Format of Deep SOLO X messages: Argo Version Manual/Decoder V0.4
latest update: 27 Oct 2017
(ROM 01Jul2015)

An X message is used to transfer data from ISU to GS 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 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 0xf -> 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 (previous DEEP float version = v0.3)
Fields that are moved relative to the previous DEEP float version are highlighted in cyan
New fields relative to the DEEP previous version are highlighted in yellow

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 **$** 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.
;	=	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. For a 1000 sample profile, there will be 6 messages for each of the pressure, salinity and temperature sensors.

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 (scaled 1st difference) x=0-7: upper ocean bin averaged data x=8-F: deep spot sampled data
Temperature	2x	depth series of temperature (scaled 1st difference) x=0-7: upper ocean bin averaged data x=8-F: deep spot sampled data
Salinity	3x	depth series of salinity (scaled 1st difference) x=0-7: upper ocean bin averaged data x=8-F: deep spot sampled data
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
Time	7x	depth series of time-referenced to 0x40 (scaled 1st difference): x=8-F: Only returned during deep spot sampling
High Resolution Pressure	9x	High Resolution Pressure (scaled 1st difference)
High Resolution Temperature	ax	High Resolution Temperature (scaled 1st difference)
High Resolution Salinity	bx	High Resolution Salinity (scaled 1st difference)
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	d0	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
Engineering	e6	engineering data BIT test fail
EEPROM dump	d0	Float configuration dump
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)
Time data (ID=0x7n)

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 SOLO-D ascends/descends and stores the values internally. When SOLO-D reaches the surface/park pressure, it takes the continuously sampled data (0-2000dbar) data from the CTD and block averages it in depth into PRO_BINS (= 1000) bins. Data sampled deeper than 2000dbar is recorded in spot sampled mode and returned in different messages. Time of the spot sampled data is also returned.

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 and discrete) profile data. Which ever packing is requested is returned within the data stream within the first nibble of the jj variable (see below).

1. Difference Packing (Standard to versions previous to V0.4, Optional in V0.4)

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

2. Curvature Packing (New to V0.4 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=1, for succeeding messages m > 0, B > 1. The position (n) of the first value recorded in a message (VVV) can be computed as TopIndx = m + (B-1) *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.

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 0. Using 12 bytes for the packing factors, there can be up to 32 sub-blocks, or 512 values if the packing factor is 0 (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 – 1) * 16 through n = 1 + m + (B - 1 + 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 = 1 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 affect 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)

SOLOII/SOLO-D has the ability to return a high resolution P,T,S profile spanning a subsection of the primary binned profile (upper 2000dbar). Data is packed and decoded similarly to the binned profile (ID=0x10, 0x20, 0x30). 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: When the High Resolution data is requested, the averaging of the primary binned profile must be done by the float (not within the CTD). Typical SOLOII/SOLO-D 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. High Resolution profile data is decoupled from BinMod and is set to always use 'difference packing'.

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

The float can be set to return a time-series 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 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, n=0:f)</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 = 0x40.

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 fall (seconds since 1Jan2000) in 4 bytes (MSB first).

time(i) = time since start_time in 2 bytes, i=1, ..., m, (resolution of 10 seconds;)

code(i) = Code representing float phase while data value recorded in 1 nibble, i=1, …, m.

Possible Phase codes values

Last of Continuous profile records	=0,
START_OF_SINK	=1,
Buoyancy at 200db	=2,
SEEK	=3,
BEGINNING_OF_DRIFT	=4,
SEEK_DURING_DRIFT	=5,
END_OF_SINK	=6,
START_OF_RISE	=7,
END_OF_RISE	=8,
TURN_AROUND	=9,
SINKING	=10,
DRIFTING	=11,
RISING	=13,
SURFACE	=14 ;

time resolution =10 s/count

depth(i) = depth (LSB=0.1 db) at time(i) in 2.5 bytes, i=1, ..., m.
dBar = 0.1 * depth(i) -10
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. The SOLO-D logs values every <SkSLsc> s during the continuous-profile. Others are recorded at: the same time as the first spot sample; at the end of sink(); at the beginning of each seek(); and at the beginning of park().

Fall Rate data can be found over multiple messages.

<u>Rise Rate data (ID=0x5n, n=0:f)</u>

The rise rate message is identical in structure to the fall rate message. For the SOLOII, the rise rate time series begins as its valve is opened to descend 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. Rise Rate data can be found over multiple messages.

For the SOLO-D the rise rate time series begins at the end of park() (start of ascend()), sampling periodically (<AsSLsc>), until the surface is reached. time resolution =10 s/count.

Pump data (ID=0x6n, n=0:f)

. 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 = 0x60.
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.
code(i) = Code representing float phase in 1 nibble, i=1, …, m (See Fall for values).
depth(i) = depth (LSB=0.1 db) at time(i) in 2.5 bytes, i=1, ..., m.
dBar = .1 * depth(i) -10. This is the depth when the pump STARTED.
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
t_mn(i) = (2 bytes) time[minutes] since start-of-fall that this pump event STOPPED.

Pump time series can be found over multiple messages.

A bug in the software of V0.4 limits the pump message to a single message with ID 0x60. This message will be the latest message data. So for instance if the float should have packed a 0x60 with 25 data, and a 0x61 with 11 data. The float will transmit only the 11 data and label it as 0x60. In addition, if a message has exactly 25 data, it will not be sent at all. This is a fun bug.

Engineering data (ID=0xe0, 0xe2, 0xe3, 0xe5, 0xe6)

The engineering data is used to diagnose SOLOII/SOLO-D anomalies. A different format is used in each of the distinct phases of a SOLOII/SOLO-D 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 = BITfailed
1-2	Number of bytes in the message, depends on mission phase as described below
3 -> ??	Depends on mission phase as described below

ALL ID's have the same first four bytes:

Byte	**Contents**
0	ID/Mission phase = 0xe***n*** (***n***=0,2, 3, 5, 6)
1-2	Number of bytes = (0xe0 = ; 0xe2 = ; 0xe3 = 0x1a; 0xe5 = 0x3a; 0xe6= 0x3c)
3	Engineering message version =3
4	#packets to send in the current session.

Instead of byte position below, the parameter type is given:
char, uchar = 1-byte field (char = signed, uchar = unsigned).
short, ushort = 2-byte field (short = signed, ushort = unsigned).
string[n] = string of bytes, length n.

ID=0xe6, Engineering message following FailedBITest

type	**Contents**
	(starts with same 4 eng. header bytes).
ushort	BITstatus (failure status) (first two bytes are the bit-fail status)
	this is followed by the same parameters for a successful BITtest (0xe5):

ID=0xe5, Engineering message following a successful BITest

type	**Contents**
	(starts with same 4 eng. header bytes).
short	SBE P Offset(*800)
short	CPU battery voltage 0.01 V
short	no load pump battery voltage 0. 01 V
short	pump battery voltage counts at end of last pump (0.01V)
short	DP->HPavgl = average pump current at bottom, LSB=1ma
short	seconds pumped out during test
uchar	Oil sensor before filling bladder [0..255 counts]
uchar	Oil sensor after filling bladder [0..255 counts]
short	DP-> Air[0] = Pcase Vacuum at beginning of BIT. (Oil Bladder Empty) 0.01 inHg
short	DP → Air[1] = Pcase Vacuum at end of BIT with air bladder inflated. 0.01 inHg
uchar	Number of tries needed to open valve
uchar	Number of tries to close valve
ushort	i.d. of last interrupt
string[30]	string returned from SBE pt command
char	**;** terminator

ID=0xe0, Engineering message in 1st diagnostic dive at start of mission (Number of bytes = 76 =0x4C)

type	Contents
	(starts with same 4 eng. header bytes).
ushort	#tries to connect in last surface session
ushort	parse_X_reply status in last surface session
ushort	ATSBD return status in last surface session
ushort	Seconds taken in sending last SBD message
ushort	current CPU battery voltage counts 0.01V
ushort	current pump battery counts 0.01V
ushort	Pump battery counts at end of last pump 0.01V.
ushort	DP->Air[0] = pcase vacuum at beginning of BIT 0.01 inHg
ushort	DP->Air[1] = pcase vacuum before bladder full 0.01inHg
ushort	DP->Air[2] = pcase vacuum after bladder full 0.01inHg
ushort	DP->ISRID = i.d. of last interrupt
ushort	HPavgI = avg pump current at bottom, LSB=1 mA.
ushort	HPmaxI = max pump current at bottom, LSB = 1 mA.
ushort	total pump seconds on ascent.
ushort	seconds pumped at the surface.
ushort	DP->P[5] = surf press counts @ end of ASCEND (LSB=.1 dBar)
ushort	SPRX = Surf press before resetoffset (pertains to prev dive)
ushort	SPRXL = press after resetoffset (pertains to prev dive)
ushort	diagP[0] = Press when "in water" sensed
ushort	diagT[0] = Temp when "in water" sensed
ushort	diagS[0] = Salinity when "in water" sensed
short	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
ushort	Compacted SBntry,SBstrt,SBstop status (see misspec.h):
	((DP->SBntry&0xf)<<4) | ((DP->SBstrt&0x3)<<2) | (DP->SBstop&0x3))
ushort	diagP[1] = Shallowest press in profile (not filled)
ushort	diagT[1] = Shallowest Temp in profile (not filled)
ushort	diagS[1] = Shallowest Salinity in profile (not filled)
ushort	BTvac = BIT vacuum in 0.01 inHg
ushort	BTPcur = BIT motor current, LSB=1mA
ushort	BTPsec = BIT Pump seconds
uchar	BTPvac[0] = BIT oil sensor at beginning of test, before pumping
uchar	BTPvac[1] = BIT oil sensor after pumping
ushort	BTVple = BIT pump batt 0.01V
ushort	BTVcpu= BIT CPU batt 0.01V
ushort	exception flags
uchar	#0.1 seconds vent motor ran
uchar	LLD status before/after the vent ran.
uchar	AbrtCd = code for what caused the abort_miss.
char	; terminator.

ID=0xe2, Engineering message in normal dive cycle (Number of bytes = 103 =0x67)

type **Contents**
(starts with same 4 eng. header bytes).

ushort #tries to connect in last surface session
ushort parse_X_reply status in last surface session
ushort ATSBD return status in last surface session
ushort Seconds taken in sending last SBD message
ushort present CPU battery voltage counts 0.01V
ushort present pump battery counts 0.01V
ushort Pump battery counts at end of last pump 0.01V.
ushort DP->Air[0] = pcase vac during sinking @50db with oil all inside pcase ,0.01 inHg
ushort DP->Air[1] = pcase vacuum before filling oil bladder at surface 0.01 inHg
ushort DP->Air[2] = pcase vacuum after filling bladder at surface 0.01 inHg
ushort DP->ISRID = i.d. of last interrupt
ushort HPavgI = avg pump current at bottom, LSB=1 mA.
ushort HPmaxI = max pump current at bottom, LSB = 1 mA.
For SOLO-D, HPmaxI=0 as dummy-fill
ushort total pump seconds on ascent.
ushort seconds pumped at the surface.
ushort SPRX = Surf press before resetoffset (pertains to prev dive)
ushort SPRXL = press after resetoffset (pertains to prev dive)
ushort diagP[0] = Pressure before pumping for ascent
ushort diagT[0] = Temp before pumping for ascent
ushort diagS[0] = Salinity before pumping for ascent
ushort diagP[1] = First (shallowest) Pressure scan on descent
ushort diagT[1] = First (shallowest) Temperature scan on descent
ushort diagS[1] = First (shallowest) Salinity scan on descent
ushort SBnbad = # bad bins from SBE
ushort 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
ushort Compacted SBntry,SBstrt,SBstop status (see misspec.h):
((DP->SBntry&0xf)<<4) | ((DP->SBstrt&0x3)<<2) | (DP->SBstop&0x3))
ushort DP->P[0] = press counts before begin of FALL (LSB =.1 dBar)
ushort DP->P[1] = press counts at end of FALL (LSB = .1 dBar)
ushort DP->P[2] = press counts at beginning of DRIFT (LSB = .1 dBar)
ushort DP->P[3] = press counts at end of DRIFT (LSB = .1 dBar)
ushort DP->P[5] = surf press counts @ end of ASCEND (LSB = .1 dBar)
ushort DP->PAVG[0]=average pressure over first half of DRIFT
ushort DP->TAVG[0]=average temperature over first half of DRIFT
ushort DP->SAVG[0]=average salinity over first half of DRIFT
ushort DP->PAVG[1]=average pressure over second half of DRIFT
ushort DP->TAVG[1]=average temperature over second half of DRIFT
ushort DP->SAVG[1]=average salinity over second half of DRIFT
ushort DP->fall_time = seconds from open air valve to end of settle
ushort DP->fall rate = avg mm/sec while sinking
ushort DP-> SeekT = seconds pumped in 1st settle to drift
ushort DP-> SeekP = change of depth (signed 0.1 dbar in 1st settle)
ushort exception flags (can be added)
0x0001 Valve failed to open
0x0002 Valve failed to close
0x0004 Questionable pressure
0x0008 Antenna was toggled

	0x0010	Antenna switch failure. (no satellites even after toggling)
	0x0020	GPS communication error (cannot talk to GPS unit)
	0x0080	Float took too long to leave the surface. (toggled valve)
	0x1000	Valve failure during Sink phase of mission
	0x2000	Valve failure during Ascend phase of mission
uchar	vent data; # 0.1 seconds vent motor ran	
uchar	vent data; LLD status before and after vent ran	
short	SBE P offset(*800)	
ushort	PP->SeekSc; tenths of seconds pumped to target depth	
ushort	Number of Packets sent in previous cycle	
ushort	DP->W_FALL, last fall velocity in previous cycle (mm/s)	
ushort	Z_Neutral = the float's estimated neutrally-buoyant depth (no oil in ext. bladder).	
char	Compacted BinMod and SubCycle number: (first 5 bits BinMod, last 3 bits SubCycle) (BinMod & 0x1f << 3) | (MP->ThisCycle & 0x7); BinMod options: 2: Breck Curvature Compression binned by controller (float) (Always set) 10: Classic Difference Comp. binned by controller (float) for discrete data only 18: Classic Difference Comp. binned by controller (float) for binned data only 26: Classic Diff. Comp. binned by controller (float) for binned and discrete data	
char	**;** terminator	

ID=0xe3, Engineering message following mission abort

type	Contents
	(starts with same 4 eng. header bytes).
ushort	#tries to connect in last surface session
ushort	parse_X_reply status in last surface session
ushort	ATSBD return status in last surface session
ushort	Seconds taken in sending last SBD message
ushort	current CPU battery voltage counts 0.01V
ushort	current pump battery counts 0.01V
ushort	DP->Air[0] = pcase vacuum at end of last xmit (previous cycle) 0.01 inHg
ushort	DP->Air[1] = pcase vacuum at beginning of abort 0.01inHg
ushort	DP->ISRID = i.d. of last interrupt
ushort	AbrtCd = code for what caused abort_miss 0 = no error 1 = current time is later than RTCabort 2 = unable to WakeOST 3 = unable to send Dive number to SOLO II (LOdiveNo) 4 = Iridium ground station commanded to go to abort 5 = FnlDiv was completed. Mission is done 6 = Diagnostic dive failed to get GPS fix, pressure never>dBarGo, or unable to send message to Iridium 7 = pressure sensor failure
char	**;** terminator

Mission EEPROM dump (ID=0xdn, n=0-d)

Byte	Contents
0	ID/Mission phase = 0xd0,0xd1,0xd2,0xd3 [Possible values 0:d]
1-2	len=Number of bytes (variable, typically 888 for SOLO II)
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. (15 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:

BLOK=   1| **PB1**=   10| **PB2**= 1005|...   **ZN_CF**=   70| **Z_Neu**=   7600|;

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=189 bytes.

Mission Command echo (ID=0xde)

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

Argo Data ID=0xf0 Relayed in normal cycles

Byte	Contents
0	ID/Mission phase = 0xf0
1-2	Number of bytes = 25 = 0x19
3	Data Version (Minor version in high order nibble, major version in low order)=0.3
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 in minutes. V0.3, Apr14: 1 count =5 minutes.
16	Float Version: 1=Deep SOLO, 0=SOLO
17	Target ascent rate while profiling
18-19	Number of seeks
20-21	Surface Time
22-23	Seek Interval in minutes
24-25	SBE_Pgain
26-27	SBE_Poff
28-29	SBE_Tgain
30-31	SBE_Toff
32-33	SBE_Sgain
34-35	SBE_Soff
36	**;** terminator

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	
0x0002	2	Valve failed to close	
0x0004	4	Questionable pressure	
0x0008	8	Antenna was toggled	Surface
0x0010	16	Antenna switch failure (no satellites even after toggling)	Surface
0x0020	32	GPS communication error: No GPS	Surface
0x0040	64		
0x0080	128		
0x0100	256	Float took too long to leave the surface. (toggled valve)	Surface
0x0200	512		
0x0400	1024	Excessive Fall Speed: Abort Mission (return to surface immediately)	Fall,Drift
0x0800	2048		
0x1000	4096	Valve failure during Sink phase of mission	
0x2000	8192	Valve failure during Ascend phase of mission	
0x4000	16384		
0x8000	32768		