBE didn't reset // scan count before returning an old value
50-51	Compacted SBntry,SBstrt,SBstop status (see misspec.h): ((DP->SBntry&0xf)<<4) | ((DP->SBstrt&0x3)<<2) | (DP->SBstop&0x3))
52-54	diagP[1] = Shallowest press in profile
55-57	diagT[1] = Shallowest Temp in profile
58-60	diagS[1] = Shallowest Salinity in profile
61-62	BTvac = BIT vacuum in 0.01 inHg
63-64	BTPcur = BIT motor current OUT, LSB=1mA
65-66	BTPsec = BIT Pump seconds
67	BTPvac[0] = BIT Pump vacuum at beginning of test, before pumping
68	BTPvac[1] = BIT Pump vacuum after pumping
69-70	BTVple = BIT pump batt 0.01V
71-72	BTVcpu= BIT CPU batt 0.01V
73-74	exception flags

75	vent data; MSB=#0.1 seconds vent motor ran
76	LSB LLD status before/after vent ran
77-78	AbrtCd = code for what caused abort_miss
79	CPU board temperature at start of ascent (= (T+2) *8)
80	Relative Humidity at start of ascent (%)
81	**;** terminator

ID=0xe2, Engineering message in normal dive cycle

Byte	Contents
0	ID/Mission phase = 0xe2
1-2	Number of bytes = 104 = 0x68
3	Engineering message version = 6
4	#packets sent in current surface session
5-6	#tries to connect in previous surface session
7-8	parse_X_reply 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->HPavgI = average pump current at bottom, LSB=1ma
29-30	DP->HPmaxI = 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	SBnscan = # scans recorded by SBE // -1 (0xffff) indicates unable to get scan count from SBE // -2 (0xfffe) indicates SBE never started so SBE didn't reset // scan count before returning an old value
61-62	Compacted SBntry,SBstrt,SBstop status (see misspec.h): ((DP->SBntry&0xf)<<4) **|** ((DP->SBstrt&0x3)<<2) **|** (DP->SBstop&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-dectect 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
101	CPU board temperature at start of ascent (= (T+2) *8)
102	Relative Humidity at start of ascent (%)
103	**;** terminator

ID=0xe3, Engineering message following mission abort

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

ID=0xe5, Engineering message following BITest

Byte	Contents
0	ID/Mission phase = 0xe5
1-2	Number of bytes = 60 = 0x3c
3	Engineering message version =6
4	#packets sent in this surface session
5-6	SBE P Offset(*800)
7-8	CPU battery voltage 0.01 V
9-10	no load pump battery voltage 0. 01 V
11-12	pump battery voltage counts at end of last pump (0.01V)
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	CPU board temperature (= (T+2) *8)
58	Relative Humidity (%)
59	**;** terminator

ID=0xe6, Engineering message following Failed BITest

Byte	Contents
0	ID/Mission phase = 0xe6
1-2	Number of bytes = 62 = 0x3c
3	Engineering message version =6
4	#packets sent in this surface session
5-6	BITest status register
7-8	SBE P Offset(*800)
9-10	CPU battery voltage 0.01 V
11-12	no load pump battery voltage 0. 01 V
13-14	pump battery voltage counts at end of last pump (0.01V)
15-16	DP->HPavgl = average pump current at bottom, LSB=1ma
17-18	seconds pumped out during test
19	Oil vacuum before filling bladder 0.01inHG
20	Oil vacuum after filling bladder 0.01 inHG
21-22	DP-> Air[0] = Pcase Vacuum at beginning of BIT. (Oil Bladder Empty) 0.01 inHg
23-24	DP → Air[1] = Pcase Vacuum at end of BIT with air bladder inflated. 0.01 inHg
25	Number of tries needed to open valve
26	Number of tries to close valve
27-28	i.d. of last interrupt
29-58	string returned from SBE pt command
59	CPU board temperature (= (T+2) *8)
60	Relative Humidity (%)
61	**;** terminator

Mission EEPROM dump (ID=0xdn)

Byte	Contents
0	ID/Mission phase = 0xd0,0xd1,0xd2,0xd3 [Possible values 0:d]
1-2	len=Number of bytes (variable, see below)
3- (len-2)	ASCII listing of mission parameters Each EEPROM parameter has a 6 character name and 5 char value: NAMExx**=**vvvvv **|** The **=** & **|** signs are present in the listing of each parameter. (13 bytes/parameter) Successive parameters follow without gaps.
len-1	**;** terminator at the end of the dump

An example showing only the initial 3 and final 2 elements follows:
PROup = 1| Nregms=   3| Zbin0=   10|...| FstSrf=   0| Z1Pump= 100| ;

The EEPROM dump message is sent in cycle -1 and in response to a command "**P**" from the ground station. It is sent over 4 SBD messages (0xd0=316 bytes, 0xd1=316 bytes, 0xd2=316 bytes, 0xd3=238 bytes).

Float Echo (ID=0xde)

Byte	Contents
0	ID/Mission phase = 0xde
1-2	len=Number of bytes (includes ID and ;)
3- (len-2)	ASCII string
len-1	; terminator at the end of the echo

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.

Argo Data ID=0xf0 Relayed in normal cycles

Byte	Contents
0	ID/Mission phase = 0xf0
1-2	Number of bytes = 37 = 0x25
3	Data Version (Minor version in high order nibble, major version in low order)
4-5	Target profile depth
6-7	Target parking depth
8-9	Maximum rise time in minutes
10-11	Target (maximum) fall to parking depth time in minutes
12-13	Maximum fall-from-parking-to-profile-depth time in second
14-15	Target drift time (units=5 minutes; To convert to minutes multiply packed value by 5)
16	Float version (0 SOLOII/S2A, 1 Deep SOLO; 3 Alamo)
17	Target ascent rate while profiling
18-19	Number of seeks
20-21	Surface Time
22-23	Seek Interval in minutes
24-25	Pressure scaling gain; db=counts/gain-offset
26-27	Pressure scaling offset
28-29	Temperature scaling gain; degreesC=counts/gain-offset
30-31	Temperature scaling offset
32-33	Salinity scaling gain; PSU=counts/gain-offset
34-35	Salinity scaling offset
36	**;** terminator

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 communcation. Modifying these parameters will effect all variables returned.

Test Data (ID=0xf1)

Byte	Contents
0	ID/Mission phase = 0xf1
1-2	Number of bytes = variable
3	modulo
4-n	test data

Exception Flag (Engineering Message) Table [Value sent by float can be sum from multiple errors]

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

SndBak Mode

SndBak=0	Return no data; CONFIG and Echo still available via 2-way
SndBak=1	Return GPS, Engineering, and Argo messages
SndBak=2	Return High Resolution data (0x9X, 0xAX, 0xBX where X<8)
SndBak=4	Return Rise data (0x5X)
SndBak=8	Return Fall data (0x4X)
SndBak=16	Return Pump data (0x6X)
SndBak=32	Return Drift data (0x9X, 0xAX, 0xBX where X>=8)
SndBak=64	Return Binned profile pressure data (0x1X)
SndBak=128	Return Binned profile temperature data (0x2X)
SndBak=256	Return Binned profile salinity data (0x3X)

SndBak=511 will return everything (default)
The 0xdX (CONFIG dump and Echo) are unaffected by SndBak;

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 MxSfP 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.
```