



# DATA PRODUCT SPECIFICATION FOR OXYGEN CONCENTRATION FROM "STABLE" INSTRUMENTS

Version 1-01

Document Control Number 1341-00520

2013-09-25

Consortium for Ocean Leadership  
1201 New York Ave NW, 4<sup>th</sup> Floor, Washington DC 20005  
[www.OceanLeadership.org](http://www.OceanLeadership.org)

in Cooperation with

University of California, San Diego  
University of Washington  
Woods Hole Oceanographic Institution  
Oregon State University  
Scripps Institution of Oceanography  
Rutgers University

**Document Control Sheet**

<b>Version</b>	<b>Date</b>	<b>Description</b>	<b>Author</b>
0-01	2012-01-31	Initial Draft	M.Vardaro
0-02	2012-02-01	Minor formatting updates.	S. Webster
0-03	2012-04-20	Changes to equations and formatting to incorporate Stern-Volmer equation and firmware update	M.Vardaro, O. Kawka, G. Proskurowski, M.B. Neely
0-04	2012-05-15	Copied test data set into document.	S. Webster
0-05	2012-05-23	Incorporated comments from focused review and potential density calculation	M. Vardaro
0-06	2012-06-10	Added language about interpolated timing and Appendix E regarding collocated CTDs	M. Vardaro
0-07	2012-06-14	Incorporated comments and corrections from S. Webster	M. Vardaro
0-08	2012-07-03	Incorporated comments from formal review	M. Vardaro
1-00	2012-07-05	Initial Release	E. Chapman
1-01	2013-09-25	Revision to incorporate updated firmware information and three deployment configurations	M. Vardaro

### Signature Page

This document has been reviewed and approved for release to Configuration Management.

OOI Senior Systems Engineer:

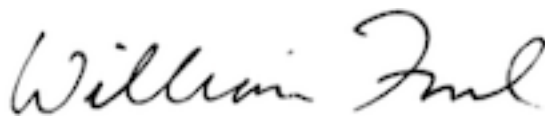


---

Date: 2012-07-05

This document has been reviewed and meets the needs of the OOI Cyberinfrastructure for the purpose of coding and implementation.

OOI CI Signing Authority:



---

Date: 2012-07-05

## Table of Contents

1	Abstract .....	1
2	Introduction.....	1
2.1	Author Contact Information .....	1
2.2	Metadata Information .....	1
2.3	Instruments .....	2
2.4	Literature and Reference Documents .....	2
2.5	Terminology .....	2
3	Theory .....	3
3.1	Description .....	3
3.2	Mathematical Theory .....	3
3.3	Known Theoretical Limitations .....	4
3.4	Revision History .....	4
4	Implementation.....	4
4.1	Overview .....	4
4.2	Inputs .....	4
4.3	Processing Flow.....	6
4.4	Outputs .....	7
4.5	Computational and Numerical Considerations .....	8
4.6	Code Verification and Test Data Set .....	8
Appendix A	DO Example Code.....	1
Appendix B	Potential Density Example Code .....	1
Appendix C	Output Accuracy .....	1
Appendix D	Sensor Calibration Effects .....	1
Appendix E	Collocated CTD Lookup Table for DOCONCS Data .....	1

## 1 Abstract

This document describes the computation used to calculate the core OOI Level 2 Dissolved Oxygen data product (DOCONCS) from "stable" instruments (DOSTA), which is calculated using the L0 output from Aanderaa oxygen optodes. This document is intended for use by OOI programmers to construct appropriate processes to create the L2 oxygen data product.

## 2 Introduction

### 2.1 Author Contact Information

Please contact Michael Vardaro ([mvardaro@coas.oregonstate.edu](mailto:mvardaro@coas.oregonstate.edu)) or the Data Product Specification lead ([DPS@lists.oceanobservatories.org](mailto:DPS@lists.oceanobservatories.org)) for more information concerning the computation and other items in this document.

### 2.2 Metadata Information

#### 2.2.1 Data Product Name

The OOI Core Data Product Name for this product is

- DOCONCS

The OOI Core Data Product Descriptive Name for this product is

- Oxygen Concentration from "Stable" Instruments (DOSTA)

#### 2.2.2 Data Product Abstract (for Metadata)

The OOI Level 2 Oxygen Concentration core data product DOCONCS is computed using the L0 output from the DOSTA class of instruments, specifically Aanderaa oxygen optodes, as well as the L2 Practical Salinity Data Product (PRACSAL); potential density, derived from the L2 Density Data Product (DENSITY); and the L1 Pressure (Depth) Data Product (PRESWAT), all derived from collocated Conductivity, Temperature, and Depth (CTD) instruments.

#### 2.2.3 Computation Name

Not required for data products.

#### 2.2.4 Computation Abstract (for Metadata)

This computation calculates the OOI Level 2 Oxygen Concentration core data product **DOCONCS** using a series of polynomial equations that incorporate the L0 DOCONCS phase data from the Aanderaa oxygen optodes and L2 PRACSAL, potential density (derived from L2 DENSITY), and L1 PRESWAT Data Products, all derived from collocated Conductivity, Temperature and Depth (CTD) instruments. The timing of the input data will be interpolated using the time stamps from the DOSTA and collocated CTD instruments and the INTERP1 QC algorithm (1341-10002)

#### 2.2.5 Instrument-Specific Metadata

See Section 4.4 for instrument-specific metadata fields that must be part of the output data.

#### 2.2.6 Data Product Synonyms

Synonyms for this data product are

- Dissolved Oxygen
- Dissolved Oxygen, Stable

### 2.2.7 Similar Data Products

This data product may be confused with the Level 1 Oxygen concentration data product (DOCONCF) from "fast response" oxygen instruments (DOFST).

## 2.3 Instruments

For information on the instruments from which the L2 Oxygen Concentration for "stable" instrument (DOCONCS) core data product inputs are obtained, see the DOSTA Processing Flow document (DCN 1342-00520) and Appendix E (Collocated CTD Lookup Table). The flow document contains information on the instrument class and make/model; it also describes the flow of data for DOSTA through all of the relevant QC, calibration, and data product computations and procedures.

Please see the Instrument Application in the OOI Software Application Framework (SAF) for specifics of instrument locations and platforms.

## 2.4 Literature and Reference Documents

TD 269 Operating Manual Oxygen Optode 4330, 4831, 4835. August 2012. 84 pp.  
(see *DPS Artifacts* >> [1341-00520 DOCONCS](#) >> [TD269 Oxygen Optode 4330 4835 4831.pdf](#))

Hydes, D.J., Hartman, M.C., Hartman, S.E., Barger, C.P. 2007. Evaluation of the Aanderaa Oxygen Optode in continuous use in the NOC Portsmouth Bilbao FerryBox system 2005, 2006, with an assessment of the likely errors in the estimation of oxygen concentration anomalies. 74 pp.  
(see *DPS Artifacts* >> [1341-00520 DOCONCS](#) >> [Hydes\\_etal\\_2007.pdf](#))

Thierry, V., Gilbert, D., Kobayashi, T. 2013. Processing Argo OXYGEN data at the DAC level, Version 1.3. 20 pp.  
(see *DPS Artifacts* >> [1341-00520 DOCONCS](#) >> [ARGO\\_oxygen\\_proposition\\_v1p2.pdf](#))

Uchida, H., Kawano, T., Kaneko, I., & Fukasawa, M. 2008. *In situ* Calibration of Optode-Based Oxygen Sensors. *Journal of Atmospheric and Oceanic Technology*, 25, 2271-2281.  
(see *DPS Artifacts* >> [1341-00520 DOCONCS](#) >> [Uchida 2008 Stern Volmer optode.pdf](#))

## 2.5 Terminology

### 2.5.1 Definitions

Definitions of general OOI terminology are contained in the Level 2 Reference Module in the OOI requirements database (DOORS).

### 2.5.2 Acronyms, Abbreviations and Notations

General OOI acronyms, abbreviations, and notations are contained in the Level 2 Reference Module in the OOI requirements database (DOORS). The following acronyms and abbreviations are defined here for use throughout this document.

CTD = Conductivity, Temperature, and Depth family of instruments

CT = Conservative Temperature

### 2.5.3 Variables and Symbols

The following variables and symbols are defined here for use throughout this document:

$O_2$  = the measured  $O_2$  concentration in micromoles per liter ( $\mu\text{mol L}^{-1}$ ).

$T_{\text{ctd}}$  = temperature in degrees Celsius ( $^{\circ}\text{C}$ ) as measured by the CTD instrument

$T_{\text{opt}}$  = temperature in degrees Celsius ( $^{\circ}\text{C}$ ) as measured by the optode  
 $V_p$  = voltage signal from a CTD, converted into phase (Pt) using analog conversion coefficients  
 $A_p$  = phase analog conversion offset coefficient  
 $B_p$  = phase analog conversion slope coefficient  
 $V_t$  = voltage signal from a CTD, converted into optode temperature ( $T_{\text{opt}}$ ) using analog conversion coefficients  
 $A_t$  = optode temperature analog conversion offset coefficient  
 $B_t$  = optode temperature analog conversion slope coefficient  
  
 $P_t$  = the uncorrected phase  
 $\text{csv}_1 - \text{csv}_7$  = calibration coefficients provided by the vendor (and potentially modified following post-deployment calibration)  
 $O_{2c}$  = pressure and salinity compensated  $O_2$  in micromoles per kilogram ( $\mu\text{mol kg}^{-1}$ )  
 $p$  = pressure in dbar  
 $p_r$  = reference pressure ( $p_{\text{ref}}$ ) = 0 dbar  
 $S$  = salinity  
 $\sigma$  = PDENS = potential density in  $\text{kg m}^{-3}$

### 3 Theory

#### 3.1 Description

The oxygen optodes use temperature readings, fluorescent response, and the luminescence quenching principle to internally calculate the oxygen-dependent phase change, the L0 data product. This value must then be converted to a temperature-corrected concentration using the Stern-Volmer-Uchida equation (L1 data product) and then further corrected to compensate for local salinity and depth (L2 data product).

#### 3.2 Mathematical Theory

The modified Stern-Volmer-Uchida equation is given by

$$O_2 = [(P_0/P_c) - 1] / K_{SV}$$

where:

$$K_{SV} = \text{csv}_1 + \text{csv}_2 T_{\text{opt}} + \text{csv}_3 T_{\text{opt}}^2$$

$$P_0 = \text{csv}_4 + \text{csv}_5 T_{\text{opt}}$$

$$P_c = \text{csv}_6 + \text{csv}_7 P_t$$

where  $T_{\text{opt}}$  is temperature from the optode instrument in  $^{\circ}\text{C}$ ,  $P_t$  is the uncorrected phase,  $P_0$  is the zero-oxygen phase shift, and  $\text{csv}_1$  through  $\text{csv}_7$  are calibration coefficients to be determined (see Uchida et al, 2008).

The above equations yield dissolved freshwater  $O_2$  in  $\mu\text{mol L}^{-1}$  or  $\mu\text{M}$ . This must be compensated for pressure and salinity and converted to derived SI units ( $\mu\text{mol kg}^{-1}$ ):

$$O_{2c} = k_{\text{salinity}} \cdot k_{\text{pressure}} \cdot 1000 O_2 / \sigma$$

$$k_{\text{pressure}} = 1 + (0.032p / 1000)$$

$$k_{\text{salinity}} = (e^{[S(B_0 + B_1 t_S + B_2 t_S^2 + B_3 t_S^3)]}) + C_0 S^2$$

where:

$$t_s = \ln[(298.15 - T) / (273.15 + T)]$$

$$B_0 = -6.24097e - 3$$

$$B_1 = -6.93498e - 3$$

$$B_2 = -6.90358e - 3$$

$$B_3 = -4.29155e - 3$$

$$C_0 = -3.11680e - 7$$

and where  $p$ ,  $T_{\text{ctd}}$ ,  $S$  and  $\sigma$  are pressure, temperature, salinity and potential density (Uchida, et al. 2008). Potential density ( $\sigma$ ) is calculated using a function in the L2 DENSITY DPS “`gsw_rho(SA,CT,p_ref)`”, where  $p_{\text{ref}} = 0$  (see Appendix B).

### 3.3 Known Theoretical Limitations

The sensor is limited to a range of 0-120% saturation and 0-500  $\mu\text{M}$ . The sensors and the foils are only calibrated to 500 $\mu\text{M}$ ; beyond these limits, a lower accuracy and precision should be expected. The 120% saturation limit is given for extreme conditions, which will rarely occur in reality. Global range quality control algorithms should be set to flag values that approach that magnitude.

### 3.4 Revision History

Revised to version 1-01 on September 20, 2013 to incorporate firmware upgrades.

## 4 Implementation

### 4.1 Overview

This is a mathematical operation to convert the calibrated phase recorded by the Aanderaa optode into temperature-corrected oxygen concentration output in  $\mu\text{mol L}^{-1}$ , then correct that output for local salinity and instrument depth and convert into derived SI units ( $\mu\text{mol kg}^{-1}$ ), as detailed in the Aanderaa manual and Uchida (2008). The computational methods described here are based on the data model used by Dr. Steven Riser’s group for optodes on Argo floats. This product is calculated using “potential density,” which is different from the OOI L2 DENSITY data product, so as to correct for compression-related changes in the volume and temperature of the parcel of water being measured. L2 DENSITY is computed using *in situ* temperature and pressure, while Potential Density is calculated using Conservative Temperature (CT) and a reference pressure of 0 dbar. Timing will be dealt with by interpolating the nearest CTD data to the time of sampling of the oxygen/CO2 data using the INTERP1 QC algorithm (1341-10002). The oxygen measurement will be delayed until the following CTD measurement.

Three different processing flows are presented here, one for an optode that is communicating through a CTD via an analog and digital connection (as on the BEP or EA moorings), one for optodes communicating through a CTD via digital connection only (with no analog connection), and a third for optodes that are operating autonomously and connected directly to a mooring DCL or junction box. Optodes communicating through a CTD via analog and digital connection will report raw phase and optode temperature in voltage units, optodes communicating through a CTD only via digital connection will report the L1 uncorrected oxygen concentration, while autonomous optodes will report the L0 calibrated phase as a 5 character floating point number (DOCONCS) and optode temperature as a 5 character floating point.

### 4.2 Inputs

If the optode is connected through a CTD via digital and analog connection (as at Regional and most Endurance sites), the inputs are:



- **L0, instrument-derived Calibrated Phase in raw voltage units ( $P_t$ ), DOCONCS**, as a 5 character floating point number, %.4f, extracted from the CTD data in hex code format
- **Instrument-derived raw thermistor temperature in raw voltage units ( $T_{opt}$ )**, as a 5 character floating point number, %.4f, extracted from the CTD data in hex code format
- **Phase scaling coefficients  $A_p$  and  $B_p$**  as 7 character floating point numbers, %.6f, available on optode startup.
- **Optode temperature coefficients  $A_t$  and  $B_t$**  as 7 character floating point numbers, %.6f, available on optode startup.
- CTD-derived temperature data ( $T_{ctd}$ ) in  $^{\circ}\text{C}$ , as a 5 character floating point number, %.3f
- CTD-derived pressure data ( $p$ ) in dbar,
- Calibration coefficients  $csv_1 - csv_7$

If the optode is connected through a CTD via a digital only connection (as at some Endurance sites), the inputs are:

- **L1, uncorrected oxygen concentration, DOCONCS**, as a 7 character floating point number, %.3f, extracted from the CTD data in hex code format
- CTD-derived temperature data ( $T$ ) in  $^{\circ}\text{C}$ , as a 5 character floating point number, %.3f
- Calibration coefficients  $csv_1 - csv_7$

If the optode is autonomous (as at Global and Pioneer sites), the inputs are:

- **L0 Instrument-derived Calibrated Phase ( $P_t$ ), DOCONCS**, as a 5 character floating point number, %.3f
- Optode-derived temperature data ( $T_{opt}$ ) in  $^{\circ}\text{C}$ , as a 5 character floating point number, %.3f
- Calibration coefficients  $csv_1 - csv_7$
- DOSTA Timestamp

Inputs from a collocated (or the closest possible) CTD (see Appendix E):

- CTD Timestamp
- L2 PRACSAL Practical Salinity (PSS-78) [unitless] (see 1341-00040)
- L1 PRESWAT Pressure ( $P$ ; dbar) (see 1341-00020)
- L1 TEMPWAT Temperature [ $^{\circ}\text{C}$ ] (see 1341-00010\_Data Product\_SPEC\_TEMPWAT)
- Latitude and longitude where the input data was collected is required. For the SBE 37IM and the SBE 16plusV2 this information is the lat/long of the mooring on which the instrument is fixed and is part of the metadata. For the SBE GPCTD this information is from the position of the glider or AUV when the CTD data were collected.

When available use L1b PRESWAT, L1b TEMPWAT, and L2b PRACSAL data products instead of L1a and L2a data products.

The potential density computation described in section 4.3 only produces valid results when the inputs are within the range of standard oceanographic values:

Absolute salinity:	$0 < SA < 42 \text{ g/kg}$
Temperature:	$-2.65^{**} < t < 40 \text{ }^{\circ}\text{C}$
Pressure:	$0 < p < 10,000 \text{ dbar}$

**\*\* Technically, the lower limit is  $-(2.65 + (p + 0.101325) * 0.0743)$  degrees C for  $p$  given in MPa, as per the IAPWS-09 standard for the pure liquid water component of the Gibbs function. However, a lower limit of  $-2.65$  degrees C can be used as a static replacement for OOI implementation.**

For potential density calculated from L1b TEMPWAT, L1b PRESWAT, and L2b PRACSAL, ranges checks will already have been applied as part of the calculation of these data products. For L1a and L2a CTD data products, the QC steps described in the CTD processing flow diagram should be performed (DCN 1342-00001).

**Input Data Formats:**

If the optode is connected through a CTD (as at Regional and all Endurance sites), the input format will be CTD output 0 (hex format):

0404F00A738B8173EA8526000E00142843A818820F6E

where the digits refer to:

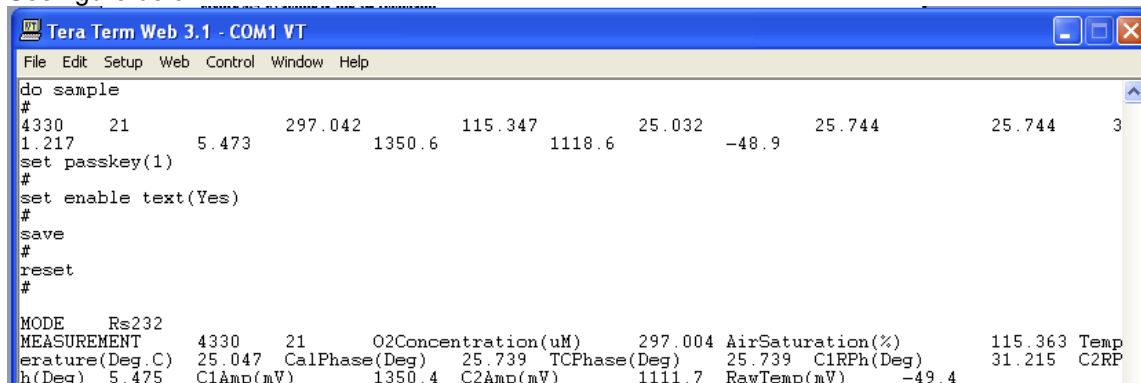
tttttccccppppppPTPTv0v0v1v1ooooooodddddd

When converted from hex, this is:

20.9831, 0.00004, 0.271, 0.0010, 0.0015, 253.976, 10 Jan 2013 23:37:28  
 ttt.tttt, cc.cccc, ppp.ppp, .v0v0, .v1v1, ooo.ooo, dd mmm yyyy, hh:mm:ss

Where oooo.ooo is the L1 uncorrected oxygen concentration (used only when a digital CTD connection is being used), and v.0v0v is the raw phase in voltage units (used when an analog CTD connection is available), which must be converted using the optode phase scaling coefficients, available in the header information upon optode startup in analog mode.

If the optode is autonomous (as at Pioneer and Global sites), refer to the Aanderaa manual, pages 21-30, for information on the RS232 commands and text string format of the data output. See figure below.



### 4.3 Processing Flow

The specific steps necessary to create all calibrated and quality controlled data products for each OOI core instrument are described in the instrument-specific Processing Flow documents (DCN 1342-00520). These processing flow documents contain flow diagrams detailing all of the specific procedures (data product and QC) necessary to compute all levels of data products from the instrument and the order in which these procedures are performed.

The processing flow for the pressure- and temperature-compensated oxygen concentration computation is as follows:

- Step 1: Ensure that salinity, temperature, and pressure data are available from a collocated CTD (or the closest available CTD at the same depth as the optode).
- Step 2: Interpolate the nearest CTD data (practical salinity, temperature, pressure, and the latitude and longitude associated with those data) to the time of sampling of the oxygen/CO2 data using the INTERP1 QC algorithm (1341-10002), where the inputs to the INTERP1 QC algorithm are

X = time of last two CTD data points

- Y = last two practical salinity/temperature/pressure data points or the last two lat/long associated with those measurements  
 XI = time of DOSTA sample.
- Step 3: Absolute salinity (SA) is calculated from practical salinity (SP, interpolated step 2 output), sea pressure (p, interpolated step 2 output), latitude (lat, interpolated step 2 output) and longitude (long, interpolated step 2 output) using the function from the TEOS-10 Toolbox  
 $[SA, in\_ocean] = gsw\_SA\_from\_SP(SP, p, long, lat)$   
 where in\_ocean is a flag indicating that the lat/long is well inside the boundaries of dry land (and is not used in this calculation).
- Step 4: Conservative temperature is computed from absolute salinity (SA, step 3 output), *in situ* temperature (t, interpolated step 2 output), and sea pressure (p, interpolated step 2 output) using a function from the TEOS-10 Toolbox.  
 $CT = gsw\_CT\_from\_t(SA, t, p)$
- Step 5: Potential density (rho) is calculated by the function  
 $\rho = gsw\_rho(SA, CT, p)$   
 from absolute salinity (SA, step 3 output), conservative temperature (CT, step 4 output), and a reference pressure (p, 0 dbar). This function is also part of the TEOS-10 Toolbox and uses the computationally efficient 48-term expression for density (McDougall et al., 2011).
- Step 6: Input the 7 csv calibration coefficients for the selected optode, and the conversion coefficients, A and B, for both phase and optode temperature (if in analog mode).
- Step 7: Download the optode data, extracting the measured temperature and excitation phase.  
**If the optode is autonomous, skip to step 9;**  
**If the optode is connected to a CTD via digital connection skip to step 9 but implement only the pressure and practical salinity correction code;**  
**If the optode is connected to a CTD via digital and analog connection, continue to step 8.**
- Step 8: Convert the phase raw voltage to phase using the analog conversion coefficients, where  $P_t = V_p \cdot B_p + A_p$ . Convert the optode temperature raw voltage to raw temperature using the analog conversion coefficients, where  $T_{opt} = V_t \cdot B_t + A_t$ .
- Step 9: Implement the provided Matlab code using the input values, creating temperature-compensated oxygen measurement (optode temperature,  $T_{opt}$ ), followed by the pressure correction (p, interpolated step 2 output), and finally the practical salinity correction (SP, interpolated step 2 output, using CTD Temperature,  $T_{ctd}$ )
- Step 10: Report the final temperature- and pressure-compensated dissolved oxygen concentration ( $O_{2c}$ ) and interpolated timestamp
- Step 11: Publish empty or dummy value upon computational error
- Step 12: Perform the necessary QC steps as outlined in the processing flow document

#### 4.4 Outputs

The outputs of the oxygen concentration computation are

- L2 DOCONCS—Salinity and depth compensated  $O_2$ -concentration,  $O_{2c}$ , as a 7 character floating point number, %.4f
- Interpolated timestamp from INTERP1 algorithm

The metadata that must be included with the output are

- The source of the PRACSAL, TEMPWAT, and PRESWAT inputs (which should be from the closest CTD to the DOSTA instrument, see Appendix E)
- The interpolated PRACSAL, TEMPWAT, and PRESWAT measurements used to perform the calculation.

See Appendix B for a discussion of the accuracy of the output.

## 4.5 Computational and Numerical Considerations

### 4.5.1 Numerical Programming Considerations

There are no numerical programming considerations for this computation. No special numerical methods are used.

### 4.5.2 Computational Requirements

L2 PRACSAL, Potential Density (derived from L2 DENSITY), and L1 PRESWAT Data Products must be available from a co-located CTD instrument, as well as the timestamp information

## 4.6 Code Verification and Test Data Set

Test data has also been uploaded to Alfresco.

(See REFERENCE >> DPS Artifacts >> [1341-00520 DOCONCS](#) >> [argo\\_DO\\_optode\\_test\\_data\\_withDOcal.mat](#))

Using the following example calibration values:

csv1	0.002848
csv2	0.000114
csv3	1.51E-06
csv4	70.42301
csv5	-0.10302
csv6	-12.9462
csv7	1.265377

L0 DOCONCS (P <sub>t</sub> )	Temperature (T) derived from optode	L2 PRACSAL	Potential Density ( $\sigma$ )	L1 PRESWAT	L2 DOCONCS (O <sub>2c</sub> )
Units	[degrees C]	[unitless]	[kg m <sup>-3</sup> ]	[dbar]	[ $\mu\text{mol kg}^{-1}$ ]
33.99	1.97	33.716	1026.94528	5.4	335.9830
33.99	1.964	33.715	1026.94495	9.9	336.1135
33.99	1.959	33.715	1026.94537	19.7	336.2852
34	1.954	33.716	1026.94658	27.1	336.1811
34	1.952	33.714	1026.94518	39.9	336.3503
34.03	1.894	33.713	1026.94879	50.5	336.4945
34.05	1.889	33.711	1026.9476	59.9	336.1723
34.1	1.86	33.716	1026.95382	70.1	335.4191
34.14	1.864	33.72	1026.95676	79.8	334.4750
34.21	1.842	33.725	1026.96245	90.7	333.1524
34.3	1.659	33.745	1026.99191	99.9	333.4128
34.39	1.577	33.767	1027.01546	110.2	332.3527
34.51	1.422	33.786	1027.0416	120.5	331.5409
34.75	1.243	33.816	1027.07789	130.1	328.1318
34.98	0.932	33.849	1027.12459	139.8	326.7290
35.36	0.421	33.872	1027.17349	150.1	324.4714
35.61	0.271	33.883	1027.19063	160.4	320.6565
35.79	0.297	33.916	1027.21585	170.1	316.1884
36.19	0.175	33.948	1027.24823	180.5	308.7235
36.39	0.283	33.985	1027.2723	188.2	302.9572
36.89	0.3	34.007	1027.28914	200.4	292.0942
37.4	0.59	34.066	1027.32024	210.4	278.2383
37.8	0.747	34.1	1027.33824	220.3	268.6475
38.38	0.981	34.147	1027.36143	230.4	255.2318
38.6	1.008	34.17	1027.37822	240.6	250.9844
39.14	1.208	34.203	1027.39158	250.2	239.4786
39.52	1.365	34.24	1027.41059	260	231.5373
39.99	1.535	34.273	1027.42511	269.9	222.2873
40.53	1.733	34.317	1027.44597	279.7	212.0485
40.84	1.815	34.342	1027.4599	288.4	206.6553
41.23	1.868	34.366	1027.47517	300.1	200.4688
41.51	1.929	34.389	1027.48896	310	195.9495
41.7	1.962	34.408	1027.50168	320.1	193.0158
41.84	1.988	34.419	1027.50852	330.1	190.8858
41.9	1.999	34.423	1027.51091	339.9	190.0146
42.04	2.019	34.433	1027.5174	349.9	187.9603
42.18	2.048	34.445	1027.52479	360.5	185.8586
42.39	2.126	34.471	1027.53952	379.7	182.4983
42.62	2.15	34.493	1027.55532	400.3	179.3288
42.5	1.813	34.501	1027.58824	449.5	183.6817
42.9	2.232	34.579	1027.61807	500.1	175.4939
42.55	1.848	34.559	1027.63254	550.1	183.2432
42.67	2.206	34.643	1027.67193	597.4	179.1683
42.59	2.043	34.641	1027.68362	649.9	181.7240

42.52	2.216	34.671	1027.69408	700.4	181.6470
42.5	2.181	34.679	1027.70358	749.6	182.4421
42.47	2.158	34.687	1027.71211	800.2	183.2954
42.38	2.136	34.697	1027.72215	850	184.9567
42.32	2.077	34.707	1027.73513	900.2	186.4977
42.04	2.063	34.717	1027.7445	948.1	190.7675
42.26	2.07	34.714	1027.74156	950.5	187.6621
42.15	2.057	34.721	1027.74847	999.9	189.5712
42.03	2.046	34.729	1027.75602	1050.2	191.6269
41.99	2.029	34.733	1027.76083	1099.5	192.6114
41.98	1.976	34.735	1027.76685	1150.5	193.4663
41.92	1.944	34.739	1027.77281	1200.3	194.8660
41.92	1.931	34.741	1027.77569	1249.8	195.2624
42.01	1.845	34.738	1027.78012	1300	194.9519
41.92	1.831	34.742	1027.78467	1350.2	196.6549
41.89	1.786	34.743	1027.78913	1400	197.7472
41.91	1.743	34.743	1027.79261	1450.2	198.1037
41.88	1.698	34.743	1027.79621	1499.9	199.2043
41.9	1.649	34.742	1027.79928	1550.3	199.6140
41.93	1.574	34.739	1027.80257	1600.1	200.0870
41.95	1.531	34.738	1027.80512	1650.1	200.4470
41.99	1.48	34.735	1027.80659	1699.7	200.5779
41.98	1.469	34.736	1027.80844	1750.1	201.1205
41.99	1.442	34.735	1027.80981	1800.6	201.5020
42	1.382	34.732	1027.81183	1850	202.1506
42.05	1.312	34.728	1027.81368	1899.9	202.2921
42.07	1.271	34.726	1027.81512	1950.1	202.6407
42.09	1.232	34.726	1027.81801	2000.7	202.9724

## Appendix A      DO Example Code

```
function DO = dosv(Pt,T,S,P,PDENS,csv)
% function DO = dosv(Pt,T,S,P,PDENS,csv)
%
% Calculate DO from optode bphase and temperature
% using the modified Stern-Volmer equation.
%
% INPUTS:
%
% Pt      : Optode bphase (degrees)
% T       : Temperature (deg C)
% S       : Salinity
% P       : Pressure (dbar)
% PDENS   : Potential density (kg/m^3)
% csv     : Stern-Volmer coefficients (7-element vector)
%
% Pt,T,S,P and PDENS may be vectors or matrices.
%
% OUTPUT:
%
% DO      : Dissolved oxygen (umol/kg)
%
% rev 04/16/2012 R DRUCKER
% UNIVERSITY OF WASHINGTON

% Check input variables:
if ~isequal(size(Pt),size(T),size(S),size(P),size(PDENS))
    error('Input variable sizes do not match')
end
if length(csv)~=7
    error('Stern-Volmer coefficients incorrect size')
end
[rows,cols] = size(Pt);
Pt = Pt(:);
T = T(:);
S = S(:);
P = P(:);
PDENS = PDENS(:);

% Calculate DO using Stern-Volmer:
Ksv = csv(1) + csv(2)*T + csv(3)*(T.*T);
P0 = csv(4) + csv(5)*T;
Pc = csv(6) + csv(7)*Pt;
DO = ((P0./Pc) - 1) ./ Ksv;

% Convert from volume to mass units:
DO = 1000*DO./PDENS;

% Pressure correction:
pcomp = 1 + (0.032*P)/1000;
DO = pcomp.*DO;
```

```
% Salinity correction:
S0 = 0;
ts = log((298.15-T)./(273.15+T));
B = [-6.24097e-3
     -6.93498e-3
     -6.90358e-3
     -4.29155e-3];
C0 = -3.11680e-7;
scomp = exp((S-S0).*(vandermonde(ts,3)*B)+C0*(S.^2-S0.^2));
DO = scomp.*DO;
```

```
% Reshape output:
DO = reshape(DO,rows,cols);
```

```
function V = vandermonde(x,n)
    x = x(:);
    m = length(x);
    V = NaN(m,n);
    V(:,1) = ones(m,1);
    for k = 2:n+1
        V(:,k) = V(:,k-1).*x;
    end
```



## Appendix B      Potential Density Example Code

This Appendix contains all Matlab subroutines necessary for calculating Potential Density from Temperature, Salinity, and Pressure using the GSW toolbox (TEOS-10) available under Creative Commons license from <http://www.teos-10.org/>. This code has been verified by the originators of the code using examples from the oceanographic community. The current version at the time of writing is GSW Toolbox 3.0.

Matlab, Fortran, and C routines for calculating density are available in the GSW toolbox from the TEOS-10 website.

<http://www.teos-10.org/>  
<http://www.teos-10.org/software.htm#1>

A copy of the toolbox has been placed on Alfresco.

REFERENCE >> DPS Artifacts >> DENSITY\_Code\_gsw\_matlab\_v30\_0.zip

### B.1      In-situ Density (48-term equation)

USAGE:                      rho = gsw\_rho(SA,CT,p\_ref)

DESCRIPTION:              Calculates in-situ density from Absolute Salinity and Conservative Temperature, using the computationally-efficient 48-term expression for density in terms of SA, CT and p (McDougall et al., 2011).

Note that potential density with respect to reference pressure, p\_ref, is obtained by calling this function with the pressure argument being p\_ref (i.e. "gsw\_rho(SA,CT,p\_ref)"), where p\_ref = 0 dbar.

Note that the 48-term equation has been fitted in a restricted range of parameter space, and is most accurate inside the "oceanographic funnel" described in McDougall et al. (2011). The GSW library function "gsw\_infunnel(SA,CT,p)" is available to be used if one wants to test if some of one's data lies outside this "funnel".

INPUT:                      SA = Absolute Salinity [ g/kg ]  
                                CT = Conservative Temperature [ deg C ]  
                                p\_ref = surface pressure [ dbar ] (i.e. 0 dbar)

SA & CT need to have the same dimensions.  
p may have dimensions 1x1 or Mx1 or 1xN or MxN, where SA & CT are MxN.

OUTPUT:                    rho = potential density [ kg m<sup>-3</sup> ]

AUTHOR:                    Paul Barker and Trevor McDougall

VERSION NUMBER:        3.0 (23rd May, 2011)

REFERENCES:              IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of seawater - 2010: Calculation and use of thermodynamic properties. Intergovernmental Oceanographic Commission, Manuals and Guides No. 56, UNESCO (English), 196 pp. Available from [the TEOS-10 web site](#). See appendix A.20 and appendix K of this TEOS-10 Manual.

McDougall T.J., P.M. Barker, R. Feistel and D.R. Jackett, 2011: A computationally efficient 48-term expression for the density of seawater in terms of Conservative Temperature, and related properties of seawater. To be submitted to Ocean Science Discussions.

```
function rho = gsw_rho(SA,CT,p_ref)
```

```
% gsw_rho                potential density (48-term equation)
%=====
%
% USAGE:
% rho = gsw_rho(SA,CT,p_ref)
%
% DESCRIPTION:
% Calculates in-situ density from Absolute Salinity and Conservative
% Temperature, using the computationally-efficient 48-term expression for
% density in terms of SA, CT and p (McDougall et al., 2011).
%
% Note that potential density with respect to reference pressure, pr, is
% obtained by calling this function with the pressure argument being pr
% (i.e. "gsw_rho(SA,CT,pr)").
%
% Note that the 48-term equation has been fitted in a restricted range of
% parameter space, and is most accurate inside the "oceanographic funnel"
% described in McDougall et al. (2011). The GSW library function
% "gsw_infunnel(SA,CT,p)" is available to be used if one wants to test if
% some of one's data lies outside this "funnel".
%
% INPUT:
% SA = Absolute Salinity                [ g/kg ]
% CT = Conservative Temperature (ITS-90) [ deg C ]
% p_ref = surface pressure                [ dbar ]
%       ( i.e. 0 dbar )
%
% SA & CT need to have the same dimensions.
% p may have dimensions 1x1 or Mx1 or 1xN or MxN, where SA & CT are MxN.
%
% OUTPUT:
% rho = potential density                [ kg/m ]
%
% AUTHOR:
% Paul Barker and Trevor McDougall      [ help_gsw@csiro.au ]
%
% VERSION NUMBER: 3.0 (18th March, 2011)
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
% See appendix A.20 and appendix K of this TEOS-10 Manual.
%
% McDougall T.J., P.M. Barker, R. Feistel and D.R. Jackett, 2011: A
% computationally efficient 48-term expression for the density of
% seawater in terms of Conservative Temperature, and related properties
% of seawater. To be submitted to Ocean Science Discussions.
%
```

```
% The software is available from http://www.TEOS-10.org
%
%=====

%-----
% Check variables and resize if necessary
%-----

if ~(nargin == 3)
    error('gsw_rho: Requires three inputs')
end %if

[ms,ns] = size(SA);
[mt,nt] = size(CT);
[mp,np] = size(p);

if (mt ~= ms | nt ~= ns)
    error('gsw_rho: SA and CT must have same dimensions')
end

if (mp == 1) & (np == 1)           % p scalar - fill to size of SA
    p_ref = p_ref*ones(size(SA));
elseif (ns == np) & (mp == 1)     % p is row vector,
    p_ref = p_ref (ones(1,ms), :); % copy down each column.
elseif (ms == mp) & (np == 1)     % p is column vector,
    p_ref = p_ref (:,ones(1,ns));  % copy across each row.
elseif (ns == mp) & (np == 1)     % p is a transposed row vector,
    p_ref = p_ref. ';              % transposed then
    p_ref = p_ref (ones(1,ms), :); % copy down each column.
elseif (ms == mp) & (ns == np)
    % ok
else
    error('gsw_rho: Inputs array dimensions arguments do not agree')
end %if

if ms == 1
    SA = SA';
    CT = CT';
    p = p';
    transposed = 1;
else
    transposed = 0;
end

%-----
% Start of the calculation
%-----

% These few lines ensure that SA is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

v01 = 9.998420897506056e+2;
v02 = 2.839940833161907;
```

v03 = -3.147759265588511e-2;  
v04 = 1.181805545074306e-3;  
v05 = -6.698001071123802;  
v06 = -2.986498947203215e-2;  
v07 = 2.327859407479162e-4;  
v08 = -3.988822378968490e-2;  
v09 = 5.095422573880500e-4;  
v10 = -1.426984671633621e-5;  
v11 = 1.645039373682922e-7;  
v12 = -2.233269627352527e-2;  
v13 = -3.436090079851880e-4;  
v14 = 3.726050720345733e-6;  
v15 = -1.806789763745328e-4;  
v16 = 6.876837219536232e-7;  
v17 = -3.087032500374211e-7;  
v18 = -1.988366587925593e-8;  
v19 = -1.061519070296458e-11;  
v20 = 1.550932729220080e-10;  
v21 = 1.0;  
v22 = 2.775927747785646e-3;  
v23 = -2.349607444135925e-5;  
v24 = 1.119513357486743e-6;  
v25 = 6.743689325042773e-10;  
v26 = -7.521448093615448e-3;  
v27 = -2.764306979894411e-5;  
v28 = 1.262937315098546e-7;  
v29 = 9.527875081696435e-10;  
v30 = -1.811147201949891e-11;  
v31 = -3.303308871386421e-5;  
v32 = 3.801564588876298e-7;  
v33 = -7.672876869259043e-9;  
v34 = -4.634182341116144e-11;  
v35 = 2.681097235569143e-12;  
v36 = 5.419326551148740e-6;  
v37 = -2.742185394906099e-5;  
v38 = -3.212746477974189e-7;  
v39 = 3.191413910561627e-9;  
v40 = -1.931012931541776e-12;  
v41 = -1.105097577149576e-7;  
v42 = 6.211426728363857e-10;  
v43 = -1.119011592875110e-10;  
v44 = -1.941660213148725e-11;  
v45 = -1.864826425365600e-14;  
v46 = 1.119522344879478e-14;  
v47 = -1.200507748551599e-15;  
v48 = 6.057902487546866e-17;

sqrtSA = sqrt(SA);

v\_hat\_denominator = v01 + CT.\*(v02 + CT.\*(v03 + v04\*CT)) ...  
+ SA.\*(v05 + CT.\*(v06 + v07\*CT)) ...  
+ sqrtSA.\*(v08 + CT.\*(v09 + CT.\*(v10 + v11\*CT)))) ...  
+ p\_ref.\*(v12 + CT.\*(v13 + v14\*CT) + SA.\*(v15 + v16\*CT)) ...  
+ p\_ref.\*(v17 + CT.\*(v18 + v19\*CT) + v20\*SA));

v\_hat\_numerator = v21 + CT.\*(v22 + CT.\*(v23 + CT.\*(v24 + v25\*CT))) ...

```
+ SA.*(v26 + CT.*(v27 + CT.*(v28 + CT.*(v29 + v30*CT))) + v36*SA ...  
+ sqrtSA.*(v31 + CT.*(v32 + CT.*(v33 + CT.*(v34 + v35*CT)))) ...  
+ p_ref.*(v37 + CT.*(v38 + CT.*(v39 + v40*CT)) ...  
+ SA.*(v41 + v42*CT) ...  
+ p_ref.*(v43 + CT.*(v44 + v45*CT + v46*SA) ...  
+ p_ref.*(v47 + v48*CT));
```

```
rho = v_hat_denominator./v_hat_numerator;
```

```
%-----  
% This function calculates rho using the computationally-efficient  
% 48-term expression for density in terms of SA, CT and p_ref. If one wanted to  
% compute rho from SA, CT, and p with the full TEOS-10 Gibbs function,  
% the following lines of code will enable this.  
%  
%   pt0 = gsw_pt_from_CT(SA,CT);  
%   pr0 = zeros(size(SA));  
%   t = gsw_pt_from_t(SA,pt0,pr0, p_ref);  
%   rho = gsw_rho_t_exact(SA,t,p);  
%  
%   or call the following, it is identical to the lines above.  
%  
%   rho = gsw_rho_CT_exact(SA,CT, p_ref)  
%  
%   or call the following, it is identical to the lines above.  
%  
%   [rho, ~, ~] = gsw_rho_alpha_beta_CT_exact(SA,CT, p_ref)  
%  
%-----This is the end of the alternative code-----  
  
if transposed  
    rho = rho.';  
end  
  
end
```

## B.2 Absolute Salinity from Practical Salinity

USAGE: [SA, in\_ocean] = gsw\_SA\_from\_SP(SP,p,long,lat)

DESCRIPTION: Calculates absolute salinity from practical salinity. Since SP is non-negative by definition, this function changes any negative input values of SP to be zero.

INPUT: SP = Practical Salinity (PSS-78) [ unitless ]  
p = sea pressure [ dbar ] ( i.e., absolute pressure - 10.1325 dbar )  
long = longitude in decimal degrees [ 0 ... +360 ] or [ -180 ... +180 ]  
lat = latitude in decimal degrees north [ -90 ... +90 ]

p, lat & long may have dimensions 1x1 or Mx1 or 1xN or MxN, where SP is MxN.

OUTPUT: SA = Absolute Salinity [g/kg ]  
in\_ocean = 0, if long and lat are a long way from the ocean  
= 1, if long and lat are in the ocean

Note. This flag is only set when the observation is well and truly on dry land; often the warning flag is not set until one is several hundred kilometres inland from the coast.

AUTHOR: Trevor McDougall, Paul Barker & David Jackett

VERSION NUMBER: 3.0 (23rd May, 2011)

REFERENCES: IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of seawater - 2010: Calculation and use of thermodynamic properties. Intergovernmental Oceanographic Commission, Manuals and Guides No. 56, UNESCO (English), 196 pp. Available from [the TEOS-10 web site](#). See section 2.5 and appendices A.4 and A.5 of this TEOS-10 Manual.

McDougall, T.J., D.R. Jackett and F.J. Millero, 2010: An algorithm for estimating Absolute Salinity in the global ocean. Submitted to Ocean Science. A preliminary version is available at Ocean Sci. Discuss., 6, 215-242. <http://www.ocean-sci-discuss.net/6/215/2009/osd-6-215-2009-print.pdf>

```
function [SA, in_ocean] = gsw_SA_from_SP(SP,p,long,lat)
```

```
% gsw_SA_from_SP Absolute Salinity from Practical Salinity
```

```
%=====
```

```
%
```

```
% USAGE:
```

```
% [SA, in_ocean] = gsw_SA_from_SP(SP,p,long,lat)
```

```
%
```

```
% DESCRIPTION:
```

```
% Calculates Absolute Salinity from Practical Salinity. Since SP is
```

```
% non-negative by definition, this function changes any negative input
```

```
% values of SP to be zero.
```

```
%
```

```
% INPUT:
```

```
% SP = Practical Salinity (PSS-78) [ unitless ]
% p = sea pressure [ dbar ]
% ( i.e. absolute pressure - 10.1325 dbar )
% long = longitude in decimal degrees [ 0 ... +360 ]
% or [ -180 ... +180 ]
% lat = latitude in decimal degrees north [ -90 ... +90 ]
%
% p, lat & long may have dimensions 1x1 or Mx1 or 1xN or MxN,
% where SP is MxN.
%
% OUTPUT:
% SA = Absolute Salinity [ g/kg ]
% in_ocean = 0, if long and lat are a long way from the ocean
% = 1, if long and lat are in the ocean
% Note. This flag is only set when the observation is well and truly on
% dry land; often the warning flag is not set until one is several
% hundred kilometres inland from the coast.
%
% AUTHOR:
% David Jackett, Trevor McDougall & Paul Barker [ help_gsw@csiro.au ]
%
% VERSION NUMBER: 3.0 (31st May, 2011)
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
% See section 2.5 and appendices A.4 and A.5 of this TEOS-10 Manual.
%
% McDougall, T.J., D.R. Jackett and F.J. Millero, 2010: An algorithm
% for estimating Absolute Salinity in the global ocean. Submitted to
% Ocean Science. A preliminary version is available at Ocean Sci. Discuss.,
% 6, 215-242.
% http://www.ocean-sci-discuss.net/6/215/2009/osd-6-215-2009-print.pdf
%
% The software is available from http://www.TEOS-10.org
%
%=====
%-----
% Check variables and resize if necessary
%-----

if ~(nargin==4)
    error('gsw_SA_from_SP: Requires four inputs')
end %if

[ms,ns] = size(SP);
[mp,np] = size(p);

if (mp == 1) & (np == 1) % p is a scalar - fill to size of SP
    p = p*ones(size(SP));
elseif (ns == np) & (mp == 1) % p is row vector,
    p = p(ones(1,ms), :); % copy down each column.
elseif (ms == mp) & (np == 1) % p is column vector,
```

```
p = p(:,ones(1,ns));          % copy across each row.
elseif (ns == mp) & (np == 1) % p is a transposed row vector,
    p = p.';                  % transposed then
    p = p(ones(1,ms), :);     % copy down each column.
elseif (ms == mp) & (ns == np)
    % ok
else
    error('gsw_SA_from_SP: Inputs array dimensions arguments do not agree')
end %if

[m1a,n1a] = size(lat);

if (m1a == 1) & (n1a == 1)    % lat is a scalar - fill to size of SP
    lat = lat*ones(size(SP));
elseif (ns == n1a) & (m1a == 1) % lat is a row vector,
    lat = lat(ones(1,ms), :); % copy down each column.
elseif (ms == m1a) & (n1a == 1) % lat is a column vector,
    lat = lat(:,ones(1,ns)); % copy across each row.
elseif (ns == m1a) & (n1a == 1) % lat is a transposed row vector,
    lat = lat.';              % transposed then
    lat = lat(ones(1,ms), :); % copy down each column.
elseif (ms == m1a) & (ns == n1a)
    % ok
else
    error('gsw_SA_from_SP: Inputs array dimensions arguments do not agree')
end %if

[m1o,n1o] = size(long);
[lwest] = find(long < 0);
if ~isempty(lwest)
    long(lwest) = long(lwest) + 360;
end

if (m1o == 1) & (n1o == 1)    % long is a scalar - fill to size of SP
    long = long*ones(size(SP));
elseif (ns == n1o) & (m1o == 1) % long is a row vector,
    long = long(ones(1,ms), :); % copy down each column.
elseif (ms == m1o) & (n1o == 1) % long is a column vector,
    long = long(:,ones(1,ns)); % copy across each row.
elseif (ns == m1o) & (n1o == 1) % long is a transposed row vector,
    long = long.';             % transposed then
    long = long(ones(1,ms), :); % copy down each column.
elseif (ms == n1o) & (m1o == 1) % long is a transposed column vector,
    long = long.';             % transposed then
    long = long(:,ones(1,ns)); % copy down each column.
elseif (ms == m1o) & (ns == n1o)
    % ok
else
    error('gsw_SA_from_SP: Inputs array dimensions arguments do not agree')
end %if

if ms == 1
    SP = SP.';
    p = p.';
    lat = lat.';
    long = long.';
```



```
    transposed = 1;
else
    transposed = 0;
end

[lout_of_range] = find(p < 100 & SP > 120);
SP(lout_of_range) = NaN;
[lout_of_range] = find(p >= 100 & SP > 42);
SP(lout_of_range) = NaN;

[Inan] = find(abs(SP) == 99999 | abs(SP) == 999999);
SP(Inan) = NaN;
[Inan] = find(abs(p) == 99999 | abs(p) == 999999);
p(Inan) = NaN;
[Inan] = find(abs(long) == 9999 | abs(long) == 99999);
long(Inan) = NaN;
[Inan] = find(abs(lat) == 9999 | abs(lat) == 99999);
lat(Inan) = NaN;

if ~isempty(find(p < -1.5 | p > 12000))
    error('gsw_SA_from_SP: pressure is out of range')
end
if ~isempty(find(long < 0 | long > 360))
    error('gsw_SA_from_SP: longitude is out of range')
end
if ~isempty(find(abs(lat) > 90))
    error('gsw_SA_from_SP: latitude is out of range')
end

%-----
% Start of the calculation
%-----

% These few lines ensure that SP is non-negative.
[l_neg_SP] = find(SP < 0);
if ~isempty(l_neg_SP)
    SP(l_neg_SP) = 0;
end

[locean] = find(~isnan(SP.*p.*lat.*long));

SA = nan(size(SP));
SAAR = nan(size(SP));
in_ocean = nan(size(SP));

% The following function (gsw_SAAR) finds SAAR in the non-Baltic parts of
% the world ocean. (Actually, this gsw_SAAR look-up table returns values
% of zero in the Baltic Sea since SAAR in the Baltic is a function of SP,
% not space.
[SAAR(locean), in_ocean(locean)] = gsw_SAAR(p(locean),long(locean),lat(locean));

SA(locean) = (35.16504/35)*SP(locean).*(1 + SAAR(locean));

% Here the Practical Salinity in the Baltic is used to calculate the
% Absolute Salinity there.
SA_baltic(locean) = gsw_SA_from_SP_Baltic(SP(locean),long(locean),lat(locean));
```

```
[lbaltic] = find(~isnan(SA_baltic(locean)));  
  
SA(locean(lbaltic)) = SA_baltic(locean(lbaltic));  
  
if transposed  
    SA = SA';  
    in_ocean = in_ocean';  
end  
  
end
```

### B.3 Conservative Temperature from in-situ Temperature

USAGE: CT = gsw\_CT\_from\_t(SA,t,p)

DESCRIPTION: Calculates Conservative Temperature of seawater from in-situ temperature.

INPUT: SA = Absolute Salinity [ g/kg ]  
t = in-situ temperature (ITS-90) [ deg C ]  
p = sea pressure [ dbar ] ( i.e. absolute pressure - 10.1325 dbar )

SA & t need to have the same dimensions.  
p may have dimensions 1x1 or Mx1 or 1xN or MxN, where SA & t are MxN.

OUTPUT: CT = Conservative Temperature [ deg C ]

AUTHOR: David Jackett, Trevor McDougall and Paul Barker

VERSION NUMBER: 3.0 (16th May, 2011)

REFERENCES: IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of seawater - 2010: Calculation and use of thermodynamic properties. Intergovernmental Oceanographic Commission, Manuals and Guides No. 56, UNESCO (English), 196 pp. Available from [the TEOS-10 web site](#). [See section 3.3 of this TEOS-10 Manual](#).

function CT = gsw\_CT\_from\_t(SA,t,p)

```
% gsw_CT_from_t      Conservative Temperature from in-situ temperature
%=====
%
%
% USAGE:
% CT = gsw_CT_from_t(SA,t,p)
%
% DESCRIPTION:
% Calculates Conservative Temperature of seawater from in-situ
% temperature.
%
% INPUT:
% SA = Absolute Salinity                [ g/kg ]
% t = in-situ temperature (ITS-90)      [ deg C ]
% p = sea pressure                      [ dbar ]
%      ( i.e. absolute pressure - 10.1325 dbar )
%
% SA & t need to have the same dimensions.
% p may have dimensions 1x1 or Mx1 or 1xN or MxN, where SA & t are MxN.
%
% OUTPUT:
% CT = Conservative Temperature (ITS-90) [ deg C ]
%
% AUTHOR:
% David Jackett, Trevor McDougall and Paul Barker [ help_gsw@csiro.au ]
%
```

```
% VERSION NUMBER: 3.0 (27th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
% See section 3.3 of this TEOS-10 Manual.
%
% The software is available from http://www.TEOS-10.org
%
%=====

%-----
% Check variables and resize if necessary
%-----

if ~(nargin==3)
    error('gsw_CT_from_t: Requires three inputs')
end %if

[ms,ns] = size(SA);
[mt,nt] = size(t);
[mp,np] = size(p);

if (mt ~= ms | nt ~= ns)
    error('gsw_CT_from_t: SA and t must have same dimensions')
end

if (mp == 1) & (np == 1)          % p scalar - fill to size of SA
    p = p*ones(size(SA));
elseif (ns == np) & (mp == 1)    % p is row vector,
    p = p(ones(1,ms), :);        % copy down each column.
elseif (ms == mp) & (np == 1)    % p is column vector,
    p = p(:,ones(1,ns));         % copy across each row.
elseif (ns == mp) & (np == 1)    % p is a transposed row vector,
    p = p.';                    % transposed then
    p = p(ones(1,ms),:);         % copy down each column.
elseif (ms == mp) & (ns == np)
    % ok
else
    error('gsw_CT_from_t: Inputs array dimensions arguments do not agree')
end %if

[lout_of_range] = find(p < 100 & (t > 80 | t < -12));
if (~isempty(lout_of_range))
    t(lout_of_range) = NaN;
end
[lout_of_range] = find(p >= 100 & (t > 40 | t < -12));
if (~isempty(lout_of_range))
    t(lout_of_range) = NaN;
end

if ms == 1
    SA = SA.';
```

```
t = t.';
p = p.';
transposed = 1;
else
    transposed = 0;
end

%-----
% Start of the calculation
%-----

pt0 = gsw_pt0_from_t(SA,t,p);
CT = gsw_CT_from_pt(SA,pt0);

if transposed
    CT = CT.';
end

end
```

## B.4 Potential Temperature w/ Ref Sea Pressure of Zero dbar

```
function pt0 = gsw_pt0_from_t(SA,t,p)

% gsw_pt0_from_t          potential temperature with a
%                          reference sea pressure of zero dbar
%
=====
%
% USAGE:
% pt0 = gsw_pt0_from_t(SA,t,p)
%
% DESCRIPTION:
% Calculates potential temperature with reference pressure, p_ref = 0 dbar.
% The present routine is computationally faster than the more general
% function "gsw_pt_from_t(SA,t,p,p_ref)" which can be used for any reference
% pressure value.
% This subroutine calls "gsw_entropy_part(SA,t,p)",
% "gsw_entropy_part_zerop(SA,pt0)" and "gsw_gibbs_pt0_pt0(SA,pt0)".
%
% INPUT:
% SA = Absolute Salinity          [ g/kg ]
% t  = in-situ temperature (ITS-90) [ deg C ]
% p  = sea pressure                [ dbar ]
%      ( i.e. absolute pressure - 10.1325 dbar )
%
% SA & t need to have the same dimensions.
% p may have dimensions 1x1 or Mx1 or 1xN or MxN, where SA & t are MxN.
%
% OUTPUT:
% pt0 = potential temperature      [ deg C ]
%      with reference sea pressure (p_ref) = 0 dbar.
% Note. The reference sea pressure of the output, pt0, is zero dbar.
%
% AUTHOR:
% Trevor McDougall, David Jackett, Claire Roberts-Thomson and Paul Barker.
% [ help_gsw@csiro.au ]
%
% VERSION NUMBER: 3.0 (29th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
% See section 3.1 of this TEOS-10 Manual.
%
% McDougall T.J., P.M. Barker, R. Feistel and D.R. Jackett, 2011: A
% computationally efficient 48-term expression for the density of
% seawater in terms of Conservative Temperature, and related properties
% of seawater. To be submitted to Ocean Science Discussions.
%
% The software is available from http://www.TEOS-10.org
%
```

```
%=====
%-----
% Check variables and resize if necessary
%-----

if ~(nargin == 3)
    error('gsw_pt0_from_t: Requires 3 inputs - Absolute Salinity, temperature, and pressure')
end %if

[ms,ns] = size(SA);
[mt,nt] = size(t);
[mp,np] = size(p);

if (ms ~= mt | ns ~= nt)
    error('gsw_pt0_from_t: Input arguments do not have the same dimensions')
end %if

if (mp == 1) & (np == 1)          % p scalar - fill to size of SA
    p = p*ones(size(SA));
elseif (ns == np) & (mp == 1)    % p is row vector,
    p = p(ones(1,ms), :);        % copy down each column.
elseif (ms == mp) & (np == 1)    % p is column vector,
    p = p(:,ones(1,ns));          % copy across each row.
elseif (ns == mp) & (np == 1)    % p is a transposed row vector,
    p = p.';                      % transposed then
    p = p(ones(1,ms), :);         % copy down each column.
elseif (ms == mp) & (ns == np)
    % ok
else
    error('gsw_pt0_from_t: Inputs array dimensions arguments do not agree')
end %if

if ms == 1
    SA = SA.';
    t = t.';
    p = p.';
    transposed = 1;
else
    transposed = 0;
end

%-----
% Start of the calculation
%-----

% These few lines ensure that SA is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

cp0 = 3991.86795711963;          % from Eqn. (3.3.3) of IOC et al. (2010).
SSO = 35.16504;                  % from section 2.4 of IOC et al. (2010).

s1 = SA*(35./SSO);
```

```
pt0 = t + p.*( 8.65483913395442e-6 - ...
    s1.* 1.41636299744881e-6 - ...
    p.* 7.38286467135737e-9 + ...
    t.*(-8.38241357039698e-6 + ...
    s1.* 2.83933368585534e-8 + ...
    t.* 1.77803965218656e-8 + ...
    p.* 1.71155619208233e-10));

entropy_dt = cp0./((273.15 + pt0).*(1-0.05.*(1 - SA./SSO)));

true_entropy_part = gsw_entropy_part(SA,t,p);

for Number_of_iterations = 1:2
    pt0_old = pt0;
    entropy = gsw_entropy_part_zeropt(SA,pt0_old) - true_entropy_part;
    pt0 = pt0_old - entropy./entropy_dt ; % this is half way through the modified method
    pt0m = 0.5*(pt0 + pt0_old);
    entropy_dt = -gsw_gibbs_pt0_pt0(SA,pt0m);
    pt0 = pt0_old - entropy./entropy_dt;
end

if transposed
    pt0 = pt0.';
end

% maximum error of 6.3x10^-9 degrees C for one iteration.
% maximum error is 1.8x10^-14 degrees C for two iterations
% (two iterations is the default, "for Number_of_iterations = 1:2").
% These errors are over the full "oceanographic funnel" of
% McDougall et al. (2011), which reaches down to p = 8000 dbar.

end
```



## B.5 Conservative Temperature from Potential Temperature

```
function CT = gsw_CT_from_pt(SA,pt)

% gsw_CT_from_pt    Conservative Temperature from potential temperature
%=====
%
% USAGE:
% CT = gsw_CT_from_pt(SA,pt)
%
% DESCRIPTION:
% Calculates Conservative Temperature of seawater from potential
% temperature (whose reference sea pressure is zero dbar).
%
% INPUT:
% SA = Absolute Salinity                [ g/kg ]
% pt = potential temperature (ITS-90)    [ deg C ]
%
% SA & pt need to have the same dimensions.
%
% OUTPUT:
% CT = Conservative Temperature (ITS-90)    [ deg C ]
%
% AUTHOR:
% David Jackett, Trevor McDougall and Paul Barker  [ help_gsw@csiro.au ]
%
% VERSION NUMBER: 3.0 (29th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
% See section 3.3 of this TEOS-10 Manual.
%
% The software is available from http://www.TEOS-10.org
%=====

%-----
% Check variables and resize if necessary
%-----

if ~(nargin == 2)
    error('gsw_CT_from_pt: Requires two inputs')
end %if

[ms,ns] = size(SA);
[mt,nt] = size(pt);

if (mt ~= ms | nt ~= ns)
    error('gsw_CT_from_pt: SA and pt must have same dimensions')
end
```

```
if ms == 1
    SA = SA.';
    pt = pt.';
    transposed = 1;
else
    transposed = 0;
end

%-----
% Start of the calculation
%-----

% These few lines ensure that SA is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

cp0 = 3991.86795711963;    % defined in Eqn. (3.3.3) of IOC et al. (2010).

sfac = 0.0248826675584615;    % sfac = 1/(40.*(35.16504/35)).

x2 = sfac.*SA;
x = sqrt(x2);
y = pt.*0.025;    % normalize for F03 and F08.

pot_enthalpy = 61.01362420681071 + y.*(168776.46138048015 + ...
    y.*(-2735.2785605119625 + y.*(2574.2164453821433 + ...
    y.*(-1536.6644434977543 + y.*(545.7340497931629 + ...
    (-50.91091728474331 - 18.30489878927802.*y).*y)))) + ...
    x2.*(268.5520265845071 + y.*(-12019.028203559312 + ...
    y.*(3734.858026725145 + y.*(-2046.7671145057618 + ...
    y.*(465.28655623826234 + (-0.6370820302376359 - ...
    10.650848542359153.*y).*y)))) + ...
    x.*(937.2099110620707 + y.*(588.1802812170108 + ...
    y.*(248.39476522971285 + (-3.871557904936333 - ...
    2.6268019854268356.*y).*y)) + ...
    x.*(-1687.914374187449 + x.*(246.9598888781377 + ...
    x.*(123.59576582457964 - 48.5891069025409.*x)) + ...
    y.*(936.3206544460336 + ...
    y.*(-942.7827304544439 + y.*(369.4389437509002 + ...
    (-33.83664947895248 - 9.987880382780322.*y).*y)))));

%-----
% The above polynomial for pot_enthalpy is the full expression for
% potential entahlpy in terms of SA and pt, obtained from the Gibbs
% function as below. The above polynomial has simply collected like powers
% of x and y so that it is computationally faster than calling the Gibbs
% function twice as is done in the commented code below. When this code
% below is run, the results are identical to calculating pot_enthalpy as
% above, to machine precision.
%
% n0 = 0;
% n1 = 1;
% pr0 = zeros(size(SA));
% pot_enthalpy = gsw_gibbs(n0,n0,n0,SA,pt,pr0) - ...
```

```
%      (273.15 + pt).*gsw_gibbs(n0,n1,n0,SA,pt,pr0);
%
%-----This is the end of the alternative code-----

CT = pot_enthalpy./cp0;

if transposed
    CT = CT.';
end

end
```

## B.6 Partial Calculation of Entropy

```
function entropy_part = gsw_entropy_part(SA,t,p)

%gsw_entropy_part  entropy minus the terms that are a function of only SA
%=====
% This function calculates entropy, except that it does not evaluate any
% terms that are functions of Absolute Salinity alone.  By not calculating
% these terms, which are a function only of Absolute Salinity, several
% unnecessary computations are avoided (including saving the computation
% of a natural logarithm).  These terms are a necessary part of entropy,
% but are not needed when calculating potential temperature from in-situ
% temperature.
%
%
% VERSION NUMBER: 3.0 (29th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
%=====

% These few lines ensure that SA is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

sfac = 0.0248826675584615;          % sfac = 1/(40*(35.16504/35));

x2 = sfac.*SA;
x = sqrt(x2);
y = t.*0.025d0;
z = p.*1d-4;

g03 = z.*(-270.983805184062 + ...
    z.*(776.153611613101 + z.*(-196.51255088122 + (28.9796526294175 -
2.13290083518327.*z).*z))) + ...
    y.*(-24715.571866078 + z.*(2910.0729080936 + ...
    z.*(-1513.116771538718 + z.*(546.959324647056 + z.*(-111.1208127634436 +
8.68841343834394.*z)))) + ...
    y.*(2210.2236124548363 + z.*(-2017.52334943521 + ...
    z.*(1498.081172457456 + z.*(-718.6359919632359 + (146.4037555781616 -
4.9892131862671505.*z).*z))) + ...
    y.*(-592.743745734632 + z.*(1591.873781627888 + ...
    z.*(-1207.261522487504 + (608.785486935364 - 105.4993508931208.*z).*z)) + ...
    y.*(290.12956292128547 + z.*(-973.091553087975 + ...
    z.*(602.603274510125 + z.*(-276.361526170076 + 32.40953340386105.*z))) + ...
    y.*(-113.90630790850321 + y.*(21.35571525415769 - 67.41756835751434.*z) + ...
    z.*(381.06836198507096 + z.*(-133.7383902842754 + 49.023632509086724.*z))))));

g08 = x2.*(z.*(729.116529735046 + ...
    z.*(-343.956902961561 + z.*(124.687671116248 + z.*(-31.656964386073 +
7.04658803315449.*z)))) + ...
    x.*(x.*(y.*(-137.1145018408982 + y.*(148.10030845687618 + y.*(-68.5590309679152 +
12.4848504784754.*y))) - ...
    22.6683558512829.*z) + z.*(-175.292041186547 + (83.1923927801819 -
29.483064349429.*z).*z) + ...
```

```

    y.*(-86.1329351956084 + z.*(766.116132004952 + z.*(-108.3834525034224 +
51.2796974779828.*z)) + ...
    y.*(-30.0682112585625 - 1380.9597954037708.*z + y.*(3.50240264723578 +
938.26075044542.*z)))) + ...
    y.*(1760.062705994408 + y.*(-675.802947790203 + ...
    y.*(365.7041791005036 + y.*(-108.30162043765552 + 12.78101825083098.*y) + ...
    z.*(-1190.914967948748 + (298.904564555024 - 145.9491676006352.*z).*z)) + ...
    z.*(2082.7344423998043 + z.*(-614.668925894709 + (340.685093521782 -
33.3848202979239.*z).*z))) + ...
    z.*(-1721.528607567954 + z.*(674.819060538734 + ...
    z.*(-356.629112415276 + (88.4080716616 - 15.84003094423364.*z).*z)))));

entropy_part = -(g03 + g08).*0.025d0;

end

```

## B.7 Partial Calculation of Entropy at Sea Pressure of Zero

```
function entropy_part_zerop = gsw_entropy_part_zerop(SA,pt0)

% gsw_entropy_part_zerop      entropy_part evaluated at the sea surface
%=====
% This function calculates entropy at a sea pressure of zero, except that
% it does not evaluate any terms that are functions of Absolute Salinity
% alone. By not calculating these terms, which are a function only of
% Absolute Salinity, several unnecessary computations are avoided
% (including saving the computation of a natural logarithm). These terms
% are a necessary part of entropy, but are not needed when calculating
% potential temperature from in-situ temperature.
% The inputs to "gsw_entropy_part_zerop(SA,pt0)" are Absolute Salinity
% and potential temperature with reference sea pressure of zero dbar.
%
% VERSION NUMBER: 3.0 (29th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
%=====

% These few lines ensure that SA is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

sfac = 0.0248826675584615;          % sfac = 1/(40*(35.16504/35));

x2 = sfac.*SA;
x = sqrt(x2);
y = pt0.*0.025d0;

g03 = y.*(-24715.571866078 + y.*(2210.2236124548363 + ...
    y.*(-592.743745734632 + y.*(290.12956292128547 + ...
    y.*(-113.90630790850321 + y.*21.35571525415769))));

g08 = x2.*(x.*(x.*(y.*(-137.1145018408982 + y.*(148.10030845687618 + ...
    y.*(-68.5590309679152 + 12.4848504784754.*y)))) + ...
    y.*(-86.1329351956084 + y.*(-30.0682112585625 + y.*3.50240264723578))) + ...
    y.*(1760.062705994408 + y.*(-675.802947790203 + ...
    y.*(365.7041791005036 + y.*(-108.30162043765552 + 12.78101825083098.*y))));

entropy_part_zerop = -(g03 + g08).*0.025d0;

end
```

## B.8 Second Derivative of the Gibbs Function

```
function gibbs_pt0_pt0 = gsw_gibbs_pt0_pt0(SA,pt0)

% gsw_gibbs_pt0_pt0                                gibbs_tt at (SA,pt,0)
%=====
% This function calculates the second derivative of the specific Gibbs
% function with respect to temperature at zero sea pressure. The inputs
% are Absolute Salinity and potential temperature with reference sea
% pressure of zero dbar. This library function is called by both
% "gsw_pt_from_CT(SA,CT)", "gsw_pt0_from_t(SA,t,p)" and
% "gsw_pt_from_entropy(SA,entropy)".
%
% VERSION NUMBER: 3.0 (29th March, 2011)
% This function is unchanged from version 2.0 (24th September, 2010).
%
%=====

% These few lines ensure that SP is non-negative.
[l_neg_SA] = find(SA < 0);
if ~isempty(l_neg_SA)
    SA(l_neg_SA) = 0;
end

sfac = 0.0248826675584615;          % sfac = 1/(40*(35.16504/35));

x2 = sfac.*SA;
x = sqrt(x2);
y = pt0.*0.025d0;

g03 = -24715.571866078 + ...
    y.*(4420.4472249096725 + ...
    y.*(-1778.231237203896 + ...
    y.*(1160.5182516851419 + ...
    y.*(-569.531539542516 + y.*128.13429152494615))));

g08 = x2.*(1760.062705994408 + x.*(-86.1329351956084 + ...
    x.*(-137.1145018408982 + y.*(296.20061691375236 + ...
    y.*(-205.67709290374563 + 49.9394019139016.*y))) + ...
    y.*(-60.136422517125 + y.*10.50720794170734)) + ...
    y.*(-1351.605895580406 + y.*(1097.1125373015109 + ...
    y.*(-433.20648175062206 + 63.905091254154904.*y))));

gibbs_pt0_pt0 = (g03 + g08).*0.000625;

end
```

## B.9 Absolute Salinity Anomaly Ratio

function [SAAR, in\_ocean] = gsw\_SAAR(p,long,lat)

```
% gsw_SAAR    Absolute Salinity Anomaly Ratio (excluding the Baltic Sea)
%=====
%
% USAGE:
% [SAAR, in_ocean] = gsw_SAAR(p,long,lat)
%
% DESCRIPTION:
% Calculates the Absolute Salinity Anomaly Ratio, SAAR, in the open ocean
% by spatially interpolating the global reference data set of SAAR to the
% location of the seawater sample.
%
% This function uses version 3.0 of the SAAR look up table.
%
% The Absolute Salinity Anomaly Ratio in the Baltic Sea is evaluated
% separately, since it is a function of Practical Salinity, not of space.
% The present function returns a SAAR of zero for data in the Baltic Sea.
% The correct way of calculating Absolute Salinity in the Baltic Sea is by
% calling gsw_SA_from_SP.
%
% INPUT:
% p    = sea pressure                [ dbar ]
%       ( i.e. absolute pressure - 10.1325 dbar )
% long = Longitude in decimal degrees [ 0 ... +360 ]
%       or [ -180 ... +180 ]
% lat  = Latitude in decimal degrees north [ -90 ... +90 ]
%
% p, long & lat need to be vectors and have the same dimensions.
%
% OUTPUT:
% SAAR    = Absolute Salinity Anomaly Ratio [ unitless ]
% in_ocean = 0, if long and lat are a long way from the ocean
%           = 1, if long and lat are in the ocean
% Note. This flag is only set when the observation is well and truly on
% dry land; often the warning flag is not set until one is several
% hundred kilometres inland from the coast.
%
% AUTHOR:
% David Jackett                [ help_gsw@csiro.au ]
%
% MODIFIED:
% Paul Barker and Trevor McDougall
%
% VERSION NUMBER: 3.0 (31st May, 2011)
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
%
% McDougall, T.J., D.R. Jackett and F.J. Millero, 2010: An algorithm
% for estimating Absolute Salinity in the global ocean. Submitted to
```



```
% Ocean Science. A preliminary version is available at Ocean Sci.
% Discuss., 6, 215-242.
% http://www.ocean-sci-discuss.net/6/215/2009/osd-6-215-2009-print.pdf
%
% The software is available from http://www.TEOS-10.org
%
%=====

%-----
% Check variables and resize if necessary
%-----

if ~(nargin == 3)
    error('gsw_SAAR: Requires three inputs')
end %if

[mp,np] = size(p);
[mla,nla] = size(lat);
[mlo,nlo] = size(long);

if (mp ~= mla) | (mp ~= mlo) | (np ~= nla) | (np ~= nlo)
    error('gsw_SAAR: Inputs need be of the same size')
end %if

if ~isempty(find(p < -1.5))
    error('gsw_SAAR: pressure needs to be positive')
end

[lpn] = find(p < 0 & p > -1.5);
if ~isempty(lpn)
    p(lpn) = 0;
end

%-----
% Start of the calculation (extracting from a look up table)
%-----

gsw_data = 'gsw_data_v3_0.mat';

gsw_data_file = which(gsw_data);

load (gsw_data_file,'SAAR_ref','lats_ref','longs_ref','p_ref', ...
    'ndepth_ref');

nx = length(longs_ref);
ny = length(lats_ref);
nz = length(p_ref);
n0 = length(p);

dlongs_ref = longs_ref(2) - longs_ref(1);
dlats_ref = lats_ref(2) - lats_ref(1);

indsx0 = floor(1 + (nx-1)*(long - longs_ref(1))./(longs_ref(nx) - longs_ref(1)));
indsx0 = indsx0(:);
inds = find(indsx0 == nx);
indsx0(inds) = nx - 1;
```

```
indsy0 = floor(1 + (ny-1)*(lat - lats_ref(1))./(lats_ref(ny) - lats_ref(1)));
indsy0 = indsy0(:);
inds = find(indsy0 == ny);
indsy0(inds) = ny - 1;

indsz0 = sum(ones(nz,1)*p(:)' >= p_ref(:)*ones(1,n0));
indsz0 = indsz0(:); % adjust in the vertical

indsn1 = sub2ind([ny,nx],indsy0,indsx0); % casts containing data
indsn2 = sub2ind([ny,nx],indsy0,indsx0+1);
indsn3 = sub2ind([ny,nx],indsy0+1,indsx0+1);
indsn4 = sub2ind([ny,nx],indsy0+1,indsx0);

nmax = max([ndepth_ref(indsn1)';ndepth_ref(indsn2)';ndepth_ref(indsn3)';ndepth_ref(indsn4)']);

inds1 = find(indsz0(:)' > nmax); % casts deeper than GK maximum

p(inds1) = p_ref(nmax(inds1)); % have reset p here so have to reset indsz0

indsz0 = sum(ones(nz,1)*p(:)' >= p_ref(:)*ones(1,n0));
indsz0 = indsz0(:);
inds = find(indsz0 == nz);
indsz0(inds) = nz - 1;

inds0 = sub2ind([nz,ny,nx],indsz0,indsy0,indsx0);

data_indices = [indsx0,indsy0,indsz0,inds0];
data_inds = data_indices(:,3);

r1 = (long(:) - longs_ref(indsx0))./(longs_ref(indsx0+1) - longs_ref(indsx0));
s1 = (lat(:) - lats_ref(indsy0))./(lats_ref(indsy0+1) - lats_ref(indsy0));
t1 = (p(:) - p_ref(indsz0))./(p_ref(indsz0+1) - p_ref(indsz0));

nksum = 0;
no_levels_missing = 0;

sa_upper = nan(size(data_inds));
sa_lower = nan(size(data_inds));
SAAR = nan(size(data_inds));
in_ocean = ones(size(SAAR));

for k = 1:nz-1

    inds_k = find(indsz0 == k);
    nk = length(inds_k);

    if nk>0
        nksum = nksum+nk;
        indsx = indsx0(inds_k);
        indsy = indsy0(inds_k);
        indsz = k*ones(size(indsx));
        inds_di = find(data_inds == k); % level k interpolation
        saar = nan(4,n0);
        indsn1 = sub2ind([nz,ny,nx],indsz,indsy,indsx);
        saar(1,inds_k) = SAAR_ref(indsn1);
```

```
inds2 = sub2ind([nz,ny,nx], indsz, indsy, indsx+1);
saar(2,inds_k) = SAAR_ref(inds2);          % indsz0 + ny*nz
inds3 = sub2ind([nz,ny,nx], indsz, indsy+1, indsx+1);
saar(3,inds_k) = SAAR_ref(inds3);          % indsz0 + ny*nz + nz
inds4 = sub2ind([nz,ny,nx], indsz, indsy+1, indsx);
saar(4,inds_k) = SAAR_ref(inds4);          % indsz0 + nz

inds = find(260<=long(:) & long(:)<=295.217 & ...
    0<=lat(:) & lat(:)<=19.55 & indsz0(:)==k);
if ~isempty(inds)
    saar(:,inds) = gsw_saar_add_barrier(saar(:,inds),long(inds), ...
        lat(inds),longs_ref(indsx0(inds)),lats_ref(indsy0(inds)),dlongs_ref,dlats_ref);
end

inds = find(isnan(sum(saar))' & indsz0==k);
if ~isempty(inds)
    saar(:,inds) = gsw_saar_add_mean(saar(:,inds));
end

sa_upper(inds_di) = (1-s1(inds_di)).*(saar(1,inds_k)' + ...
    r1(inds_di).*(saar(2,inds_k)'-saar(1,inds_k)')) + ...
    s1(inds_di).*(saar(4,inds_k)' + ...
    r1(inds_di).*(saar(3,inds_k)'-saar(4,inds_k)')); % level k+1 interpolation

saar = nan(4,n0);
inds1 = sub2ind([nz,ny,nx], indsz+1, indsy, indsx);
saar(1,inds_k) = SAAR_ref(inds1);
inds2 = sub2ind([nz,ny,nx], indsz+1, indsy, indsx+1);
saar(2,inds_k) = SAAR_ref(inds2);          % indsz1 + ny*nz
inds3 = sub2ind([nz,ny,nx], indsz+1, indsy+1, indsx+1);
saar(3,inds_k) = SAAR_ref(inds3);          % indsz1 + ny*nz + nz
inds4 = sub2ind([nz,ny,nx], indsz+1, indsy+1, indsx);
saar(4,inds_k) = SAAR_ref(inds4);          % indsz1 + nz

inds = find(260<=long(:) & long(:)<=295.217 & ...
    0<=lat(:) & lat(:)<=19.55 & indsz0(:)==k);
if ~isempty(inds)
    saar(:,inds) = gsw_saar_add_barrier(saar(:,inds),long(inds), ...
        lat(inds),longs_ref(indsx0(inds)),lats_ref(indsy0(inds)),dlongs_ref,dlats_ref);
end

inds = find(isnan(sum(saar))' & indsz0==k);

if ~isempty(inds)
    saar(:,inds) = gsw_saar_add_mean(saar(:,inds));
end

sa_lower(inds_di) = (1-s1(inds_di)).*(saar(1,inds_k)' + ...
    r1(inds_di).*(saar(2,inds_k)'-saar(1,inds_k)')) + ...
    s1(inds_di).*(saar(4,inds_k)' + ...
    r1(inds_di).*(saar(3,inds_k)'-saar(4,inds_k)'));

inds_different = find(isfinite(sa_upper(inds_di)) & isnan(sa_lower(inds_di)));

if ~isempty(inds_different)
    sa_lower(inds_di(inds_different)) = sa_upper(inds_di(inds_different));
```

```
end

    SAAR(inds_di) = sa_upper(inds_di) + t1(inds_di).*(sa_lower(inds_di) - sa_upper(inds_di));

else
    no_levels_missing = no_levels_missing + 1;
end
end

inds = find(~isfinite(SAAR));
SAAR(inds) = 0;

in_ocean(inds) = 0;

end

%#####

function SAAR = gsw_saar_add_mean(saar)

% gsw_saar_add_mean
%=====
%
% USAGE:
% SAAR = gsw_saar_add_mean(saar)
%
% DESCRIPTION:
% Replaces NaN's with nanmean of the 4 adjacent neighbours
%
% INPUT:
% saar = Absolute Salinity Anomaly Ratio of the 4 adjacent neighbours
%           [ unitless ]
%
% OUTPUT:
% SAAR = nanmean of the 4 adjacent neighbours          [ unitless ]
%
% AUTHOR:
% David Jackett
%
% MODIFIED:
% Paul Barker and Trevor McDougall
%
% VERSION NUMBER: 3.0
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
%
% McDougall, T. J., D. R. Jackett and F. J. Millero, 2010: An algorithm
% for estimating Absolute Salinity in the global ocean. Submitted to
% Ocean Science, a preliminary version is available at Ocean Sci.
% Discuss., 6, 215-242.
% http://www.ocean-sci-discuss.net/6/215/2009/osd-6-215-2009-print.pdf
% and the computer software is available from http://www.TEOS-10.org
```

```
%
%=====

saar_mean = mean(saar);
inds_nan = find(isnan(saar_mean));
no_nan = length(inds_nan);

for kk = 1:no_nan
    col = inds_nan(kk);
    inds_kk = find(isnan(saar(:,col)));
    [lnn] = find(~isnan(saar(:,col)));
    if ~isempty(lnn)
        saar(inds_kk,col) = mean(saar(lnn,col));
    end
end

SAAR = saar;

end

%#####

function SAAR = gsw_saar_add_barrier(saar,long,lat,longs_ref,lats_ref,dlongs_ref,dlats_ref)

% gsw_saar_add_barrier
%=====
%
% USAGE:
% SAAR = gsw_saar_add_barrier(saar,long,lat,longs_ref,lats_ref,dlongs_ref,dlats_ref)
%
% DESCRIPTION:
% Adds a barrier through Central America (Panama) and then averages
% over the appropriate side of the barrier
%
% INPUT:
% saar      = Absolute Salinity Anomaly Ratio          [ unitless ]
% long      = Longitudes of data in decimal degrees east    [ 0 ... +360 ]
% lat       = Latitudes of data in decimal degrees north   [ -90 ... +90 ]
% longs_ref = Longitudes of regular grid in decimal degrees east    [ 0 ... +360 ]
% lats_ref  = Latitudes of regular grid in decimal degrees north   [ -90 ... +90 ]
% dlongs_ref = Longitude difference of regular grid in decimal degrees [ deg longitude ]
% dlats_ref = Latitude difference of regular grid in decimal degrees [ deg latitude ]
%
% OUTPUT:
% SAAR      = Absolute Salinity Anomaly Ratio          [ unitless ]
%
% AUTHOR:
% David Jackett
%
% MODIFIED:
% Paul Barker and Trevor McDougall
%
% VERSION NUMBER: 3.0
%
% REFERENCES:
% IOC, SCOR and IAPSO, 2010: The international thermodynamic equation of
```

```
% seawater - 2010: Calculation and use of thermodynamic properties.
% Intergovernmental Oceanographic Commission, Manuals and Guides No. 56,
% UNESCO (English), 196 pp. Available from http://www.TEOS-10.org
%
% McDougall, T. J., D. R. Jackett and F. J. Millero, 2010: An algorithm
% for estimating Absolute Salinity in the global ocean. Submitted to
% Ocean Science, a preliminary version is available at Ocean Sci.
% Discuss., 6, 215-242.
% http://www.ocean-sci-discuss.net/6/215/2009/osd-6-215-2009-print.pdf
% and the computer software is available from http://www.TEOS-10.org
%
%=====

longs_pan = [260.0000 272.5900 276.5000 278.6500 280.7300 295.2170];

lats_pan = [ 19.5500 13.9700 9.6000 8.1000 9.3300 0];

lats_lines0 = interp1(longs_pan,lats_pan,long);

lats_lines1 = interp1(longs_pan,lats_pan,longs_ref);
lats_lines2 = interp1(longs_pan,lats_pan,(longs_ref+dlongs_ref));

for k0 = 1:length(long)
    if lats_lines0(k0) <= lat(k0)
        above_line0 = 1;
    else
        above_line0 = 0;
    end
    if lats_lines1(k0) <= lats_ref(k0)
        above_line(1) = 1;
    else
        above_line(1) = 0;
    end
    if lats_lines1(k0) <= (lats_ref(k0) + dlats_ref)
        above_line(4) = 1;
    else
        above_line(4) = 0;
    end
    if lats_lines2(k0) <= lats_ref(k0)
        above_line(2) = 1;
    else
        above_line(2) = 0;
    end
    if lats_lines2(k0) <= (lats_ref(k0) + dlats_ref)
        above_line(3) = 1;
    else
        above_line(3) = 0;
    end
    inds = find(above_line ~= above_line0);    % indices of different sides of CA line
    saar(inds,k0) = nan;
end

saar_mean = mean(saar);
inds_nan = find(isnan(saar_mean));
no_nan = length(inds_nan);
```

```
for kk = 1:no_nan
    col = inds_nan(kk);
    inds_kk = find(isnan(saar(:,col)));
    [Inn] = find(~isnan(saar(:,col)));
    if ~isempty(Inn)
        saar(inds_kk,col) = mean(saar(Inn,col));
    end
end

SAAR = saar;

end
```

## **Appendix C            Output Accuracy**

The DOCONCS accuracy requirement is stated as:

“Upon initial deployment, the instrument shall measure dissolved O<sub>2</sub> concentrations with an accuracy within  $\pm 2\%$  of the value provided by a Winkler titration of a corresponding water sample. <L2-SR-RQ-3495, L4-CG-IP-RQ-182, L4-RSN-IP-RQ-312>”

This measurement will be performed by the operator and logged with CI prior to deployment.

Accuracy of the L2 Dissolved Oxygen data product is dependent on the accuracy of the L1 input to the computation, which for the Aanderaa optodes is 8 $\mu$ M for readings below 160 $\mu$ M, and >8 $\mu$ m (5%) for readings above 160 $\mu$ M.



## **Appendix D            Sensor Calibration Effects**

Prior to deployment, instrument specific calibration takes place using the following steps (from the Aanderaa Optode Manual):

1) Each batch of sensing foils is delivered with calibration data describing the behavior with respect to oxygen concentration and temperature. When changing the sensing foil the following 28 coefficients must be updated:

FoilCoeffA0 through FoilCoeffA13 and  
FoilCoeffB0 through FoilCoeffB13

These coefficients are found in the Calibration Certificate for the Sensing Foil 3733/4793; refer to chapter 5.3.1 in the Aanderaa manual for instructions regarding changing foil coefficients.

2) Secondly, a two-point calibration must be done, one in air-saturated water, and one in a zero-oxygen solution. An air-saturated solution is obtained by bubbling air through freshwater in a glass using a standard aquarium pump. For a more efficient bubbling, it is recommended to use a bubble diffuser. The water should be allowed to achieve temperature stability for at least 1 hour. The zero-oxygen solution can be obtained by dissolving 5g of sodium sulphite ( $\text{Na}_2\text{SO}_3$ ) in 500ml of fresh water. This calibration compensates for individual sensor and foil variations. The two values are entered into the instrument through the RS232 interface; refer to chapter 5.3.1 in the Aanderaa manual for calibration instructions.

## Appendix E Collocated CTD Lookup Table for DOCONCS Data

Consult the following table to determine the appropriate CTD instrument from which to draw data in order to compute the corrected DOCONCS data product. Each pair of collocated instruments is highlighted. (Note that the last four digits of the References Designator may change once the new instrument series and sequence numbers are updated, estimated July 2012.)

Reference Designator	Subsite Name	Node Type	Instrument Class
CE01ISSM-MF004-01-DOSTA0999	Endurance OR Inshore Surface Mooring	MF (Multi-Function Node)	DOSTA
CE01ISSM-MF005-01-CTDBP0999	Endurance OR Inshore Surface Mooring	MF (Multi-Function Node)	CTDBP
CE01ISSM-RI002-01-DOSTA0999	Endurance OR Inshore Surface Mooring	RI (Mooring Riser)	DOSTA
CE01ISSM-RI003-01-CTDBP0999	Endurance OR Inshore Surface Mooring	RI (Mooring Riser)	CTDBP
CE02SHBP-LJ01D-02-CTDBPD102	Endurance OR Shelf Benthic Pkg	LJ (LP Jbox)	CTDBP
CE02SHBP-LJ01D-02-DOSTA0102	Endurance OR Shelf Benthic Pkg	LJ (LP Jbox)	DOSTA
CE02SHSM-RI002-01-DOSTA0999	Endurance OR Shelf Surface Mooring	RI (Mooring Riser)	DOSTA
CE02SHSM-RI002-05-CTDBP0999	Endurance OR Shelf Surface Mooring	RI (Mooring Riser)	CTDBP
CE04OSBP-LJ01C-02-CTDBPE102	Endurance OR Offshore Benthic Pkg	LJ (LP Jbox)	CTDBP
CE04OSBP-LJ01C-02-DOSTA0102	Endurance OR Offshore Benthic Pkg	LJ (LP Jbox)	DOSTA
CE04OSSM-RI002-01-DOSTA0999	Endurance OR Offshore Surface Mooring	RI (Mooring Riser)	DOSTA
CE04OSSM-RI002-05-CTDBP0999	Endurance OR Offshore Surface Mooring	RI (Mooring Riser)	CTDBP
CE05MOAS-GL001-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL001-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE05MOAS-GL002-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL002-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE05MOAS-GL003-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL003-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE05MOAS-GL004-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL004-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE05MOAS-GL005-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL005-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE05MOAS-GL006-04-DOSTA0999	Endurance Mobile Assets	GL (Gliders)	DOSTA
CE05MOAS-GL006-05-CTDGV0999	Endurance Mobile Assets	GL (Gliders)	CTDGV
CE06ISSM-MF004-01-DOSTA0999	Endurance WA Inshore Surface Mooring	MF (Multi-Function Node)	DOSTA
CE06ISSM-MF005-01-CTDBP0999	Endurance WA Inshore Surface Mooring	MF (Multi-Function Node)	CTDBP
CE06ISSM-RI002-01-DOSTA0999	Endurance WA Inshore Surface Mooring	RI (Mooring Riser)	DOSTA
CE06ISSM-RI003-01-CTDBP0999	Endurance WA Inshore Surface Mooring	RI (Mooring Riser)	CTDBP
CE07SHSM-MF004-01-DOSTA0999	Endurance WA Shelf Surface Mooring	MF (Multi-Function Node)	DOSTA
CE07SHSM-MF004-05-CTDBP0999	Endurance WA Shelf Surface Mooring	MF (Multi-Function Node)	CTDBP
CE07SHSM-RI002-01-DOSTA0999	Endurance WA Shelf Surface Mooring	RI (Mooring Riser)	DOSTA
CE07SHSM-RI002-05-CTDBP0999	Endurance WA Shelf Surface Mooring	RI (Mooring Riser)	CTDBP
CE09OSSM-MF004-01-DOSTA0999	Endurance WA Offshore Surface	MF (Multi-Function	DOSTA

Data Product Specification for Oxygen Concentration from "Stable" Instruments

	Mooring	Node)	
CE09OSSM-MF005-01-CTDBP0999	Endurance WA Offshore Surface Mooring	MF (Multi-Function Node)	CTDBP
CE09OSSM-RI002-01-DOSTA0999	Endurance WA Offshore Surface Mooring	RI (Mooring Riser)	DOSTA
CE09OSSM-RI002-05-CTDBP0999	Endurance WA Offshore Surface Mooring	RI (Mooring Riser)	CTDBP
CP01CNSM-MF004-03-DOSTA0999	Pioneer Central Surface Mooring	MF (Multi-Function Node)	DOSTA
CP01CNSM-MF005-02-CTDBP0999	Pioneer Central Surface Mooring	MF (Multi-Function Node)	CTDBP
CP01CNSM-RI002-03-CTDBP0999	Pioneer Central Surface Mooring	RI (Mooring Riser)	CTDBP
CP01CNSM-RI003-02-DOSTA0999	Pioneer Central Surface Mooring	RI (Mooring Riser)	DOSTA
CP03ISSM-MF004-01-DOSTA0999	Pioneer Inshore Surface Mooring	MF (Multi-Function Node)	DOSTA
CP03ISSM-MF004-04-CTDBP0999	Pioneer Inshore Surface Mooring	MF (Multi-Function Node)	CTDBP
CP03ISSM-RI002-04-DOSTA0999	Pioneer Inshore Surface Mooring	RI (Mooring Riser)	DOSTA
CP03ISSM-RI003-04-CTDBP0999	Pioneer Inshore Surface Mooring	RI (Mooring Riser)	CTDBP
CP04OSSM-MF004-01-DOSTA0999	Pioneer Offshore Surface Mooring	MF (Multi-Function Node)	DOSTA
CP04OSSM-MF004-04-CTDBP0999	Pioneer Offshore Surface Mooring	MF (Multi-Function Node)	CTDBP
CP04OSSM-RI002-04-DOSTA0999	Pioneer Offshore Surface Mooring	RI (Mooring Riser)	DOSTA
CP04OSSM-RI003-04-CTDBP0999	Pioneer Offshore Surface Mooring	RI (Mooring Riser)	CTDBP
CP05MOAS-AV001-02-DOSTA0999	Pioneer Mobile Assets	AV (AUV)	DOSTA
CP05MOAS-AV001-03-CTDAV0999	Pioneer Mobile Assets	AV (AUV)	CTDAV
CP05MOAS-AV002-02-DOSTA0999	Pioneer Mobile Assets	AV (AUV)	DOSTA
CP05MOAS-AV002-03-CTDAV0999	Pioneer Mobile Assets	AV (AUV)	CTDAV
CP05MOAS-AV003-02-DOSTA0999	Pioneer Mobile Assets	AV (AUV)	DOSTA
CP05MOAS-AV003-03-CTDAV0999	Pioneer Mobile Assets	AV (AUV)	CTDAV
CP05MOAS-GL001-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL001-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
CP05MOAS-GL002-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL002-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
CP05MOAS-GL003-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL003-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
CP05MOAS-GL004-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL004-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
CP05MOAS-GL005-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL005-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
CP05MOAS-GL006-03-CTDGV0999	Pioneer Mobile Assets	GL (Gliders)	CTDGV
CP05MOAS-GL006-04-DOSTA0999	Pioneer Mobile Assets	GL (Gliders)	DOSTA
GA02HYPM-SP001-03-DOSTA0999	Argentine Basin Hybrid Profiler	SP (Surface-Piercing Profiler)	DOSTA
GA02HYPM-SP001-04-CTDPF0999	Argentine Basin Hybrid Profiler	SP (Surface-Piercing Profiler)	CTDPF
GA02HYPM-WF002-03-DOSTA0999	Argentine Basin Hybrid Profiler	WF (Wire-Following Profiler)	DOSTA
GA02HYPM-WF002-04-CTDPF0999	Argentine Basin Hybrid Profiler	WF (Wire-Following Profiler)	CTDPF
GA03FLMA-RI001-03-DOSTA0999	Argentine Basin Flanking Mooring A	RI (Mooring Riser)	DOSTA
GA03FLMA-RI001-07-CTDMO0999	Argentine Basin Flanking Mooring A	RI (Mooring Riser)	CTDMO
GA03FLMB-RI001-03-DOSTA0999	Argentine Basin Flanking Mooring B	RI (Mooring Riser)	DOSTA
GA03FLMB-RI001-07-CTDMO0999	Argentine Basin Flanking Mooring B	RI (Mooring Riser)	CTDMO
GA05MOAS-GL001-02-DOSTA0999	Argentine Basin Mobile Assets	GL (Gliders)	DOSTA
GA05MOAS-GL001-04-CTDGV0999	Argentine Basin Mobile Assets	GL (Gliders)	CTDGV
GA05MOAS-GL002-02-DOSTA0999	Argentine Basin Mobile Assets	GL (Gliders)	DOSTA

# Data Product Specification for Oxygen Concentration from "Stable" Instruments

GA05MOAS-GL002-04-CTDGV0999	Argentine Basin Mobile Assets	GL (Gliders)	CTDGV
GA05MOAS-GL003-02-DOSTA0999	Argentine Basin Mobile Assets	GL (Gliders)	DOSTA
GA05MOAS-GL003-04-CTDGV0999	Argentine Basin Mobile Assets	GL (Gliders)	CTDGV
GI02HYPM-SP001-03-DOSTA0999	Irminger Sea Hybrid Profiler	SP (Surface-Piercing Profiler)	DOSTA
GI02HYPM-SP001-04-CTDPF0999	Irminger Sea Hybrid Profiler	SP (Surface-Piercing Profiler)	CTDPF
GI02HYPM-WF002-03-DOSTA0999	Irminger Sea Hybrid Profiler	WF (Wire-Following Profiler)	DOSTA
GI02HYPM-WF002-04-CTDPF0999	Irminger Sea Hybrid Profiler	WF (Wire-Following Profiler)	CTDPF
GI03FLMA-RI001-03-DOSTA0999	Irminger Sea Flanking Mooring A	RI (Mooring Riser)	DOSTA
GI03FLMA-RI001-07-CTDMO0999	Irminger Sea Flanking Mooring A	RI (Mooring Riser)	CTDMO
GI03FLMB-RI001-03-DOSTA0999	Irminger Sea Flanking Mooring B	RI (Mooring Riser)	DOSTA
GI03FLMB-RI001-07-CTDMO0999	Irminger Sea Flanking Mooring B	RI (Mooring Riser)	CTDMO
GI05MOAS-GL001-02-DOSTA0999	Irminger Sea Mobile Assets	GL (Gliders)	DOSTA
GI05MOAS-GL001-04-CTDGV0999	Irminger Sea Mobile Assets	GL (Gliders)	CTDGV
GI05MOAS-GL002-02-DOSTA0999	Irminger Sea Mobile Assets	GL (Gliders)	DOSTA
GI05MOAS-GL002-04-CTDGV0999	Irminger Sea Mobile Assets	GL (Gliders)	CTDGV
GI05MOAS-GL003-02-DOSTA0999	Irminger Sea Mobile Assets	GL (Gliders)	DOSTA
GI05MOAS-GL003-04-CTDGV0999	Irminger Sea Mobile Assets	GL (Gliders)	CTDGV
GP02HYPM-SP001-03-DOSTA0999	Station Papa Hybrid Profiler	SP (Surface-Piercing Profiler)	DOSTA
GP02HYPM-SP001-04-CTDPF0999	Station Papa Hybrid Profiler	SP (Surface-Piercing Profiler)	CTDPF
GP02HYPM-WF002-03-DOSTA0999	Station Papa Hybrid Profiler	WF (Wire-Following Profiler)	DOSTA
GP02HYPM-WF002-04-CTDPF0999	Station Papa Hybrid Profiler	WF (Wire-Following Profiler)	CTDPF
GP03FLMA-RI001-03-DOSTA0999	Station Papa Flanking Mooring A	RI (Mooring Riser)	DOSTA
GP03FLMA-RI001-07-CTDMO0999	Station Papa Flanking Mooring A	RI (Mooring Riser)	CTDMO
GP03FLMB-RI001-03-DOSTA0999	Station Papa Flanking Mooring B	RI (Mooring Riser)	DOSTA
GP03FLMB-RI001-07-CTDMO0999	Station Papa Flanking Mooring B	RI (Mooring Riser)	CTDMO
GP05MOAS-GL001-02-DOSTA0999	Station Papa Mobile Assets	GL (Gliders)	DOSTA
GP05MOAS-GL001-04-CTDGV0999	Station Papa Mobile Assets	GL (Gliders)	CTDGV
GP05MOAS-GL002-02-DOSTA0999	Station Papa Mobile Assets	GL (Gliders)	DOSTA
GP05MOAS-GL002-04-CTDGV0999	Station Papa Mobile Assets	GL (Gliders)	CTDGV
GP05MOAS-GL003-02-DOSTA0999	Station Papa Mobile Assets	GL (Gliders)	DOSTA
GP05MOAS-GL003-04-CTDGV0999	Station Papa Mobile Assets	GL (Gliders)	CTDGV
GS02HYPM-SP001-03-DOSTA0999	Southern Ocean Hybrid Profiler	SP (Surface-Piercing Profiler)	DOSTA
GS02HYPM-SP001-04-CTDPF0999	Southern Ocean Hybrid Profiler	SP (Surface-Piercing Profiler)	CTDPF
GS02HYPM-WF002-03-DOSTA0999	Southern Ocean Hybrid Profiler	WF (Wire-Following Profiler)	DOSTA
GS02HYPM-WF002-04-CTDPF0999	Southern Ocean Hybrid Profiler	WF (Wire-Following Profiler)	CTDPF
GS03FLMA-RI001-03-DOSTA0999	Southern Ocean Flanking Mooring A	RI (Mooring Riser)	DOSTA
GS03FLMA-RI001-07-CTDMO0999	Southern Ocean Flanking Mooring A	RI (Mooring Riser)	CTDMO
GS03FLMB-RI001-03-DOSTA0999	Southern Ocean Flanking Mooring B	RI (Mooring Riser)	DOSTA
GS03FLMB-RI001-07-CTDMO0999	Southern Ocean Flanking Mooring B	RI (Mooring Riser)	CTDMO
GS05MOAS-GL001-02-DOSTA0999	Southern Ocean Mobile Assets	GL (Gliders)	DOSTA
GS05MOAS-GL001-04-CTDGV0999	Southern Ocean Mobile Assets	GL (Gliders)	CTDGV
GS05MOAS-GL002-02-DOSTA0999	Southern Ocean Mobile Assets	GL (Gliders)	DOSTA
GS05MOAS-GL002-04-CTDGV0999	Southern Ocean Mobile Assets	GL (Gliders)	CTDGV
GS05MOAS-GL003-02-DOSTA0999	Southern Ocean Mobile Assets	GL (Gliders)	DOSTA
GS05MOAS-GL003-04-CTDGV0999	Southern Ocean Mobile Assets	GL (Gliders)	CTDGV

RS01SBVM-DP01A-01-CTDPFA104	Slope Base Vertical Mooring	DP (Deep Profiler)	CTDPF
RS01SBVM-DP01A-01-DOSTAA104	Slope Base Vertical Mooring	DP (Deep Profiler)	DOSTA
RS01SBVM-LJ01A-06-CTDPFA101	Slope Base Vertical Mooring	LJ (LP Jbox)	CTDPF
RS01SBVM-LJ01A-06-DOSTAA101	Slope Base Vertical Mooring	LJ (LP Jbox)	DOSTA
RS01SBVM-PC01A-08-CTDPFA103	Slope Base Vertical Mooring	PC (Platform Interface Controller)	CTDPF
RS01SBVM-PC01A-08-DOSTAA103	Slope Base Vertical Mooring	PC (Platform Interface Controller)	DOSTA
RS03AXVM-DP03A-05-CTDPFA304	Axial Mooring	DP (Deep Profiler)	CTDPF
RS03AXVM-DP03A-05-DOSTAA304	Axial Mooring	DP (Deep Profiler)	DOSTA
RS03AXVM-LJ03A-06-CTDPFA301	Axial Mooring	LJ (LP Jbox)	CTDPF
RS03AXVM-LJ03A-06-DOSTAA301	Axial Mooring	LJ (LP Jbox)	DOSTA
RS03AXVM-PC03A-08-CTDPFA303	Axial Mooring	PC (Platform Interface Controller)	CTDPF
RS03AXVM-PC03A-08-DOSTAA303	Axial Mooring	PC (Platform Interface Controller)	DOSTA