Rev. 1.0 **8. September 2000**

SBE SOLO DOCUMENTATION
(ROM version SBE520)

The following manual covers the algorithms required to post–process the ARGOS data from a SBE SOLO.

Section	Contents	page

1. Description of the typical SBE SOLO cycle

(Starting with the SOLO on the surface with the piston fully extended and the air sleeve bladder inflated).

1) The air valve is opened, emptying the air sleeve bladder, and the piston is retracted to its parked position determined from the last dive cycle.
2) The SOLO descends for a pre–programmed X hours. At the end of X hours it will have settled out at its neutral depth.
3) 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.
4) 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).
5) The ascent profile is performed (the piston is fully extended and data sampled as the SOLO rises). If it is a SBE–CP (continuous–profiler), data are acquired at a 1 Hz rate, and then averaged over the set depth range bins. If it is a SBE–DP(discrete–profiler), P, T and S are acquired at the set depth points only. The ascent lasts a set time (programmable) to ensure SOLO gets to the surface.
6) The air sleeve bladder is inflated. This ensures the antenna is comfortably out of the water.
7) The data is further processed and compacted into the ARGOS messages.
8) SOLO transmits the ARGOS messages through a rotating buffer over a 24 hr period (also programmable).
9) 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 (10May00 refers to when the LOG file was written):

```
SN1108 ID 23457  -99 SB5.20  03Dec1998
MISSION 10May00    17   24.00  166.90
MULTCYC 10May00    10  166.9   20  100.0    15 200.0     800  -1   600
CALIBRT 10May00  0.5000   -10.0   0.065  0.26   0.027  0.03
THRMCAL 10May00    8.00    0.00 0.0 0.0 0.0 0.0
PROFORM 10May00      5    2    4   20   40
SBE_CAL 10May00      78  8.000    32000.00
END DATA
```

Each line has the following meaning:

```
SN1108      ID 23457     -99          SB5.20        03DEC1998
SOLO s/n  ARGOS ID   2nd ARGOS ID   ROM ver.    ROM creation date
```

```
MISSION 10MAY00    17   24.00  166.90
                   ID   TMSRF  TMCYCLE
```

ID = i.d. type.   ID = 17 = SBE SOLO
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 10May00    10  166.9   20  100.0    15 200.0     800  -1   600
                 CYC0  TIME0 CYC1  TIME1   CYC2 TIME2   PTAR1 DIR  PTAR2
```

See Section 4 for description of the multi–cyle variables.

```
CALIBRT 10May00  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 (2.1)

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

```
THRMCAL 10May00    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 10May00      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 10May00      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. If the SBE–DP (discrete–profiler) is used, the CTD values are discrete samples at each of these depth values. If it is a SBE–CP (continuous–profiler) the data are averages centered on these depth values.

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

```
PROFORM 10May00       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 thedeep 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 10May00    10  166.9   20  100.0    15 200.0     800  -1   600
                 CYC0  TIME0 CYC1  TIME1  CYC2 TIME2   PTAR1 DIR  PTAR2

At the start of the mission the SOLO will do:
      CYC0 dives, each cycle taking TIME0 (hrs), seeking a depth of
PTAR2.
      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 AND CYC2 for the remainder
of its life, i.e. in psuedo-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
      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.

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 descriptional parameter for the SOLO.
```

5. Description of the PTT messages and Header information

The PTT alternates between 4 messages, each containing 32 data bytes. Each message contains 4 bytes of header information, and 28 bytes of profile information.

For a SBE−SOLO, each message contains 14 data bins:
Message #(1, 2, 3, 4) contain bins (1−14, 15−28, 29−42, 43−56) respectively.

Header information

The first four bytes of each message contains information about the SOLO health. Since these four bytes are represented by 8 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 *12345678* denote character placement in the 8−character header.

Message #1:

char placement	*1*	*234*	*56*	*78*
generic data	**0**	**ABC**	**BC**	**BC**
description	i.d.	Pavg1	Tavg2	Vcpu

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.
Vcpu = counts of the cpu Voltage. Use 2.1 to convert to Volts.

Message #2:

char placement	*1*	*234*	*56*	*78*
generic data	**1**	**ABC**	**BC**	**BC**
description	i.d.	Tavg1	Pavg2	Vpmp

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.
Vpmp = counts of the pump Voltage. Use 2.1 to convert to Volts.

Message #3:

char placement	*1*	*234*	*5678*
generic data	**2**	**ABC**	**ABCD**
description	i.d.	Sprss	Savg1

i.d.= 2 = message i.d.
Sprss = P counts at the surface at the end of ascent.
Savg1 = average S counts over the first half of the down time. Use 2.3 to convert to PSU.

Message #4:

char placement	*1*	*2*	*34*	*56*	*78*
generic data	**3**	**C**	**BC**	**BC**	**CD**
description	i.d.	err	Imin	Bmax	Savg2

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 should be flagged.
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.
Bmax= maximum depth bin with valid data. 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.
Savg2 = the 8 lsb of avg S counts over the 2nd half of down time. The upper bits must be taken from Savg1 (message 3), i.e.
Savg2 = **CD** + **AB**(Savg1) ∗ 256. Use 2.3 to convert to PSU.
If abs(Savg2−Savg1) > 128, correct value to minimize difference.

Diagnostic Message: Every 13th message sent by the SBE SOLO is a diagnostic. The first character of this message is an **F**. See the appendix A for information.

6. 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:
   Scalar ∗ LOOKUP(Kmin) < DTmin (Scalar = 256 for T, 64 for S)
   Scalar ∗ LOOKUP(Kmax) 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]

For the SBE−SOLO characters 9−64 of each ARGOS message denote:

```
char# 9     10    11    12    13    14    15-16....   41-42
data  KTmin KTmax KSmin KSmax TMSB2 TMSB1 TLSB(1)..   TLSB(14)

char# 43    44-46       47-49       ..62-64
data  SMSB1 SLSB(1,2)   SLSB(3,4)   ..SLSB(13,14)  SLSB are 6-bit values
```

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 = # 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 theSOLO processor uses k=0 as the first index value into an array, while Fortran uses k=1.

A. Diagnostics Appendix

This appendix is to help interpret some of the diagnostic messages not fully explained in the main section.

err : this variable is sent back in the ARGOS message #4 and should equal zero. It is mainly 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:)**
	1	SCI serial System
	2	SPI serial system
	3	Pulse Accum. Input
	4	Pulse Accum. Overflow
	5	Timer Overflow
	6	Timer Output Compare
	7	Timer Input Capture
	8	Real Time Interrupt
	9	PACE timer overflow
A	10	XIRQ
B	11	Software Interrupt
C	12	Illegal Op Code
D	13	COP failure
E	14	Clock Monitor Failure
F	15	FORTH STACK NOT EMPTY

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

SBE SOLO DIAGNOSTIC MESSAGE

Every 13th message transmitted by the SBE SOLO is a diagnostic, containing both discrete samples from the SBE and more engineering parameters for the SOLO in general. Its first character is the message ID, equal to ’F’. Presently this data is only being used in−house for float diagnostics.