Newton

Rev. 1.0 May 24, 1996

TP AND CTD ALACE DOCUMENTATION
(ROM versions TP420 and CTD420)

$\uparrow$ TYPE 11 $\quad$ $\uparrow$ TYPE 12

The following manual covers the algorithms required to post-process the ARGOS data from either a Temperature-Profiling (TP) or CTD ALACE. Fortran programs are also included to provide sample code, along with some simulated data for practice.

CLAUDIA SCHMID
NOAA/AOML/PHOD
4301 RICKENBACKER CAUSEWAY
MIAMI, FL 33149

1. Description of the typical ALACE cycle

(Starting with the ALACE on the surface with a full external bladder).

1) The valve is opened, emptying the oil in the external bladder.
2) The ALACE falls for ~14 hours (this interval is programmed before deployment). After this time it is assumed that the ALACE is at its neutral depth.
3) The ALACE waits for N hours (programmable) at its neutral depth.
   a) It measures P and T every 2.8 hrs during this period.
   b) It averages P and T for the first N/2 hrs (values Pavg1, Tavg1).
   c) And averages P and T for the last N/2 hrs (Pavg2, Tavg2).
4) The ascent profile is performed:
   a) P, T (and C for a CTD) are sampled every 2.5 s
   b) At the start of ascent the pump is run for 10-15 minutes (programmable). This extra buoyancy causes the ALACE to rise at ~10-15 cm/s.
   c) A depth bin is computed from the Pressure a/d counts (Pcnts) as:
      JBIN = Pcnts/BLOK (1.1)
   where BLOK is programmable. One Pcnt is equal to ~1 dBar so BLOK=10 would set the depth bin to ~10 dBar width.
   d) T and C are averaged over each JBIN on the ascent.
   e) The ascent lasts a set time (programmable) to ensure ALACE gets to the surface.
5) The pump runs for up to 90 minutes more to add extra buoyancy. This ensures the antenna is comfortably out of the water.
6) The T and C profiles are further averaged in depth.
7) Four ARGOS messages are composed.
8) ALACE transmits the 4 ARGOS messages through a rotating buffer over a 24 hr period (also programmable). While transmitting, surface P and T are sampled and averaged. These values are sent back in the next cycle's ARGOS messages.
9) Returns to 1) for the start of the next cycle.

The N hour interval in 3) is set so the total cycle time matches the user's desired value.

2. More information on how the profile data are processed

1) The shallowest bin to have data is the first bin to be sent back in the ARGOS message #1. This bin is called Imin and is equal to:
   Imin = Pmin/BLOK (2.1)
where Pmin is the minimum pressure sampled on ascent. Since surface pressure is set to ~100 counts in air, Imin ~100/BLOK. If Imin is much greater than this value, this suggests that the ALACE did not get to the surface before the end of the set profiling time. This could occur if the pump failed to work correctly, or was running at low speed due to low batteries. Imin is sent back in message ARGOS #4.

2) Since the ALACE T and C probes are mounted on the top end cap, they will spend part of their time in air at the surface.. Therefore the shallowest bin will have some air samples, and this will bias the measurements for the top bin. BEWARE of this when interpreting the data.

3) As mentioned above (Eqn 1.1), the original profile is averaged into depth bin sizes of BLOK pressure counts. To compact the data into only 4 messages, further averaging is performed for deeper bins. The final profile can have up to 3 different averaged-bin sizes. This will be more apparent in § 3 below.

4) For a CTD the conductivity values are corrected to second-order to remove the majority of the temperature signal. This new variable will be denoted as ES (for Estimated Salinity). Converting ES back to C is discussed in § 5.

5) The depth-averaged values of T and ES are divided into 4 sections (each section having the same number of depth bins). Each section is rescaled and then packed into its own ARGOS message. These 4 messages are then transmitted back using a rotating buffer scheme. Unpacking these messages will be discussed in § 8.

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

The profile is comprised of N bins (N=104 for a TP and N=56 for a CTD). To compact the data, deeper bins are averaged in depth to a greater extent. The amount of averaging and where that averaging begins are set by the parameters BLOK, AV1, AV2, PB1, and PB2. Their values can be found in the LOGxxxx file, PROFORM line (§ 6). The gain of the pressure sensor, PGAIN, is nominally 1.0 count/dBar. Its calibrated value is the first variable in the CALIBRT line in the LOGxxxx file (§ 6).

The variables PB1 and PB2 refer to the first and second breakpoints of the initial profile, corresponding to where additional depth averaging is desired. The variables AV1 and AV2 refer to the amount of further depth averaging to perform. As discussed above (Eqn. 1.1), BLOK refers to the initial amount of depth averaging performed while the profile is actually taken.

To compute the pressure values for the ARGOS-transmitted profile, it is useful to define one new breakpoint, OPB2, which refers to where the 2nd breakpoint occurs for the output profile:

$$OPB2 = PB1 + (PB2 - PB1)/AV1 \qquad (3.1)$$

1-10

The first PB1 output bins are sent back at the BLOK resolution. Between PB2 and PB1 of the input profile the bins are further averaged by AV1. The number of output bins for this region are reduced from (PB2-PB1) to (PB2 - PB1)/AV1, thus leading to the definition of OPB2 above. All deeper bins are averaged by AV2. This leads to the three different depth-averaging regions, defined by:

2

34

Let i equal the data bin # (i=1..N) and z(i) equal the center pressure (dBar) of bin i :

17

$$\begin{aligned} &\text{If } i \le PB1,^{25} && z(i) = (i - 0.5) \times PGAIN \times BLOK && (3.2a)\\ &\text{If } i > PB1, && z(i) = (i - PB1 - 0.25) \times AV1 \times PGAIN \times BLOK + z(PB1) && (3.2b)\\ &\text{If } i > \underset{45}{OPB2}, && z(i) = (i - OPB2 - 0.25) \times AV2 \times PGAIN \times BLOK + z(OPB2) && (3.2c) \end{aligned}$$

50m

The array z(i) can be used to plot P vs. T, C.

70m

100m

400m

ES = 32 + EScnt * .008
^
(CDIV * 4)

4. Converting Temperature counts to degrees C

After the 4 ARGOS messages have been unpacked (§ 8), temperature a/d counts (denoted as Tcnts) must be converted to degrees C. This is a two-step process.

1) First Tcnts is converted to a value of resistance R:

$$R \text{ (k-ohms)} = AT \times (BT + Tcnts) / (CT + Tcnts) \qquad (4.1)$$

where AT, BT and CT are from the THRMCAL line in the LOGxxxx file (§ 6).

2) R is then converted to T (degrees C). This will depend upon how the thermistor was actually calibrated. For the SIO calibration method the following eqn. is solved for T:

$$\ln(R) = h1 + h2/T + h2 \times h3 \times T^{-3} \qquad (4.2)$$

where h1, h2 and h3 are from the THRMCAL line in the LOGxxxx file (§ 6). Example code of solving for T from (4.2) is shown in the Fortran code unpack.for.

5. Converting Estimated Salinity (ES) to true salinity

After the values of ES are unpacked from the ARGOS messages (§ 8) they must be converted to useful values. The coefficients required to do this are:
CDIV, asal, bsal, and csal .
These are found in the LOGxxxx file, CONDCAL line (§ 6).

ES is first converted back to conductivity C (mmho/cm):

$$C \text{ (mmho/cm)} = 32 + ES \times \underline{CDIV \times 4} + asal + Tcnts \times (bsal + csal \times Tcnts) \qquad (5.1)$$

where Tcnts are the temperature a/d counts after being unpacked (§ 8).

It has been found that the conductivity calibration has not been reliable (although hopefully future calibrations will be better). To allow post calibration there is an additional line FSICAL in the LOGxxxx file (§ 6) which contains the variables COFF and CGAIN (initially set to 0 and 1.0 respectively). If the ALACE deployment is next to a CTD station and one is comfortable with knowing the in-situ values, the end user can do his own in-situ calibration and modify the COFF and CGAIN values in the FSICAL line. Corrected conductivity is then computed from:

$$C = COFF + CGAIN \times C \qquad (5.2)$$

In addition, FSI (Falmouth Scientific, Inc.) lists the following correction due to temperature and pressure effects on the conductivity cell geometry:

$$C = C \times \{ \; 1 + 7.5e\text{-}6 \times (CTREF - T) + 1.5e\text{-}8 \times z(i) \; \} \qquad (5.3)$$

where CTREF is the temperature of the calibration bath at the time of calibration, and T is the temperature (deg C) at depth z(i) (dBar). CTREF is found in the FSICAL line.

After correcting C, salinity is computed from the equations of state using z(dBar) from § 3, T (deg C) in § 4, and the above value of C. Sample code for changing ES to C is in Unpack.for

For C = 60 mmho/cm , Z = 2000 dBar
C error = .002 mmho/cm
for (CTREF - T) = 20 °C
C error = $\underline{0.009}$ mmho/cm
Correction for T is of O(1) LSB count

6. Description of the Calibration file LOGxxxx (xxxx = ALACE S/N)

All of the pre-deployment calibration information has been collated into the LOGxxxx file. A sample CTD file is given here (12Feb96 refers to when the LOG file was written):

```
SN0601 ID 23457  CT4.20   31Jan1996
MISSION 12Feb96    12    24.00   191.19
CALIBRT 12Feb96  1.0509 -108.19 0.06557 0.32787 0.02740 0.10959
THRMCAL 12Feb96 -27.1052 -11753.77  8636.74 -11.7865 4357.21 -3186.18
PROFORM 12Feb96     10     2     4    30   80
CONDCAL 12Feb96  0.002     -0.69192E+01    0.40600E-02    0.14631E-05
FSICAL  12Feb96  -0.000    1.000    18.9
END DATA
```

Each line has the following meaning:

```
SN0601      ID 23457      CT4.20         31Jan1996
ALACE s/n   ARGOS i.d.    ROM version    ROM creation date code
```

```
MISSION 12Feb96     12     24.00   191.19
                    ID     TMSRF   TMCYCLE
```

ID = i.d. type. ID = 11 = TP ALACE, version 420 code
ID = 12 = CTD ALACE, version 420 code

TMSRF = time spent on surface transmitting to ARGOS (hrs)
TMCYCLE = time to complete one cycle (as defined in § 1), in hours.

```
CALIBRT 12Feb96  1.0509 -108.19 0.06557 0.32787 0.02740 0.10959
                 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:

$$x = xCNTS \times xGAIN + xOFF \tag{6.1}$$

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

```
THRMCAL 12Feb96 -27.1052 -11753.77  8636.74 -11.7865 4357.21 -3186.18
                    AT           BT         CT         h1        h2         h3
```

The coefficients are defined in § 4 and are used to calculate temperature.

```
PROFORM 12Feb96     10     2     4     30    80
                  BLOK   AV1   AV2    PB1   PB2
```

The parameters define the amount of depth averaging performed (§ 3).

```
CONDCAL 12Feb96  0.002     -0.69192E+01     0.40600E-02     0.14631E-05
                 CDIV          asal              bsal             csal
```

The coefficients are used to convert estimated salinity ES to conductivity (§ 5).

```
FSICAL   12Feb96  -0.000    1.000    18.9
                   COFF     CGAIN    CTREF
```

where the variables are used to further correct conductivity (Eqn. 5.3).

7) 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 CTD, each message contains 14 data bins:
Message #(1, 2, 3, 4) contain bins (1-14, 15-28, 29-42, 43-56) respectively.

For a TP, each message contains 26 data bins:
Message #(1, 2, 3, 4) contain bins (1-26, ~~17~~-52, 53-78, 79-104) respectively.
27

Header information

The first four bytes of each message contains information about the ALACE 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. Same for both CTD and TP ALACEs:

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. 6.1 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.  Follow § 4 to convert to deg. C.
        If abs(Tavg2-Tavg1) > 128, correct value to minimize difference.
Vcpu  = counts of the cpu Voltage.  Use 6.1 to convert to Volts.
```

Message #2. Same for both CTD and TP ALACEs :

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.
        Follow § 4 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 6.1 to convert to dBars.
        If abs(Pavg2-Pavg1) > 128, correct value to minimize difference.
Vpmp  = counts of the pump Voltage.  Use 6.1 to convert to Volts.
```

Pavg1 [illegible]
14 count

Message #3. TP ALACE only:

char placement	*1*	*234*	*56*	*78*
generic data	**2**	**ABC**	**BC**	**AB**
description	i.d.	Sprss	Tpmp	Tsrf

i.d.= 2 = message i.d.
Sprss = avg P counts at the surface during the last cycle.
Tpmp = total pump time during second pump to empty internal reservoir.
time (minutes) = $0.666 \times$ Tpmp.
Tsrf = (avg T counts)/16 at the surface during last ARGOS transmit.
Tsrf $\times$ 16 = T counts to use to compute T in normal way (§ 4).

Message #4. TP ALACE only:

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

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= minimum depth bin with data: this always corresponds to the first bin in message #1. Since Press counts ~ 100 in air, Imin ~ 100/BLOK. Larger values imply ALACE did not reach the surface before profiling ended.
Bmax= maximum depth bin with data. Since only 104 data bins are transmitted for a TP, if Bmax>104, all depth bins have data.
Vptt= PTT voltage during ARGOS transmit. No calibration is given to convert this to volts. However, it can be used to tell when the switched-down regulator gets too low, and PTT transmission uses the Pump batteries directly (should see a jump up in Vptt).

Message #3 CTD ALACE only:

char placement	*1*	*234*	*5*	*678*
generic data	**2**	**ABC**	**C**	**ABC**
description	i.d.	Sprss	V5C	V100

i.d., Sprss are the same as in Message #3, TP ALACE.
V5C = most-significant bits to the 50% conductivity reference counts.
V100 = counts of the conductivity's 100% reference value.
V100 and V50 are taken right after the first pump cycle.

Message #4 CTD ALACE only:

char placement	*1*	*2*	*34*	*56*	*78*
generic data	**3**	**C**	**BC**	**BC**	**AB**
description	i.d.	err	Imin	Bmax	V50

i.d., err, Imin and Bmax are the same as in Message #4, TP ALACE.
V50 = V5C $\times$ 256 + V50 =counts of the conductivity's 50% reference value.

V50 and V100 provide diagnostics on the FSI electronics drift (Diagnostics Appendix).

New Message: Every 13th message sent by the CTD ALACE is a diagnostic containing a time series of the a/d counts of the conductivity cell. The first character of this message is an **F**. See the appendix A for information.

8. Unpacking and Rescaling the ARGOS profile data

In general, T and ES data are processed in the ALACE 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 ES)
   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 ES:

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] (8.1)

<u>For the TP-ALACE</u>
Bytes 5-32 of each message contain the profile information. Expressing these bytes in terms of character position in the message (characters 9-64), the profile portion contains:

char#	*9*	*10*	*11*	*12*	*13-14*		*15-16....*	*63-64*
data	Kmin	Kmax	MSB2	MSB1	LSB(1)	LSB(2)....		LSB(26)

Kmin and Kmax are indices into the LOOKUP table for the minimum and maximum scalars for this section of the profile.
MSB1, MSB2 are the most-significant bits for the first and last bin in this message.
LSB(i) i=2..26 are the rescaled data for the 26 bins (8 bits per bin).

<u>For the CTD-ALACE</u> characters 9-64 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*	$\overset{23}{44\text{-}46}$	*47-49*	..$\overset{32}{62\text{-}64}$	
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 ES 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 ES bin in the message.
SLSB(i) i=2..14 are the rescaled ES data for the 14 bins (6 bits per bin). An easy way to unpack the 6-bit values can be found in Unpack.for, subroutine rdsal.

Algorithm to rescale either T or ES :

Define Tscale = 256 (use for T), Sscale = 64 (use for ES),
and substitute the correct value for Scale in the below.
Let nbins = 26 for TP-ALACES and nbins=14 for CTD = # 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) check that the last bin has the correct most-significant bits:
        (not done for ES since SMSB2 is not sent back)
        diff = ABS( cnts(nbins) - MSB2 × Tscale )
        if diff<(Scale/2) then computed the right value
        else flag the error
```

See code p. 22

Sample code is given in the Fortran program Unpack.for (§ 9)

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

64 32 16 8 4 2 1
0 0 0 0 0 0

256 16 1

SMSB2

9. Sample Fortran Programs

Included are two sample Fortran programs which demonstrate how to select the most-likely correct ARGOS messages (Argosget.for) and then how to unpack the data and convert to engineering units (unpack.for). They are meant for diagnostic purposes only. Both programs have been compiled on a 486PC using Lahey Fortran, and on a Macintosh using Language Systems Fortran. Other platforms have not been tested.

These programs don't do a number of tasks required for general ALACE bookkeeping, such as:

1) There is no estimate of surface drift, nor of the ALACE drift rate at its neutral depth.
2) The programs do not archive the data. There is a wide range of personal preferences on the best archiving technique, so it is left up to the user to provide what is most convenient for him.

Argosget.for Takes the ARGOS data dumped over a modem using the **DS** command format and creates an output file **argxxxx** (xxxx = ALACE s/n). This file includes some header information plus the four ARGOS messages which occur most frequently. These are assumed to be the correct (error-free) messages to process.

This program will ask for your PROGRAM CODE. For the below simulation data this has been set to '12345.' If you are not sure of your program code, check the output of the ARGOS DS command. The program code is right before the ARGOS i.d. number, and is a 5-character string.

Unpack.for Takes the above output file **argxxxx**, unpacks and rescales the T and C data and converts to engineering units, writing out the results to **datxxxx**. It also writes out the unpacked profile, but not yet converted to engineering data, to **unpack.out**. This file may be helpful in diagnosing post-processing errors.

The above programs require some additional files to operate correctly:

argosid is used by Argosget to match the ARGOS ID number to the ALACE s/n, thus creating the correct **argxxxx** output file name. A sample file is included on floppy, whose contents looks like:

```
%each line of data file contains: ALACE serial number, ARGOS id
600  23456
601  23457
```

LOGxxxx has already been described in §6, and contains all of the calibration data. It is also used by Unpack.for to decide if the ALACE is a TP or CTD.

For demonstration purposes, the following files are included on floppy as well:

LOG0600 simulated log file of a TP ALACE
LOG0601 simulated log file of a CTD ALACE

INP0600 simulated ARGOS data file of a TP ALACE
INP0601 simulated ARGOS data file of a CTD ALACE

By running the INPxxxx files through Argosget.for, **argxxxx** files will be created. In turn, running these through Unpack.for will create **datxxxx** files. These output files are also included on floppy for comparison purposes.

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 which 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 to WRC.

CTD ALACE DIAGNOSTIC MESSAGE

Every 13th message transmitted by the CTD ALACE is a diagnostic of the FSI conductivity cell. This data is taken just before the profile is started (§1.4). The contents of this message is (a generic value of ABCD below denotes a 16-bit value):

char placement	*1*	*234*	*5-8*	*9-12*	*13-16*	*17-20*	*21-24*	*25-26*	*27-28*
generic data	**F**	**ABC**	**0ABC**	**ABCD**	**ABCD**	**ABCD**	**ABCD**	**CD**	**CD**
description	id	Pcnts	T1	V100	V50	V0	CVAL	V100	V50

char placement	*29-30*	*31-32*	*33-34*	*35-36*	....	*55-56*	*57-58*	*59-60*	*61-64*
generic data	**CD**	**CD**	**CD**	**CD**		**CD**	**CD**	**CD**	**0ABC**
description	CVAL	V100	V50	CVAL	....	V100	V50	CVAL	T2

```
id = F identifies this message as the diagnostic message.
Pcnts, T1 = P, T counts sampled after turning on the CTD.
V100, V50, V0 are the FSI 100%, 50%, 0% sampled reference values.
CVAL = FSI conductivity value (also in counts).
The above variables are sent with full resolution.
This is followed by a time series of conductivity comprised of 6
additional samples, separated by 2.5 s in time between samples.  Only
the least-significant byte of V100, V50 and CVAL are sent for this time
series.  This time series shows the temporal variability of the FSI.
T2 = T counts at the end of the conductivity time series.
```

The above values of V100, V50, V0 and CVAL have a range of 0..16384. To directly compare these values with the 12-bit values of V100, V50 in ARGOS messages 3 and 4, the above values should be divided by 4.