SBE SOLO DOCUMENTATION
(ROM version SBE601)

The following manual covers the algorithms required to post- process the ARGOS data from a SBE SOLO. This version is specific for ROM version SBE601 and later which allow the ARGO PTT ID to be either a 20 bits or 28 bits. The data message is necessarily reduced from 256 bits to 248 bits since ARGOS gets the extra 8 bits of ID information from the space previously allotted to the sensor data.

Rev. 1.0 : 3 June 2002 Update Notes:
Initial version of documentation created by modification of Rev 1.3 of documentation for SBE524 code. SBE600 is essentially the same code as SBE524 (for the HC11 controller) translated from FORTH to C for use with the HC12 controller.

Rev. 1.3 : 29 July 2002 Update Notes:
SBE601 changes the diagnostic messages to be more useful and elucidating and pops to the surface immediately after launch to send the diagnostic information. A new data format identifier 19 has been created. The format of messages 0,1,2,3 which contain the SBE profile data is the same as for data format 18. However SBE601 SOLOs have a special first dive with a diagnostic "C" message and the diagnostic messages have identifier "E" rather than "F" as with type 18 SOLOs.

Rev. 1.4 : 23 Sep 2002 Update Notes:

The scaling of temperature in the header messages is made consistent with that used in formats 18 and earlier. Changed descriptions of parameters in ARGOS messages. Changed format of "C" message. Changes since rev 1.3 are marked by dark bars at left and right end of each line.

Rev. 1.5 : 7 November 2002 Update Notes:
Changed description of ARGOS data formats

Rev. 1.6 : 1 June 2005 Update Notes:
Changed description of Bmax, Imin in Message #3

1. Description of the typical SBE SOLO cycle

The first dive following deployment is different from all successive dives. If SOLO successfully passes all the dockside tests, the success of the tests is indicated to the operator by filling the air and oil bladders to their full extent. After a half hour, the bladders are retracted to prepare SOLO for launch and SOLO begins to check periodically the depth reading from the SBE to see if it is in the water. When SOLO is placed in the water it will begin to sink since the bladders are fully retracted. When the depth is found to exceed 50 decibars, SOLO knows it has been deployed and will begin operations to return to the surface. It will take it about 10 minutes to reach 50 decibars. It starts pumping but it will fall to about 100 decibars before it turns around and begins its ascent. CTD data is taken during this short profile. About 40 minutes after deployment it will reach the surface and begin to send a special message which contains diagnostic information from the dockside tests as well as the CTD profile data from the deployment dive. This diagnostic message has a special identifying code "C" and is described in Appendix C.

All successive dives have the same pattern which we now describe starting with the SOLO on the surface with the piston fully extended and the air sleeve bladder inflated.

1) The piston is retracted fully. The air valve is opened, emptying the air sleeve bladder. SOLO begins to sink. Once SOLO is deeper than 100 dBar, the piston is moved to the position which kept it at the target depth on the previous dive cycle.
1) The SOLO descends for a pre-programmed X hours. At the end of X hours it will have settled out at its neutral depth.
2) The SOLO does multiple seeks, comprised of moving the piston to get closer to the desired parked depth, and then waiting to settle out. The final piston position is then used as the starting point for the next cycle in step 1.
3) The SOLO waits for N hours (programmable) at its neutral depth.
    a) It measures P , T and S every Y hrs (programmable) during this period.
    b) It averages P , T and S for the first N/2 hrs (values Pavg1, Tavg1, Savg1).
    c) And averages P , T and S for the last N/2 hrs (Pavg2, Tavg2, Savg2).
4) SOLO fully retracts the piston to sink as deep as possible. The ascening profile is performed (the piston is fully extended and data sampled) as the SOLO rises. The SBE CTD acquires data at a 1 Hz rate and then averages them over depth bins, the specifications of which are programmed into the SBE. The ascent lasts a set time (programmable) to ensure SOLO gets to the surface.
5) The air sleeve bladder is inflated. This ensures the antenna is comfortably out of the water.
6) The data is further processed and compacted into the ARGOS messages.
7) SOLO transmits the ARGOS messages through a rotating buffer over a programmable period (typically 18 - 24 hours).
8) The SOLO returns to 1) for the start of the next cycle.

2. Description of the Calibration file LOGxxxx (xxxx = SOLO S/N)

All of the pre- deployment calibration information is collated into the LOGxxxx file. A sample file is given here (19Aug02 refers to when the LOG file was written):

```
SN2100 ID 23457  -99 SB6.01  21Jul2002
MISSION 19Aug02    19   24.00  166.90
MULTCYC 19Aug02    10  166.9   20  100.0    15 200.0     800  -1   600   600
CALIBRT 19Aug02  0.5000   -10.0   0.065  0.26   0.027  0.03
THRMCAL 19Aug02    8.00    0.00 0.0 0.0 0.0 0.0
PROFORM 19Aug02      5    2    4   20   40
SBE_CAL 19Aug02      78  8.000    32000.00
END DATA
```

Each line has the following meaning:

```
SN2100     ID 23457     -99         SB6.01        21Jul2002
SOLO s/n  ARGOS ID   2nd ARGOS ID   ROM ver.    ROM creation date
```

```
MISSION 19Aug02    19   24.00  166.90
                   ID   TMSRF  TMCYCLE
```
ID = i.d. type.

ID = 19 = SBE SOLO 601 **(revised 29July2002)**

TMSRF = time spent on surface transmitting to ARGOS (hrs)

TMCYCLE = time to complete one cycle. If this is a multi- cycle float, the following line takes precedent:

```
MULTCYC 19Aug02    10  166.9   20  100.0    15 200.0     800  -1   600    600
                 CYC0  TIME0 CYC1  TIME1  CYC2 TIME2   PTAR1 DIR  PTAR2 PTAR0
```
See Section 4 for description of the multi- cycle variables.

```
CALIBRT 19Aug02  0.5000   -10.0   0.065  0.26   0.027  0.03
                 PGAIN    POFF   PMPGAIN PMPOFF CPUGAIN CPUOFF
```

where xGAIN, xOFF are the gain and offset calibration coefficients for the x sensor. x=P refers to Pressure (dBar), x=PMP refers to the pump battery voltage (Volts), and x=CPU is the CPU battery voltage (Volts). To convert to engineering units:

$$x = xCNTS * xGAIN + xOFF \quad (2.1)$$

where xCNTS are the a/d counts for the x sensor from the ARGOS message.

```
THRMCAL 19Aug02    8.00    0.00 0.0 0.0 0.0 0.0
                   TGAIN   TOFF |…dummy variables…|
```
where T (degrees C) = (TCNTS * TGAIN + TOFF) * 0.001 (2.2)

```
SBE_CAL 19Aug02      78  8.000    32000.00
                    s/n  SGAIN    SOFF
```
s/n is the SBE serial number

and S (PSU) = (SCNTS * SGAIN + SOFF) * 0.001 (2.3)

```
PROFORM 19Aug02      5    2    4   20   40
                   BLOK  AV1  AV2  PB1  PB2
```
See Section 3 on how to convert the above to the profile depth bin values.

3. Converting the data bin # of the profile to pressure (dBar)

The profile is comprised of 56 bins, with varying resolution with depth. For instance, shallow bins are typically spaced 5 m apart, medium bins 10 m apart, and deep bins spaced 20- 40 m apart. The data are averages within each depth bin. The depth of a bin is the **bottom** of the bin.

The depth bin parameters are found in the PROFORM line in the LOGxxxx file :

```
PROFORM 19Aug02       5    2    4   20   40
                   BLOK  AV1  AV2  PB1  PB2
```

BLOK = bin spacing for the shallow bins (I <= PB1)
AV1*BLOK = bin spacing for the medium bins (PB1 < I <= PB2)
AV2*BLOK = bin spacing for the deep bins (I > PB2)

For the above example,
there are 20 bins (=PB1) of 5 dBar spaced bins (=BLOK)
20 bins (=PB2- PB1) of 10 dBar spaced bins (=AV1*BLOK)
16 bins (=56 - PB2) of 20 dBar spaced bins (=AV2*BLOK)
giving a depth range of
20 bins from 0..100 dBar at 5 dBar resolution
20 bins from 100..300 dBar at 10 dBar resolution
16 bins from 300..620 dBar at 20 dBar resolution

The following is example FORTRAN code to compute z(I)

```
      subroutine sbe_depth(z)
c ................................................................
c     ..compute P (dBars) over 56 bins using coeff. in common pro
      real z(56)
      common /pro/blok,av1,av2,pb1,pb2

c        ...compute depths
        do i=1,56
          if (i.le. pb1) then
            z(i) = i*blok
          else
            if (i.le. pb2) then
              z(i) = z(i-1) +  av1*blok
            else
              z(i) = z(i-1) +  av2*blok
            endif
          endif
        enddo

      return
      end
c ................................................................
```

4. Description of the Multiple Cycle Parameters

```
MULTCYC 19Aug02    10  166.9   20  100.0    15 200.0     800  -1   600    600
                 CYC0  TIME0 CYC1  TIME1  CYC2 TIME2   PTAR1 DIR  PTAR2 PTAR0
```

```
At the start of the mission the SOLO will do:
      CYC0 dives, each cycle taking TIME0 (hrs), seeking a depth of PTAR0.
      The above SOLO will do 10 dives, each dive taking 166.9 hrs,
      with the park depth of 600 dBar.

The SOLO will then alternate between CYC1, CYC2, and CYC0 for the remainder
of its life, i.e. in pseudo-code:
      do while(true)
            do CYC1 dives, taking TIME1 hrs for each dive, seeking PTAR1
            do CYC2 dives, taking TIME2 hrs for each dive, seeking PTAR2
            do CYC0 dives, taking TIME0 hrs for each dive, seeking PTAR0
      enddo
The above SOLO would first do 10 dives (CYC0), then alternate between:
      20 dives at 100.0 hrs each, seeking a park depth of 800 dBar, and
      15 dives at 200.0 hrs each, seeking a park depth of 600 dBar.
      10 dives at 166.9 hrs each, seeking a park depth of 600 dBar.

DIR   = profile direction.
      =  0 = profiles on the way down.
      = -1 = profiles on the way up.
This has no effect on the way the data are processed, but is included as a descriptive
parameter for the SOLO.
```

5. Description of the Systeme ARGOS data formats

Data from SOLO is transmitted to the laboratory using Systeme ARGOS satellites. The satellites can handle messages which have at most 256 bits or 32 bytes per message. Each SOLO has a unique identifying PTT ID which Systeme ARGOS uses to route the data to the correct user. Older platforms had an ID code which contained 20 bits. As the ARGOS system became more popular, the number of ID bits was extended to 28 bits. However, the satellites already in orbit could not be modified to handle a different message length so the extra 8 bits of ID code for the 28 bit codes were obtained by usurping 8 bits from the data message. Thus a 20 bit ID platform can send 256 bits or 32 bytes of data per message while a platform with a 28 bit ID can send only 248 bits or 31 bytes of data in each ARGOS message. In order to be able to use either 20 bit or 28 bit IDS, SOLO packs it data into streams of 31 bytes or 248 bits of actual data per message.

When Systeme ARGOS sends data back to the user, the format of the message is slightly different depending upon whether the ID is a 20 bit ID or a 28 bit ID. Suppose SOLO has a 31 byte data message and that this message in hexadecimal representation is :
    0708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122232425.
This is an illustrative message and is NOT what a real SOLO message would look like. The format of the message the user gets from Systeme ARGOS also depends upon the type of processing the user has specified to ARGOS that he wants applied to the data.

Suppose the SOLO has a 28 bit ID. We assume that the user has told Systeme ARGOS that the data stream consists of 8 sensors with sensors 1 through 7 having 32 bits and sensor 8 having 24 bits. We also assume that the user has specified A2 processing which returns the data in hexadecimal form. This is the specification which was made at the time the PTT IDs were requested from Systeme ARGOS. The total data stream then consists of 7*32+24=248 bits = 31 bytes. SOLO transmits these 31 bytes and the resulting file from Systeme ARGOS would look like:

```
00559 33346   5  8 L
      2002-11-06 23:30:44  2    708090A      B0C0D0E      F101112     13141516
                               1718191A     1B1C1D1E     1F202122       232425
```

Now suppose SOLO has a 20 bit ID. The ARGOS message has room for 32 bytes, and we assume that the user has told Systeme ARGOS that the data stream consists of 8 sensors each having 32 bits and has requested A2 processing. To prevent confusion when System ARGOS processes the data all 32 bytes are filled. The software in SOLO prepends a special code byte (0xDB) to the 31 bytes of actual data and sends the 32 byte modified message. When the satellite receives this message, the file returned from Systeme ARGOS looks like:

```
00559 33346   5  8 L
      2002-11-06 23:30:44  2   DB070809      A0B0C0D      E0F1011     12131415
                               16171819     1A1B1C1D     1E1F2021     22232425
```

The parsing of the files obtained from Systeme Argos is best done by viewing the data message as a stream of bytes. Although Systeme ARGOS views the data stream as being broken into a number of fixed- length fields, the SOLO data does not have this structure. The difference between the message formats between 20 and 28 bit IDs may look confusing but things are not really that bad. Before we proceed, note that ARGOS drops any leading 0 (zero) at the start of a sensor field and replaces it with a space. Also note that for a 28 bit ID, the final sensor field only contains 24 bits = 3 bytes = 6 hexadecimal characters and that the final field is RIGHT justified by adding 2 leading spaces. It is important that these leading spaces NOT be interpreted as leading zeroes in the data stream.

The most reliable way to determine what sort of processing to apply to the ARGOS data files is by the PTT ID for the SOLO. All of the 28 bit ID codes have an ARGOS decimal ID of 32768 or greater. If the ID code is less than 32768, one can be sure that the platform has a 20 bit ID. An initial 0xDB byte in the data stream is also an unambiguous indicator that the ARGOS ID has 20 bits.

Processing 20 bit IDs

Create a 64 character hexadecimal string by copying the 8 groups of 8 characters, ignoring the 5 spaces between columns and the CRLF at end of each line. Convert any spaces (" ") in this string to zeroes ("0"). Discard the first 2 characters ("DB") to result in a string of 62 hexadecimal characters.

Processing 28 bit IDs

Create a 62 character hexadecimal string by copying the 7 groups of 8 characters, ignoring the 5 spaces between columns and the CRLF at end of each line. Concatenate the right-most 6 characters of the final field in the final line. Convert any spaces (" ") in this string to zeroes ("0").

The discussion below assumes that data from 20 and 28 bit IDs have been converted to 62 character strings in this fashion.

6. Description of the PTT messages and Header information

Because the profile data is longer than an ARGO message, SOLO sends the profile data in 4 messages, each containing 31 bytes of information (which is described by the 62 ASCII character streams above). Character 1 refers to the first character in this message; character 62 is the last character. The first 4 bits of the stream contain a message i.d. tag which will be either 0x0, 0x1, 0x2, 0x3, 0xC, or 0xE.

Each profile message contains 3 bytes of header information, and 28 bytes of profile information.
For a SBE-SOLO, each message contains 14 data bins:
Message #(0,1, 2, 3,) contain bins (1-14, 15-28, 29-42, 43-56) respectively.
Additionally, messages "E" and "C" contain diagnostic information:
Message "E" contains engineering data taken while profiling.
Message "C" contains diagnostic information taken during the self test and the deployment phases.

Header information

The first three bytes of each message contains information about the SOLO health. Since these three bytes are represented by 6 HEXADECIMAL characters in the ARGOS ASCII file, it is more appropriate to discuss the data in terms of characters .

Let a generic 12- bit value be represented by **ABC** where **A** is the most- significant character and **C** is the least significant. Also let **BC** represent a generic 8- bit value (i.e. pump voltage and cpu voltage)

In the following let *123456* denote character placement in the 6- character header.

Message #0:

char placement	*1*	*234*	*56*
generic data	**0**	**ABC**	**BC**
description	i.d.	Pavg1	Tavg2

```
i.d.= 0 = message i.d.
Pavg1 = average Press counts over the first half of the down time. Use
        Eqn. 2.1 to convert to dBars.
Tavg2 = the 8 lsb of avg T counts over the 2nd half of down time.
        The upper bits must be taken from Tavg1 (message 2), i.e.
        Tavg2 = BC + A(Tavg1) *256.  Use 2.2 to convert to deg. C.
        If abs(Tavg2-Tavg1) > 128, correct value to minimize difference.
```

Message #1:

char placement	*1*	*234*	*56*
generic data	**1**	**ABC**	**BC**
description	i.d.	Tavg1	Pavg2

```
i.d.= 1 = message i.d.
Tavg1 = average T counts over the first half of the down time.
        Use 2.2 to convert to deg. C.
Pavg2 = the 8 lsb of avg P counts over the 2nd half of down time.
        The upper bits must be taken from Pavg1 (message 1), i.e.
        Pavg2 = BC + A(Pavg1) * 256.  Use 2.1 to convert to dBars.
        If abs(Pavg2-Pavg1) > 128, correct value to minimize difference.
```

Message #2:

char placement	*1*	*234*	*56*
generic data	**2**	**ABC**	**AB**
description	i.d.	SPRX	FallT

```
i.d.= 2 = message i.d.
SPRX = average P counts at the surface at the end of transmitting in the previous
        cycle.  This is before any resetting of the pressure offset.  See PFS in
        the diagnostic message for surface pressure after the offset correction.
        In first cycle after deployment, SPRX is the pressure reading taken
        dockside during self test before the offset correction.
FallT = 0.01*( seconds elapsed from opening of air valve to sink to 50dBar)
```

Message #3:

char placement	*1*	*2*	*34*	*56*	
generic data	**3**	**C**	**BC**	**BC**	
description	i.d.	err	Imin	Bmax	**(Rev 1.1 update)**

```
i.d.= 3 = message i.d.
err = 4-bit error code, signifying a spurious interrupt,
      stack overflow, or spurious reset (see Diagnostics Appendix).
      err = 0 =no error.  Any other value indicates a problem of some sort.
Imin: (Imin+1)=minimum depth bin with valid data.  Typically Imin=0 for normal
      operation.  If for some reason the SOLO is ascending very slowly, the profile
      may time out, in which case Imin>0, and should be flagged for further
      inspection.  The profile is shifted so the data for bin (Imin+1) is in the
      first slot in the profile in message number 1.
Bmax= the number of bins with valid data.  (Imin+1) = the index of the first bin
      with valid data;  (Imax+1)=-the index of last bin with valid data.
      Then Bmax = (Imax+1) - (Imin+1) +1 = Imax-Imin+1 and (Imax+1) = Bmax + Imin.
      Since only 56 data bins are transmitted, if Bmax>56, all depth bins have
      data.  If Bmax<56, then the last data bins (I>Bmax) should be flagged as
      invalid.
```

Diagnostic Messages: Every 13th message sent by the SBE SOLO is a diagnostic. The first character of this message is an **E**. See Appendix B for information. Previous versions had a slightly different format for the diagnostic messages and were identified with the "F" character.

In the first dive following deployment, a special diagnostic message is send with Message ID="**C**". See Appendix C for information on the format of this diagnostic message.

7. Unpacking and Rescaling the ARGOS profile data

In general, T and S data are processed in the SOLO for each ARGOS message in the following way:

1) The first data bin in the message is left with its full resolution.
2) The rest of the profile is first-differenced (i.e. DT(i+1) = bin(i+1) - bin(i)).
3) The minimum and maximum values of DT are found (=DTmin and DTmax).
4) A LOOKUP table is used to find indices Kmin and Kmax such that:
   $\text{Scalar} * \text{LOOKUP(Kmin)} < \text{DTmin}$ (Scalar = 256 for T, 64 for S)
   $\text{Scalar} * \text{LOOKUP(Kmax)} \geq \text{DTmax}$
5) An offset and gain are computed as:
   OFF = LOOKUP(Kmin) * Scalar
   GAIN = LOOKUP(Kmax) - LOOKUP(Kmin)
5) DT is rescaled to form the output array ODT = (DT - OFF)/GAIN
6) The data are then packed into the ARGOS message, and the process is repeated for the next message.

The LOOKUP Table has 16 entries, and is the same for both T and S:

LOOKUP(1..16) =
[-4 -2.5 -1.5 -1 -.75 -0.5 -0.25 0 0.25 0.5 0.75 1 1.5 2.5 4 6.25]

<u>For the SBE-SOLO</u> characters 7-64 of each ARGOS messages denote:

```
char# 7     8     9     10    11    12    13-14....   39-40
data  KTmin KTmax KSmin KSmax TMSB2 TMSB1 TLSB(1)..   TLSB(14)

char# 41    42-44       45-47       ..60-62
data  SMSB1 SLSB(1,2)   SLSB(3,4)   ..SLSB(13,14)  SLSB are 6-bit values

char# 63-64
data  0x0000
```

KTmin and KTmax are indices into LOOKUP for the T data.
KSmin and KSmax are indices into LOOKUP for the S data.
TMSB1, TMSB2 are the most-significant bits for the first & last T bin in this message.
TLSB(i) i=2..14 are the rescaled T data for the 14 bins (8 bits per bin).
SMSB1 are the most-significant bits for the first S bin in the message.
SLSB(i) i=2..14 are the rescaled S data for the 14 bins (6 bits per bin). An easy way to unpack the 6-bit values is to read in 3 characters at a time and then split it into the two 6-bit values.

Algorithm to rescale either T or S :

Define Tscale = 256 (use for T), Sscale = 64 (use for S),
and substitute the correct value for Scale in the below.
Let nbins=14, the number of bins in one message

```
1) compute gain : GAIN = LOOKUP(Kmax+1) - LOOKUP(Kmin+1)
2) compute offset: OFF = LOOKUP(Kmin+1) * Scale
3) compute counts for the first bin :
        cnts(1) = MSB1 * Scale + LSB(1)
4) compute counts for i=2..nbins
        cnts(i) = cnts(i- 1) + LSB(i) * GAIN + OFF
5) use 2.2 or 2.3 to convert from counts to engineering units.
```

NOTE index values of Kmin+1, Kmax+1 are used for LOOKUP. This is because the SOLO processor uses k=0 as the first index value into an array, while Fortran uses k=1.

A. Unexpected Interrupt Error

If an unexpected interrupt occurs, an indicator of the source is return in variable **err** in ARGOS message #3. **Err** should always be zero. It is used to flag interrupt service routines that should never happen. In general, err>0 signifies a CPU or programming problem. Non-zero values are:

err	**Source (unexpected interrupt from:)**
0	no spurious interrupts
1	
2	
3	
4	RESET
5	Magnetic switch
6	A/D
7	SCI1
8	SCI0
9	SPI
A	pulse accumulator edge or overflow
B	Timer channel 1-7,timer overflow, or RTI
C	IRQ or XIRQ
D	Illegal instruction or software interrupt
E	clock monitor or computer operating properly
F	undefined interrupt

If any non-zero values are observed they should be reported.

B. Diagnostic "E" Message

Every 13th message transmitted by the SBE SOLO is a diagnostic, containing both discrete samples from the SBE and other engineering parameters. The following describes the 62 character message, where column 'Char'=character placement, with ‘4,5,6,7 signifying characters 4,5,6,and 7 comprise the 16 bits for parameter P1. The number in the last column refers to the corresponding stage that the datum was taken, as referenced by the outline in the *Description of the typical SBE SOLO cycl*e (p.2).

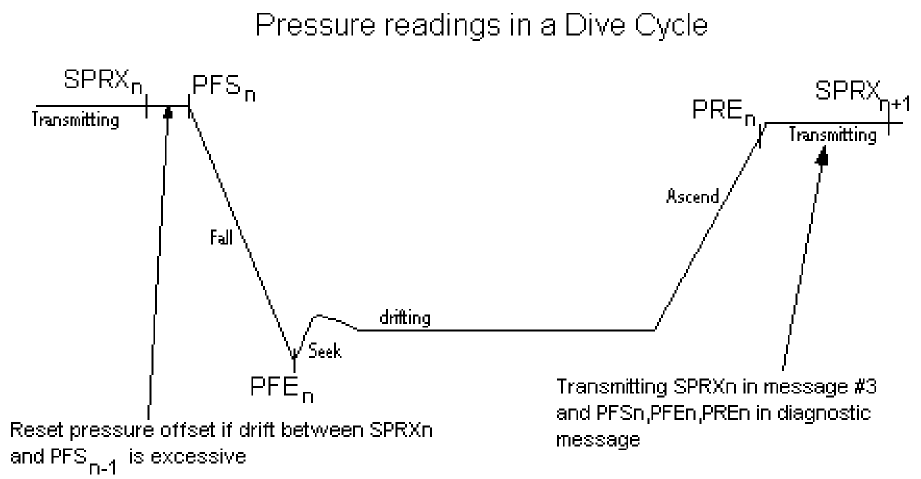
Char.	Name	Description	
1	id	Diagnostic message identifier = 'E'	
2	BST	4- bit status of miscellaneous operations (below)	
3,4,5,6	P1	Pressure counts before the start of ascent.	5
7,8,9,10	T1	Temperature counts at same time as P1	5
11 - 14	S1	Salinity counts at same time as P1	5
15 - 18	SBnscan	Number of Scans recorded by SBE	5
19	SBntry	Count of number of tries to start SBE	5
20	SBstat	SBE start/stop status	5
21,22a	VACb	Vacuum before fill air bladder 0.1 inHg (7bits)	
22b,23,24 a	VACa	Vacuum after fill air bladder 0.1 inHg (7bits)	
24b	LIMsw	IN lim@XMIT start, OUT lim@ASCEND start (1bit)	
25,26	PMPc	motor current @ end of ASCEND (mA=10*PMPc)	
27,28	Vcpu	CPU Voltage, LSB=0.1 Volts	
29,30	Vpmp	Pump battery Voltage, LSB=0.1 Volts	
31,32,33, 34	Savg1	Av S counts over the first half of the down time. Use 2.3 to convert to PSU.	

35,36	DS	DS = signed 8 LSB of Savg2 - Savg1.
		If DS>127, DS = DS - 256 gives the correct signed value
		Savg2 = Savg1 + DS.	

37,38	numbad	Number of bins in the profile with invalid data	
39,40,41	ATE	Air pressure inside of SOLO at end of last surface time	8
42,43,44	ATS	Air pressure inside of SOLO at start of last surface time	8
45,46,47	PFS	Pressure counts at the start of the SOLO Fall time after any reset of pressure offset that was required. In first cycle after deployment, PFS is the pressure reading taken dockside during self test after the offset correction.	1
48,49,50	PFE	Pressure counts at the end of the SOLO Fall time	2
51,52,53	PRE	Pressure counts at the end of the SOLO Rise time	5
54,55,56	TSK	seconds that piston ran during first SEEK cycle	3
57,58,59	PSK	(signed) dBar change in1st SEEK cycle=PGAIN*PSK	3
60,61,62	TIP	Seconds to run piston UP to get to SEEK depth	3

The 4 bits of SBstat(bit 0=LSB, bit 3=MSB) are assigned:
bit 0 = SBstop0 = 1 if SBE does not answer 'slp' command on ascent, else =0; ‘slp’
should return the pressure, while a profile is in process.
bit 1 = SBstop1 = 1 if no response to SBE ‘stop profile’ command
bits 2,4 = SBstart = 0 if no error, 1 if failed getting ‘slp’ response after SBE start- up,
2 if getting a response at sbe_start()

The 4 bits of BST (bit 0 = LSB, bit 3 =MSB) are assigned:
bit 3 = Alow = 1 if the air bladder was refilled during the surface transmit time
(not assigned for the first dive cycle, otherwise signifies a potential leaky
air valve).
bit 2 = PQUEST = 1 if the pressure counts is questionable at the end of the Fall time,
true if P counts puts us deeper than 2000 dBar OR shallower than Ptol,
a variable set during final programming. If set, the SOLO does no SEEKing,
and pulls the piston all of the way in.
bit 1 = OUT = 1 if the piston OUT limit switch is detected with the SOLO at the surface at the beginning
of transmit.(normally OUT=1)
bit 0 = IN = 1 if the piston IN limit switch is detected at the start of ascent.
This will depend upon profile direction: if profiling on ascent, then
normally IN=1, since SOLO sinks from target depth to bottom before starting the
ascent.


Pressure readings in a Dive Cycle
SPRXn
PFSn
Transmitting
Fall
Seek
PFEn
drifting
Ascend
PREn
SPRXn+1
Transmitting
Reset pressure offset if drift between SPRXn and PFSn-1 is excessive
Transmitting SPRXn in message #3 and PFSn,PFEn,PREn in diagnostic message

C. "C" Diagnostic Message in First Dive

In the first dive following deployment, the engineering "**E**" message is replaced by a "**C**" message with special diagnostic information.

Char.	Name	Description
1	id	Diagnostic message identifier = 'C'
2,3	LIMsw	8 bits of limit switch status. Bit 0 =LSB
		Bits7,6=IN,OUT state at start of BIT
		Bits5,4=IN,OUT state at end of BIT
		Bits3,2=IN,OUT state when mission starts
		Bits1,0=IN,OUT state at 1st surface XMIT
4,5,6	SPRXL	BIT pressure counts after resetoffset
7,8,9	diagP0	Pressure counts when "in water" sensed
10,11,12, 13	diagT0	Temperature counts when "in water" sensed
14,15,16, 17	diagS0	Salinity counts when "in water" sensed
18,19,20	diagP1	Shallowest pressure counts in profile
21,22,23, 24	diagT1	Shallowest temperature counts in profile
25,26,27, 28	diagS1	Shallowest salinity counts in profile
29,30	BTvac	BIT vacuum 0.1inHg
31,32	VACb	Vacuum before fill air bladder 0.1 inHg
33,34	VACa	Vacuum after fill air bladder 0.1 inHg
35,36	BTPcur	BIT motor current out (mA=10*BTPcur)
37,38	OUTcur	Motor current OUT to ASCEND (mA=10*OUTcur)
39,40,41	BTPsecs	BIT motor seconds OUT
42,43,44	INsecs	motor seconds IN to sink
45,46,47	OUTsecs	motor seconds OUT to ASCEND
48,49	BTpb	BIT pump battery LSB=0.1V
50,51	Vple	Pump battery @end ASCEND LSB=0.1V
52,53	BTcb	BIT CPU battery LSB=0.1V
54,55	Vcpu	CPU battery LSB=0.1V
56,57,58, 59	STsecs	.01*(seconds from BIT bladders empty to start of mission)
60,61,62	DURsecs	0.1*(seconds from start of mission to end 1st profile)