

Supplement to the manuscript:

**Numerical Implementation and Oceanographic Application of  
the Thermodynamic Potentials of Liquid Water, Water Vapour, Ice, Seawater  
and Humid Air.  
Part 2: The Library Routines**

D.G. Wright, R. Feistel, J.H. Reissmann, K. Miyagawa, D.R. Jackett, W. Wagner,  
U. Overhoff, C. Guder, A. Feistel and G.M. Marion

Correspondence to:

R. Feistel ([Rainer.Feistel@IO-Warnemuende.de](mailto:Rainer.Feistel@IO-Warnemuende.de))

Table S1: The SIA library contents. Module names are in bold and user-accessible routines are in plain type. The entries under Uses are modules that are used by one or more of the local routines and Public Routines are available from other modules and from user routines. The bracketed module names under Uses are used through nested use associations. The underlined routines are thermodynamic potential functions including first and second derivatives. The bracketed numbers preceding module names are used to identify the cell blocks and give the related supplementary table numbers to consult for additional information. For example, the cell containing the module **Convert\_0** with (S2) preceding the module name is referred to as Table S1 (Block S2) and Table S2 provides additional information on this module.

<b>Level 0 routines</b>			
<b>Constants_0</b> <u>Public Parameter Values</u> celsius_temperature_si check_limits cp_chempot_si cp_density_si cp_pressure_si cp_temperature_si dry_air_dmax dry_air_dmin dry_air_tmax dry_air_tmin errorreturn flu_dmax flu_dmin flu_tmax flu_tmin gas_constant_air_si gas_constant_air_L2000 gas_constant_molar_si gas_constant_molar_L2000 gas_constant_h2O_si gas_constant_h2O_iapws95 ice_pmax ice_pmin ice_tmax ice_tmin isextension2010 isok	<b>Constants_0 (Cont'd)</b> <u>Parameter Values (cont'd)</u> mix_air_dmax mix_air_dmin mix_air_tmax mix_air_tmin molar_mass_air_si molar_mass_air_l2000 molar_mass_h2o_si molar_mass_seasalt_si pi sal_pmax sal_pmin sal_smax sal_smin sal_tmax sal_tmin sealevel_pressure_si so_salinity_si so_temperature_si so_pressure_si tp_density_ice_iapws95_si tp_density_liq_iapws95_si tp_density_vap_iapws95_si tp_enthalpy_ice_si tp_enthalpy_vap_si tp_pressure_exp_si tp_pressure_iapws95_si tp_temperature_si	<b>Maths_0</b> <u>Uses</u> constants_0 <u>Public Routines</u> get_cubicroots matrix_solve	<b>(S2) Convert_0</b> <u>Uses</u> constants_0 <u>Public Routines</u> air_massfraction_air_si air_massfraction_vap_si air_molar_mass_si air_molfraction_air_si air_molfraction_vap_si asal_from_psal psal_from_asal

<b>Level 1 routines</b>			
<b>(S3) Flu_1 (IAPWS95)</b> <u>Uses</u> constants_0 <u>Public Routines</u> chk_iapws95_table6 chk_iapws95_table7 flu_f_si	<b>(S4) Ice_1 (IAPWS06)</b> <u>Uses</u> constants_0 <u>Public Routines</u> chk_iapws06_table6 ice_q_si	<b>(S5) Sal_1 (IAPWS08)</b> <u>Uses</u> constants_0 <u>Public Routines</u> sal_q_term_si	<b>(S6) Air_1</b> <u>Uses</u> constants_0 <u>Public Routines</u> air_baw_m3mol air_caaw_m6mol2 air_caww_m6mol2 dry_f_si dry_init_clear dry_init_Lemmon2000

## Level 2 routines

<p><b>(S7) Flu_2</b></p> <p><u>Uses</u> constants_0, flu_1</p> <p><u>Public Routines</u> flu_cp_si flu_cv_si flu_enthalpy_si flu_entropy_si flu_expansion_si flu_gibbs_energy_si flu_internal_energy_si flu_kappa_s_si flu_kappa_t_si flu_lapserate_si flu_pressure_si flu_soundspeed_si</p>	<p><b>(S8) Ice_2</b></p> <p><u>Uses</u> constants_0, ice_1</p> <p><u>Public Routines</u> ice_chempot_si ice_cp_si ice_density_si ice_enthalpy_si ice_entropy_si ice_expansion_si ice_helmholtz_energy_si ice_internal_energy_si ice_kappa_s_si ice_kappa_t_si ice_lapserate_si ice_p_coefficient_si ice_specific_volume_si</p>	<p><b>(S9) Sal_2</b></p> <p><u>Uses</u> constants_0, sal_1</p> <p><u>Public Routines</u> sal_act_coeff_si sal_act_potential_si sal_activity_w_si sal_chem_coeff_si sal_chempot_h2o_si sal_chempot_rel_si sal_dilution_si <u>sal_g_si</u> sal_mixenthalpy_si sal_mixentropy_si sal_mixvolume_si sal_molality_si sal_osm_coeff_si sal_saltenthalpy_si sal_saltentropy_si sal_saltvolume_si</p>	<p><b>(S10) Air_2</b></p> <p><u>Uses</u> constants_0, flu_1, air_1</p> <p><u>Public Routines</u> <u>air_f_si</u> air_f_cp_si air_f_cv_si air_f_enthalpy_si air_f_entropy_si air_f_expansion_si air_f_gibbs_energy_si air_f_internal_energy_si air_f_kappa_s_si air_f_kappa_t_si air_f_lapserate_si <u>air_f_mix_si</u> air_f_pressure_si air_f_soundspeed_si chk_iapws10_table</p>
--	--	--	--

## Level 3 routines

<p><b>(S11) Flu_3a</b></p> <p><u>Uses</u> constants_0, convert_0, maths_0, flu_1</p> <p><u>Public Routines</u> get_it_ctrl_density liq_density_si <u>liq_g_si</u> set_it_ctrl_density vap_density_si <u>vap_g_si</u></p>		<p><b>(S12) Sea_3a</b></p> <p><u>Uses</u> constants_0, sal_1, sal_2, flu_3a (convert_0, maths_0, flu_1)</p> <p><u>Public Routines</u> chk_iapws08_table8a chk_iapws08_table8b chk_iapws08_table8c sea_chempot_h2o_si sea_chempot_rel_si sea_cp_si sea_density_si sea_enthalpy_si sea_entropy_si <u>sea_g_si</u> sea_g_contraction_t_si sea_g_expansion_si sea_gibbs_energy_si sea_internal_energy_si sea_kappa_s_si sea_kappa_t_si sea_lapserate_si sea_osm_coeff_si sea_soundspeed_si sea_temp_maxdensity_si</p>	<p><b>(S13) Air_3a</b></p> <p><u>Uses</u> constants_0, convert_0, maths_0, air_1, air_2 (flu_1)</p> <p><u>Public Routines</u> air_density_si <u>air_g_si</u> get_it_ctrl_airdensity set_it_ctrl_airdensity</p>
--	--	---	--

<p><b>(S14) Flu_3b</b></p> <p><u>Uses</u>  constants_0, flu_2, flu_3a  (convert_0, maths_0, flu_1)</p> <p><u>Public Routines</u>  liq_cp_si  liq_cv_si  liq_enthalpy_si  liq_entropy_si  liq_expansion_si  liq_gibbs_energy_si  liq_internal_energy_si  liq_kappa_s_si  liq_kappa_t_si  liq_lapserate_si  liq_soundspeed_si  vap_cp_si  vap_cv_si  vap_enthalpy_si  vap_entropy_si  vap_expansion_si  vap_gibbs_energy_si  vap_internal_energy_si  vap_kappa_s_si  vap_kappa_t_si  vap_lapserate_si  vap_soundspeed_si</p>		<p><b>(S15) Sea_3b</b></p> <p><u>Uses</u>  constants_0, sal_2, flu_3a,  sea_3a (convert_0, maths_0,  flu_1, sal_1)</p> <p><u>Public Routines</u>  <u>sea_h_si</u>  sea_h_contraction_h_si  sea_h_contraction_t_si  sea_h_contraction_theta_si  sea_h_expansion_h_si  sea_h_expansion_t_si  sea_h_expansion_theta_si  sea_potdensity_si  sea_potenthalpy_si  sea_pottemp_si  sea_temperature_si  set_it_ctrl_pottemp</p>	<p><b>(S16) Air_3b</b></p> <p><u>Uses</u>  constants_0, convert_0,  air_1, air_2, air_3a  (maths_0, flu_1)</p> <p><u>Public Routines</u>  air_g_chempot_vap_si  air_g_compressibility  factor_si  air_g_contraction_si  air_g_cp_si  air_g_cv_si  air_g_density_si  air_g_enthalpy_si  air_g_entropy_si  air_g_expansion_si  air_g_gibbs_energy_si  air_g_internal_energy_si  air_g_kappa_s_si  air_g_kappa_t_si  air_g_lapserate_si  air_g_soundspeed_si  chk_lemmon_etal_2000</p>
		<p><b>(S17) Sea_3c</b></p> <p><u>Uses</u>  constants_0, sea_3a, sea_3b  (convert_0, maths_0, flu_1,  sal_1, sal_2, flu_3a)</p> <p><u>Public Routines</u>  sea_eta_contraction_h_si  sea_eta_contraction_t_si  sea_eta_contraction_theta_si  sea_eta_density_si  sea_eta_entropy_si  sea_eta_expansion_h_si  sea_eta_expansion_t_si  sea_eta_expansion_theta_si  sea_eta_potdensity_si  sea_eta_pottemp_si  sea_eta_temperature_si  set_it_ctrl_entropy_si</p>	<p><b>(S18) Air_3c</b></p> <p><u>Uses</u>  constants_0, convert_0,  air_2, air_3a, air_3b  (maths_0, air_1, flu_1)</p> <p><u>Public Routines</u>  air_h_si  air_potdensity_si  air_potenthalpy_si  air_pottemp_si  air_temperature_si  set_it_ctrl_air_pottemp</p>
		<p><b>(S19) Sea_3d</b></p> <p><u>Uses</u>  constants_0, sal_2, flu_3a  (convert_0, maths_0, flu_1,  sal_1)</p> <p><u>Public Routines</u>  sea_sa_si  set_it_ctrl_salinity</p>	

## Level 4 routines

<p><b>(S20) Liq_Vap_4</b></p> <p><u>Uses</u> constants_0, maths_0, flu_1, flu_2, flu_3a (Convert_0)</p> <p><u>Public Routines</u> chk_iapws95_table8 liq_vap_boilingtemperature_si liq_vap_chempot_si liq_vap_density_liq_si liq_vap_density_vap_si liq_vap_enthalpy_evap_si liq_vap_enthalpy_liq_si liq_vap_enthalpy_vap_si liq_vap_entropy_evap_si liq_vap_entropy_liq_si liq_vap_entropy_vap_si liq_vap_pressure_liq_si liq_vap_pressure_vap_si liq_vap_temperature_si liq_vap_vapourpressure_si liq_vap_volume_evap_si set_liq_vap_eq_at_p set_liq_vap_eq_at_t set_it_ctrl_liq_vap</p>	<p><b>(21) Ice_Vap_4</b></p> <p><u>Uses</u> constants_0, maths_0, flu_1, flu_2, ice_1, ice_2</p> <p><u>Public Routines</u> ice_vap_chempot_si ice_vap_density_ice_si ice_vap_density_vap_si ice_vap_enthalpy_ice_si ice_vap_enthalpy_subl_si ice_vap_enthalpy_vap_si ice_vap_entropy_ice_si ice_vap_entropy_subl_si ice_vap_entropy_vap_si ice_vap_pressure_vap_si ice_vap_sublimationpressure_si ice_vap_sublimationtemp_si ice_vap_temperature_si ice_vap_volume_subl_si set_ice_vap_eq_at_p set_ice_vap_eq_at_t set_it_ctrl_ice_vap</p>	<p><b>(S22) Sea_Vap_4</b></p> <p><u>Uses</u> constants_0, maths_0, flu_1, sal_1, sal_2, flu_3a, sea_3a, flu_3b (convert_0, flu_2)</p> <p><u>Public Routines</u> sea_vap_boilingtemperature_si sea_vap_brinefraction_seavap_si sea_vap_brinesalinity_si sea_vap_cp_seavap_si sea_vap_density_sea_si sea_vap_density_seavap_si sea_vap_density_vap_si sea_vap_enthalpy_evap_si sea_vap_enthalpy_sea_si sea_vap_enthalpy_seavap_si sea_vap_enthalpy_vap_si sea_vap_entropy_sea_si sea_vap_entropy_seavap_si sea_vap_entropy_vap_si sea_vap_expansion_seavap_si <u>sea_vap_g_si</u> sea_vap_kappa_t_seavap_si sea_vap_pressure_si sea_vap_salinity_si sea_vap_temperature_si sea_vap_vapourpressure_si sea_vap_volume_evap_si set_it_ctrl_sea_vap set_sea_vap_eq_at_s_p set_sea_vap_eq_at_s_t set_sea_vap_eq_at_t_p</p>	
	<p><b>(S23) Ice_Liq_4</b></p> <p><u>Uses</u> constants_0, maths_0, flu_1, ice_1, flu_2, ice_2</p> <p><u>Public Routines</u> ice_liq_chempot_si ice_liq_density_ice_si ice_liq_density_liq_si ice_liq_enthalpy_ice_si ice_liq_enthalpy_liq_si ice_liq_enthalpy_melt_si ice_liq_entropy_ice_si ice_liq_entropy_liq_si ice_liq_entropy_melt_si ice_liq_meltingpressure_si ice_liq_meltingtemperature_si ice_liq_pressure_liq_si ice_liq_temperature_si ice_liq_volume_melt_si set_ice_liq_eq_at_p set_ice_liq_eq_at_t set_it_ctrl_ice_liq</p>	<p><b>(S24) Sea_Liq_4</b></p> <p><u>Uses</u> constants_0, flu_1, sal_1, flu_2, sal_2, flu_3a (convert_0, maths_0)</p> <p><u>Public Routines</u> sea_liq_osmoticpressure_si set_sea_liq_eq_at_s_t_p set_it_ctrl_sea_liq</p>	

		<p><b>(S25) Sea_Ice_4</b></p> <p><u>Uses</u>  constants_0, convert_0, maths_0,  flu_1, ice_1, sal_1, ice_2, sal_2,  flu_3a, sea_3a, flu_3b (flu_2)</p> <p><u>Public Routines</u>  sea_ice_brinefraction_seaice_si  sea_ice_brinesalinity_si  sea_ice_cp_seaice_si  sea_ice_density_ice_si  sea_ice_density_sea_si  sea_ice_density_seaice_si  sea_ice_dtfdp_si  sea_ice_dtfds_si  sea_ice_enthalpy_ice_si  sea_ice_enthalpy_melt_si  sea_ice_enthalpy_sea_si  sea_ice_enthalpy_seaice_si  sea_ice_entropy_ice_si  sea_ice_entropy_sea_si  sea_ice_entropy_seaice_si  sea_ice_expansion_seaice_si  sea_ice_freezingtemperature_si  sea_ice_g_si  sea_ice_kappa_t_seaice_si  sea_ice_meltingpressure_si  sea_ice_pressure_si  sea_ice_salinity_si  sea_ice_temperature_si  sea_ice_volume_melt_si  set_it_ctrl_sea_ice  set_sea_ice_eq_at_s_p  set_sea_ice_eq_at_s_t  set_sea_ice_eq_at_t_p</p>	
		<p><b>(S26) Sea_Air_4</b></p> <p><u>Uses</u>  constants_0, convert_0, maths_0,  flu_1, sal_1, air_1, flu_2, sal_2,  air_2, flu_3a, sea_3a, air_3a,  air_3b, liq_vap_4, liq_air_4a</p> <p><u>Public Routines</u>  sea_air_chempot_evap_si  sea_air_condense_temp_si  sea_air_density_air_si  sea_air_density_vap_si  sea_air_enthalpy_evap_si  sea_air_entropy_air_si  sea_air_massfraction_air_si  sea_air_vapourpressure_si  set_it_ctrl_sea_air  set_sea_air_eq_at_s_a_p  set_sea_air_eq_at_s_t_p</p>	

<p><b>(S27) Liq_Ice_Air_4</b></p> <p><u>Uses</u>  constants_0, convert_0,  maths_0, flu_1, ice_1, air_1,  flu_2, ice_2, air_2, air_3b,  ice_liq_4 (air_3a)</p> <p><u>Public Routines</u>  liq_ice_air_airfraction_si  liq_ice_air_density_si  liq_ice_air_dryairfraction_si  liq_ice_air_enthalpy_si  liq_ice_air_entropy_si  liq_ice_air_ifl_si  liq_ice_air_uml_si  liq_ice_air_liquidfraction_si  liq_ice_air_pressure_si  liq_ice_air_solidfraction_si  liq_ice_air_temperature_si  liq_ice_air_vapourfraction_si  set_liq_ice_air_eq_at_a  set_liq_ice_air_eq_at_p  set_liq_ice_air_eq_at_t  set_liq_ice_air_eq_at  _wa_eta_wt  set_liq_ice_air_eq_at  _wa_wl_wi  set_it_ctrl_liq_ice_air</p>		<p><b>(S28) Sea_Ice_Vap_4</b></p> <p><u>Uses</u>  constants_0, maths_0, flu_1,  ice_1, sal_1, sal_2</p> <p><u>Public Routines</u>  sea_ice_vap_density_vap_si  sea_ice_vap_pressure_si  sea_ice_vap_salinity_si  sea_ice_vap_temperature_si  set_it_ctrl_sea_ice_vap  set_sea_ice_vap_eq_at_p  set_sea_ice_vap_eq_at_s  set_sea_ice_vap_eq_at_t</p>	
<p><b>(S29) Liq_Air_4a</b></p> <p><u>Uses</u>  constants_0, convert_0,  maths_0, flu_1, air_1, flu_2,  air_2, flu_3a, air_3a, air_3b,  liq_vap_4</p> <p><u>Public Routines</u>  liq_air_a_from_rh_cct_si  liq_air_a_from_rh_wmo_si  liq_air_condensationpressure_  si  liq_air_density_air_si  liq_air_density_liq_si  liq_air_density_vap_si  liq_air_dewpoint_si  liq_air_enthalpy_evap_si  liq_air_entropy_air_si  liq_air_ict_si  liq_air_ict_si  liq_air_massfraction_air_si  liq_air_pressure_si  liq_air_rh_cct_from_a_si  liq_air_rh_wmo_from_a_si  liq_air_temperature_si  set_it_ctrl_liq_air  set_liq_air_eq_at_a_eta  set_liq_air_eq_at_a_p  set_liq_air_eq_at_a_t  set_liq_air_eq_at_t_p</p>	<p><b>(S30) Ice_Air_4a</b></p> <p><u>Uses</u>  constants_0, convert_0, maths_0,  air_1, ice_1, ice_2, air_2, air_3a,  air_3b, ice_vap_4 (flu_1, flu_2)</p> <p><u>Public Routines</u>  ice_air_a_from_rh_cct_si  ice_air_a_from_rh_wmo_si  ice_air_condensationpressure_si  ice_air_density_air_si  ice_air_density_ice_si  ice_air_density_vap_si  ice_air_enthalpy_subl_si  ice_air_frostpoint_si  ice_air_ict_si  ice_air_ict_si  ice_air_massfraction_air_si  ice_air_pressure_si  ice_air_rh_cct_from_a_si  ice_air_rh_wmo_from_a_si  ice_air_sublimationpressure_si  ice_air_temperature_si  set_ice_air_eq_at_a_eta  set_ice_air_eq_at_a_p  set_ice_air_eq_at_a_t  set_ice_air_eq_at_t_p  set_it_ctrl_ice_air</p>		

<p><b>(S31) Liq_Air_4b</b></p> <p><u>Uses</u>  constants_0, flu_3a, air_3a,  liq_air_4a (convert_0, maths_0,  flu_1, air_1, flu_2, air_2, air_3b,  liq_vap_4)</p> <p><u>Public Routines</u>  liq_air_g_si  liq_air_g_cp_si  liq_air_g_density_si  liq_air_g_enthalpy_si  liq_air_g_entropy_si  liq_air_g_expansion_si  liq_air_g_kappa_t_si  liq_air_g_lapserate_si  liq_air_liquidfraction_si  liq_air_vapourfraction_si</p>	<p><b>(S32) Ice_Air_4b</b></p> <p><u>Uses</u>  constants_0, convert_0, ice_1,  air_3a, ice_air_4a (maths_0,  flu_1, air_1, flu_2, ice_2, air_2,  air_3b, ice_vap_4)</p> <p><u>Public Routines</u>  ice_air_g_si  ice_air_g_cp_si  ice_air_g_density_si  ice_air_g_enthalpy_si  ice_air_g_entropy_si  ice_air_g_expansion_si  ice_air_g_kappa_t_si  ice_air_g_lapserate_si  ice_air_solidfraction_si  ice_air_vapourfraction_si</p>		
<p><b>(S33) Liq_Air_4c</b></p> <p><u>Uses</u>  constants_0, air_3a, ice_liq_4,  liq_air_4a, liq_air_4b  (convert_0, maths_0, flu_1,  ice_1, air_1, flu_2, ice_2 air_2,  flu_3a, air_3b, liq_vap_4)</p> <p><u>Public Routines</u>  liq_air_h_si  liq_air_h_cp_si  liq_air_h_density_si  liq_air_h_kappa_s_si  liq_air_h_lapserate_si  liq_air_h_temperature_si  liq_air_potdensity_si  liq_air_potenthalpy_si  liq_air_pottemp_si  set_it_ctrl_liq_air_pottemp</p>	<p><b>(S34) Ice_Air_4c</b></p> <p><u>Uses</u>  constants_0, convert_0, ice_liq_4,  ice_air_4b (maths_0, flu_1, ice_1,  air_1, flu_2, ice_2, air_2, air_3a,  air_3b, ice_air_4a, ice_vap_4)</p> <p><u>Public Routines</u>  ice_air_h_si  ice_air_h_cp_si  ice_air_h_density_si  ice_air_h_kappa_s_si  ice_air_h_lapserate_si  ice_air_h_temperature_si  ice_air_potdensity_si  ice_air_potenthalpy_si  ice_air_pottemp_si  set_it_ctrl_ice_air_pottemp</p>		



## Level 5 routines

<p><b>(S35) Flu_IF97_5</b></p> <p><u>Uses</u> constants_0</p> <p><u>Public Routines</u> chk_iapws97_table fit_liq_density_if97_si fit_liq_g_if97_si fit_vap_density_if97_si fit_vap_g_if97_si</p>	<p><b>(S36) Ice_Flu_5</b></p> <p><u>Uses</u> constants_0</p> <p><u>Public Routines</u> fit_ice_liq_pressure_si fit_ice_liq_temperature_si fit_ice_vap_pressure_si</p>	<p><b>(S37) Sea_5a</b></p> <p><u>Uses</u> constants_0, sea_3a, sea_3b, sea_3c (convert_0, maths_0, flu_1, sal_1, sal_2, flu_3a)</p> <p><u>Public Routines</u> sea_alpha_ct_si sea_alpha_pt0_si sea_alpha_t_si sea_beta_ct_si sea_beta_pt0_si sea_beta_t_si sea_cabb_ct_si sea_cabb_pt0_si sea_ctmp_from_ptmp0_si sea_ptmp0_from_ctmp_si sea_thrmb_ct_si sea_thrmb_pt0_si</p>	<p><b>(S38) Air_5</b></p> <p><u>Uses</u> constants_0, air_3b, liq_air_4a (convert_0, maths_0, flu_1, flu_2, flu_3a, air_1, air_2, air_3a, liq_vap_4)</p> <p><u>Public Routines</u> air_lapserate_moist_c 100m</p>
<p><b>(S39) Liq_F03_5</b></p> <p><u>Uses</u> constants_0</p> <p><u>Public Routines</u> chk_iapws09_table6 fit_liq_cp_f03_si fit_liq_density_f03_si fit_liq_expansion_f03_si fit_liq_g_f03_si fit_liq_kappa_t_f03_si fit_liq_soundspeed_f03_si</p>	<p><b>(S40) OS2008_5</b></p> <p><u>Uses</u> flu_1, flu_2, flu_3a, ice_1, liq_vap_4, sal_1, sal_2 (constants_0, convert_0, maths_0)</p> <p><u>Public Routines</u> chk_os2008_table</p>	<p><b>(S41) GSW_Library_5</b></p> <p><u>Uses</u> constants_0, maths_0, liq_f03_5, flu_1, flu_3a, sal_1, sal_2, sea_3a, sea_3b, sea_5a (convert_0)</p> <p><u>Public Routines</u> gsw_alpha_ct gsw_alpha_pt0 gsw_alpha_t gsw_asal_from_psal gsw_beta_ct gsw_beta_pt0 gsw_beta_t gsw_cabb_ct gsw_cabb_pt0 gsw_cp gsw_ctmp_from_ptmp0 gsw_dens gsw_enthalpy gsw_entropy gsw_g gsw_kappa gsw_kappa_t gsw_pden gsw_psal_from_asal gsw_ptmp gsw_ptmp0_from_ctmp gsw_specvol gsw_svel gsw_thrmb_ct gsw_thrmb_pt0</p>	<p><b>(S42) Convert_5</b></p> <p><u>Uses</u> constants_0, convert_0</p> <p><u>Public Routines</u> cnv_pressure cnv_salinity cnv_temperature</p>

## Level 0: The Conversion Routines

Table S2 (Convert\_0): This module implements formulae to permit easy conversion between various representations of the relative contributions of dry air and water vapour to humid air, and for conversion between various commonly used units for pressure, salinity and temperature. (See Part I, section 2.4).

### (S2) Convert\_0 Module

Function call	Mathematical formulae relating various units	Comments
air_massfraction_air_si: Converts between the mole fraction of dry air in moist air and the mass fraction of dry air in moist air. (S2.1)		
air_massfraction_air_si(x) $x = \zeta / (\text{mol mol}^{-1})$	$A = \frac{x_A}{1 - (1 - x_A)(1 - M_w / M_A)}$	$\zeta$ is the mole fraction, the number of moles of dry air per moles of humid air
air_massfraction_vap_si: Converts between the mole fraction of dry air in moist air and the mass fraction of water vapour in moist air. (S2.2)		
air_massfraction_vap_si(x) $x = \zeta / (\text{mol mol}^{-1})$	$1 - A = \frac{1 - x_A}{1 - x_A(1 - M_A / M_w)}$	$\zeta$ is the mole fraction, the number of moles of dry air per moles of humid air
air_molar_mass_si: Determines the molar mass of moist air given the mass fraction of dry air in humid air (S2.3)		
air_molar_mass_si(a) $a = A / (\text{kg kg}^{-1})$	$M_{AV} = \frac{1}{(1 - A)/M_w + A/M_A}$	A is the mass fraction of dry air in humid air
air_molfraction_air_si: Converts between the mass fraction of dry air in moist air and the mole fraction of dry air in moist air. (S2.4)		
air_molfraction_air_si(a) $a = A / (\text{kg kg}^{-1})$	$x_A = \frac{A(M_w / M_A)}{1 - A(1 - M_w / M_A)}$	A is the mass fraction of dry air in humid air
air_molfraction_vap_si: Converts between the mass fraction of dry air in moist air and the mole fraction of water vapour in moist air. (S2.5)		
air_molfraction_vap_si(a) $a = A / (\text{kg kg}^{-1})$	$1 - x_A = \frac{1 - A}{1 - A(1 - M_w / M_A)}$	A is the mass fraction of dry air in humid air
asal_from_psal: Converts from Practical Salinity to Absolute Salinity, possibly allowing for composition anomalies using a lookup table. (S2.6)		
asal_from_psal (sp, long, lat, p) sp = $S_P$ (Practical Salinity) long = longitude in degrees lat = latitude in degrees p = $P / \text{Pa}$	$S_A = [S_P \times (35.16504/35) + \delta S_A(\text{long, lat, p})] \text{ kg kg}^{-1}$ from McDougall et al (2009).	Note that $P$ , long and lat are required to estimate the effect of composition anomalies on Absolute Salinity. If absent the effect is neglected.

Table S2 (Convert\_0), continued

<b>(S2) Convert_0 Module (cont'd)</b>			
Function call	Mathematical formula	Units	Comments
psal_from_asal: Converts from Absolute Salinity to Practical Salinity, possibly allowing for composition anomalies using a lookup table. (S2.7)			
psal_from_asal (s, long, lat, p) s = $S_P$ (Practical Salinity) long = longitude in degrees lat = latitude in degrees p = $P / \text{Pa}$	$S_P = (35/35.16504) \times (S_A - \delta S_A(\text{long, lat, p}))$ from McDougall et al (2009).	1	Note that $P$ , long and lat are required to estimate the effect of composition anomalies on Absolute Salinity. If absent the effect is neglected.

### Level 1: The Primary Standard

Table S3 (Flu\_1): This module implements the Helmholtz potential of fluid water and its first and second partial derivatives with respect to temperature and density as defined in IAPWS-95. (See Part I, section 2.1.)

<b>(S3) Flu_1 Module</b>			
Function call	Mathematical formula	Units	Comments
chk_iapws95_table6 and chk_iapws95_table7: Check routines for the thermodynamic properties of fluid water (S3.1)			
chk_iapws95_table6 and chk_iapws95_table7 are simple check routines for the thermodynamic properties of fluid pure water. They print out table values determined by the local implementation together with results published in tables 6 and 7 of IAPWS-95. Table 6 compares results of the ideal gas and residual parts of the dimensionless Helmholtz free energy and its derivatives. Table 7 outputs results for pressure, specific heat capacity at constant volume, sound speed and specific entropy for specified values of temperature and density. The published results are shown to the number of digits expected to be reproduced by double precision code.			
flu_f_si: Helmholtz potential of liquid water and vapour (S3.2)			
flu_f_si(m, n, t, d) t = $T / \text{K}$ d = $\rho / (\text{kg m}^{-3})$	$\frac{\partial^{m+n}}{\partial T^m \partial \rho^n} f^F(T, \rho)$	$\frac{\text{J m}^{3n}}{\text{K}^m \text{kg}^{n+1}}$	m, n $\geq 0$ m + n $\leq 2$

Table S4 (Ice\_1): This module implements the Gibbs potential of hexagonal ice I and its first and second partial derivatives with respect to temperature and pressure as defined in IAPWS-06. (See Part I, section 2.2.)

<b>(S4) Ice_1 Module</b>			
Function call	Mathematical formula	Units	Comments
<b>chk_iapws06_table6: Check routine for the thermodynamic properties of pure ice</b> (S4.1)			
chk_iapws06_table6 is a simple check routine for the thermodynamic properties of pure water ice Ih. It prints out values corresponding to table 6 of IAPWS-06 determined by the local implementation together with the corresponding published results. The routine produces values corresponding to the experimental triple point, the normal pressure melting point and $T = 100$ K and $P = 100$ MPa. The published results are shown to the number of digits expected to be reproduced by double precision code.			
<b>ice_g_si: Gibbs potential of ice Ih</b> (S4.2)			
ice_g_si(m, n, t, p) t = T / K p = P / Pa	$\frac{\partial^{m+n}}{\partial T^m \partial P^n} g^{\text{Ih}}(T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	m, n ≥ 0 m + n ≤ 2

Table S5 (Sal\_1): Expansion coefficients  $g_i(T, P)$  and their partial derivatives for the powers of salinity in the Gibbs function  $g^{\text{SW}}$  of seawater, depending on absolute temperature in K and absolute pressure in Pa. (See Part I, section 2.3.)

<b>(S5) Sal_1 Module</b>			
Function call	Mathematical formula	Units	Comments
<b>sal_g_term_si: coefficients of the salinity expansion of the Gibbs function of seawater</b> (S5.1)			
sal_g_term_si(m, n, t, p) t = T / K p = P / Pa	$\frac{\partial^{m+n}}{\partial T^m \partial P^n} g_i(T, P)$ $g^{\text{S}} = g_1(T) S_{\text{A}} \ln S_{\text{A}} + \sum_{i=2}^7 g_i(T, P) S_{\text{A}}^{i/2}$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	1 ≤ i ≤ 7 m, n ≥ 0 m + n ≤ 2

Table S6 (Air\_1): This module implements the Helmholtz potential of dry air and its first and second partial derivatives with respect to temperature and density as defined in Lemmon et al. (2000), the second cross-virial coefficient of Harvey and Huang (2007) and the third cross-virial coefficients of Hyland and Wexler (1983). (See Part I, section 2.4.)

<b>(S6) Air_1 Module</b>			
Function call	Mathematical formula	Units	Comments
<b>air_baw_m3mol: 2nd virial coefficient air-water</b> (S6.1)			
air_baw_m3mol(n, t) t = T / K	$\frac{d^n}{dT^n} B_{aw}(T)$	$\frac{\text{m}^3}{\text{mol K}^n}$	$0 \leq n \leq 2$
<b>air_caaw_m6mol2: 3rd virial coefficient air-air-water</b> (S6.2)			
air_caaw_m6mol2(n, t) t = T / K	$\frac{d^n}{dT^n} C_{aaw}(T)$	$\frac{\text{m}^6}{\text{mol}^2 \text{K}^n}$	$0 \leq n \leq 2$
<b>air_caww_m6mol2: 3rd virial coefficient air-water-water</b> (S6.3)			
air_caww_m6mol2(n, t) t = T / K	$\frac{d^n}{dT^n} C_{aww}(T)$	$\frac{\text{m}^6}{\text{mol}^2 \text{K}^n}$	$0 \leq n \leq 2$
<b>dry_f_si: Helmholtz potential of dry air</b> (S6.4)			
dry_f_si(m, n, t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$\frac{\partial^{m+n}}{\partial T^m \partial \rho^n} f^A(T, \rho)$	$\frac{\text{J m}^{3n}}{\text{K}^m \text{kg}^{n+1}}$	$m, n \geq 0$ $m + n \leq 2$
<b>dry_init_clear: Clears the coefficients used in the definition of the dry-air Helmholtz function.</b> (S6.5)			
This routine is used to switch from the original Lemmon et al. formulation to the formulation in which enthalpy and entropy have been adjusted to zero at $T = 273.15 \text{ K}$ , $P = 101325 \text{ Pa}$ as used in the SIA library. After execution, the first call of <b>dry_f_si</b> triggers the default initialization.			
<b>dry_init_lemmon2000: Initialization routine for the original Lemmon et al. (2000) formulation for dry air.</b> (S6.6)			
Initializes the coefficients of the dry-air Helmholtz function consistent with the original formulation of Lemmon et al. (2000). This is useful for making direct comparison with the check values tabulated in the original publication. Note however that the precision of the tabulated results is limited so that a comparison with the results tabulated in IAPWS-10 is preferred. Execution overwrites any previous initialization.			

## Level 2: Directly Derived Properties

Table S7 (Flu\_2): Thermodynamic properties of pure liquid water and water vapour, depending on absolute temperature in K and density in  $\text{kg m}^{-3}$ . All quantities here are computed directly from the Helmholtz potential function  $f^F$ . (See Part I, section 3.1.)

<b>(S7) Flu_2 Module</b>			
Function call	Mathematical formula	Unit	Comments
<b>flu_cp_si: specific isobaric heat capacity</b> (S7.1)			
flu_cp_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$c_p = T \left[ \frac{(f_{T\rho}^F)^2 \rho}{2f_\rho^F + \rho f_{\rho\rho}^F} - f_{TT}^F \right]$	$\frac{\text{J}}{\text{kg K}}$	$f^F$ is the Helmholtz potential for fluid water. Subscripts indicate partial differentiation.
<b>flu_cv_si: specific isochoric heat capacity</b> (S7.2)			
flu_cv_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$c_v = -T f_{TT}^F$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S7.1)
<b>flu_enthalpy_si: specific enthalpy</b> (S7.3)			
flu_enthalpy_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$h = f^F - T f_T^F + \rho f_\rho^F$	$\frac{\text{J}}{\text{kg}}$	See comments for (S7.1)
<b>flu_entropy_si: specific entropy</b> (S7.4)			
flu_entropy_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$\eta = -f_T^F$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S7.1)
<b>flu_expansion_si: thermal expansion coefficient</b> (S7.5)			
flu_expansion_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$\alpha = \frac{f_{T\rho}^F}{2f_\rho^F + \rho f_{\rho\rho}^F}$	$\frac{1}{\text{K}}$	See comments for (S7.1)
<b>flu_gibbs_energy_si: specific Gibbs energy</b> (S7.6)			
flu_gibbs_energy_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$g = f^F + \rho f_\rho^F$	$\frac{\text{J}}{\text{kg}}$	See comments for (S7.1)
<b>flu_internal_energy_si: specific internal energy</b> (S7.7)			
flu_internal_energy_si(t, d) t = T / K d = $\rho / (\text{kg m}^{-3})$	$u = f^F - T f_T^F$	$\frac{\text{J}}{\text{kg}}$	See comments for (S7.1)

Table S7 (Flu\_2), continued

<b>(S7) Flu_2 Module (cont'd)</b>			
Function call	Mathematical formula	Unit	Comments
<b>flu_kappa_s_si: isentropic compressibility (S7.8)</b>			
flu_kappa_s_si(t, d) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\kappa_s = \frac{f_{TT}^F / \rho^2}{f_{TT}^F (2f_\rho^F + \rho f_{\rho\rho}^F) - \rho (f_{T\rho}^F)^2}$	$\frac{1}{\text{Pa}}$	See comments for (S7.1)
<b>flu_kappa_t_si: isothermal compressibility (S7.9)</b>			
flu_kappa_t_si(t, d) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\kappa_T = \frac{1}{\rho^2 (2f_\rho^F + \rho f_{\rho\rho}^F)}$	$\frac{1}{\text{Pa}}$	See comments for (S7.1)
<b>flu_lapserate_si: adiabatic lapse rate (S7.10)</b>			
flu_lapserate_si(t, d) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\Gamma = \frac{f_{T\rho}^F / \rho}{\rho (f_{T\rho}^F)^2 - f_{TT}^F (2f_\rho^F + \rho f_{\rho\rho}^F)}$	$\frac{\text{K}}{\text{Pa}}$	See comments for (S7.1) Note that this is per pascal.
<b>flu_pressure_si: absolute pressure (S7.11)</b>			
flu_pressure_si(t, d) t = T / K d = ρ / (kg m <sup>-3</sup> )	$P = \rho^2 f_\rho^F$	Pa	See comments for (S7.1)
<b>flu_soundspeed_si: sound speed (S7.12)</b>			
flu_soundspeed_si(t, d) t = T / K d = ρ / (kg m <sup>-3</sup> )	$c = \sqrt{\rho^2 \frac{f_{TT}^F f_{\rho\rho}^F - (f_{T\rho}^F)^2}{f_{TT}^F} + 2\rho f_\rho^F}$	$\frac{\text{m}}{\text{s}}$	See comments for (S7.1)

Table S8 (Ice\_2): Thermodynamic properties of hexagonal ice I, depending on absolute temperature in K and absolute pressure in Pa. All quantities here are determined directly from  $g^{\text{Ih}}$ , the Gibbs potential function for Ice Ih. (See Part I, section 3.2.)

<b>(S8) Ice_2 Module</b>			
Function call	Mathematical formula	Unit	Comments
<b>ice_chempot_si: chemical potential (S8.1)</b>			
ice_chempot_si(t, p) t = T / K p = P / Pa	$\mu = g^{\text{Ih}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{Ih}}$ is the Gibbs potential function for hexagonal ice I, defined at level 1. Ice Ih is the form that occurs under most common conditions.
<b>ice_cp_si: specific isobaric heat capacity (S8.2)</b>			
ice_cp_si(t, p) t = T / K p = P / Pa	$c_p = -T g_{TT}^{\text{Ih}}$	$\frac{\text{J}}{\text{kg K}}$	Subscripts indicate partial differentiation.
<b>ice_density_si: density of sea ice (S8.3)</b>			
ice_density_si(t, p) t = T / K p = P / Pa	$\rho = \frac{1}{g_P^{\text{Ih}}}$	$\frac{\text{kg}}{\text{m}^3}$	See comments for (S8.1) and (S8.2)
<b>ice_enthalpy_si: specific enthalpy (S8.4)</b>			
ice_enthalpy_si(t, p) t = T / K p = P / Pa	$h = g^{\text{Ih}} - T g_T^{\text{Ih}}$	$\frac{\text{J}}{\text{kg}}$	See comments for (S8.1) and (S8.2)
<b>ice_entropy_si: specific entropy (S8.5)</b>			
ice_entropy_si(t, p) t = T / K p = P / Pa	$\eta = -g_T^{\text{Ih}}$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S8.1) and (S8.2)
<b>ice_expansion_si: thermal expansion coefficient (S8.6)</b>			
ice_expansion_si(t, p) t = T / K p = P / Pa	$\alpha = \frac{g_{TP}^{\text{Ih}}}{g_P^{\text{Ih}}}$	$\frac{1}{\text{K}}$	See comments for (S8.1) and (S8.2)
<b>ice_helmholtz_energy_si: specific Helmholtz energy (S8.7)</b>			
ice_helmholtz_energy_si(t, p) t = T / K p = P / Pa	$f = g^{\text{Ih}} - P g_P^{\text{Ih}}$	$\frac{\text{J}}{\text{kg}}$	See comments for (S8.1) and (S8.2)



Table S8 (Ice\_2), continued

<b>(S8) Ice_2 Module, cont'd</b>			
Function call	Mathematical formula	Unit	Comments
<b>ice_internal_energy_si: specific internal energy (S8.8)</b>			
ice_internal_energy_si(t, p) t = T / K p = P / Pa	$u = g^{\text{lh}} - T g_T^{\text{lh}} - P g_P^{\text{lh}}$	$\frac{\text{J}}{\text{kg}}$	See comments for (S8.1) and (S8.2)
<b>ice_kappa_s_si: isentropic compressibility (S8.9)</b>			
ice_kappa_s_si(t, p) t = T / K p = P / Pa	$\kappa_S = \frac{(g_{TP}^{\text{lh}})^2 - g_{TT}^{\text{lh}} g_{PP}^{\text{lh}}}{g_P^{\text{lh}} g_{TT}^{\text{lh}}}$	$\frac{1}{\text{Pa}}$	See comments for (S8.1) and (S8.2)
<b>ice_kappa_t_si: isothermal compressibility (S8.10)</b>			
ice_kappa_t_si(t, p) t = T / K p = P / Pa	$\kappa_T = -\frac{g_{PP}^{\text{lh}}}{g_P^{\text{lh}}}$	$\frac{1}{\text{Pa}}$	See comments for (S8.1) and (S8.2)
<b>ice_lapserate_si: adiabatic lapse rate (S8.11)</b>			
ice_lapserate_si(t, p) t = T / K p = P / Pa	$\Gamma = -\frac{g_{TP}^{\text{lh}}}{g_{TT}^{\text{lh}}}$	$\frac{\text{K}}{\text{Pa}}$	See comments for (S8.1) and (S8.2)
<b>ice_p_coefficient_si pressure coefficient (S8.12)</b>			
ice_p_coefficient_si(t, p) t = T / K p = P / Pa	$\beta_P = -\frac{g_{PP}^{\text{lh}}}{g_P^{\text{lh}}}$	$\frac{\text{Pa}}{\text{K}}$	See comments for (S8.1) and (S8.2)
<b>ice_specific_volume_si: specific volume (S8.13)</b>			
ice_specific_volume_si(t, p) t = T / K p = P / Pa	$v = g_P^{\text{lh}}$	$\frac{\text{m}^3}{\text{kg}}$	See comments for (S8.1) and (S8.2)

Table S9 (Sal\_2): Thermodynamic saline properties of seawater, depending on absolute salinity in kg kg<sup>-1</sup>, absolute temperature in K and absolute pressure in Pa. All quantities here are computed directly from  $g^S$ , the Gibbs potential function for the saline component of seawater. Subscripts on the Gibbs function  $g^S$  indicate partial differentiation. (See Part I, section 3.3.)

<b>(S9) Sal_2 Module</b>			
Function call	Mathematical formula	Unit	Comments
<b>sal_act_coeff_si: activity coefficient of salt</b> (S9.1)			
sal_act_coeff_si(s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\ln \gamma = \psi + S_A (1 - S_A) \frac{\partial \psi}{\partial S_A}$	1	$\gamma$ is the activity coefficient $\psi$ is the activity potential from (S9.2)
<b>sal_act_potential_si: activity potential</b> (S9.2)			
sal_act_potential_si (s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\psi = \frac{g^S(S_A, T, P) - g_2(T, P) S_A}{g_1(T) S_A - \ln S_A + \ln(1 - S_A)}$	1	$g_1$ and $g_2$ are coefficients in the expression for $g^S$ (e5.2, 3). See comments for (S9.8).
<b>sal_activity_w_si: activity of water in seawater</b> (S9.3)			
sal_activity_w_si(s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$a_w = \exp\left\{\frac{g^S - S_A g_S^S}{R_W T}\right\}$ $R_W = R / M_W$ $= 461.51805 \text{ J kg}^{-1} \text{ K}^{-1}$	1	$R_W$ is the specific gas constant of pure water. See comments for (S9.8).
<b>sal_chem_coeff_si: chemical coefficient for seawater</b> (S9.4)			
sal_chem_coeff_si(s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$S_A^2 g_{SS}^S$	$\frac{\text{J}}{\text{kg}}$	See comments for (S9.8).
<b>sal_chempot_h2o_si: saline part of the chemical potential of water in seawater</b> (S9.5)			
sal_chempot_h2o_si (s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\mu^w = g^S - S_A g_S^S$	$\frac{\text{J}}{\text{kg}}$	See comments for (S9.8).
<b>sal_chempot_rel_si: relative chemical potential</b> (S9.6)			
sal_chempot_rel_si (s, t, p) s = S <sub>A</sub> = S <sub>A</sub> / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\mu^{\text{rel}} = g_S^S$	$\frac{\text{J}}{\text{kg}}$	See comments for (S9.8).

Table S9 (Sal\_2), continued

<b>(S9) Sal_2 Module, cont'd</b>			
Function call	Mathematical formula	Unit	Comments
<b>sal_dilution_si: dilution coefficient</b> (S9.7)			
sal_dilution_si(s, t, p) $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$D = S_A g_{SS}^S$	$\frac{\text{J}}{\text{kg}}$	See comments for (S9.8).
<b>sal_g_si: saline part of the Gibbs function</b> (S9.8)			
sal_g_si(l, m, n, s, t, p) $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\frac{\partial^{l+m+n}}{\partial S_A^l \partial T^m \partial P^n} g^S(S_A, T, P)$ $g^S = g_1(T) S_A \ln S_A$ $+ \sum_{i=2}^7 g_i(T, P) S_A^{i/2}$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{kg K}^m}$	$l, m, n \geq 0$ $l + m + n \leq 2$ $g^S$ is the saline part of the Gibbs function of seawater. Its coefficients $g_i, i = 1 \dots 7$ , are defined at level 1.
<b>sal_mixenthalpy_si: specific mixing enthalpy</b> (S9.9)			
sal_mixenthalpy_si (s1, s2, w1, t, p) $s1 = S_{S2} = S_{S2} / (\text{kg kg}^{-1})$ $s2 = S_{A2} = S_{A2} / (\text{kg kg}^{-1})$ $w1 = M_1 / (M_1 + M_2)$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\Delta h^{\text{mix}} = h^S(w_1 S_{A1} + w_2 S_{A2}, T, P)$ $- w_1 h^S(S_{A1}, T, P)$ $- w_2 h^S(S_{A2}, T, P)$ $h^S = g^S - T g_T^S$ $w_2 = 1 - w_1$	$\frac{\text{J}}{\text{kg}}$	$h^S$ is the enthalpy associated with the saline component of seawater. $M_i$ is the mass fraction of solution $i$ . See comments for (S9.8).
<b>sal_mixentropy_si: specific mixing entropy</b> (S9.10)			
sal_mixentropy_si (s1, s2, w1, t, p) $s1 = S_{S2} = S_{S2} / (\text{kg kg}^{-1})$ $s2 = S_{A2} = S_{A2} / (\text{kg kg}^{-1})$ $w1 = M_1 / (M_1 + M_2)$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\Delta \eta^{\text{mix}} = w_1 g_T^S(S_{A1}, T, P)$ $+ w_2 g_T^S(S_{A2}, T, P)$ $- g_T^S(w_1 S_{A1} + w_2 S_{A2}, T, P)$ $w_2 = 1 - w_1$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S9.8).

Table S9 (Sal\_2), continued

<b>(S9) Sal_2 Module, cont'd</b>			
Function call	Mathematical formula	Unit	Comments
<b>sal_mixvolume_si: specific mixing volume (S9.11)</b>			
<b>sal_mixvolume_si</b> (s1, s2, w1, t, p) $s1 = S_{S2} = S_{S2} / (\text{kg kg}^{-1})$ $s2 = S_{A2} = S_{A2} / (\text{kg kg}^{-1})$ $w1 = M_1 / (M_1 + M_2)$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\Delta v^{\text{mix}} = g_P^S(w_1 S_{A1} + w_2 S_{A2}, T, P)$ $- w_1 g_P^S(S_{A1}, T, P)$ $- w_2 g_P^S(S_{A2}, T, P)$ $w_2 = 1 - w_1$	$\frac{\text{m}^3}{\text{kg}}$	See comments for (S9.8).
<b>sal_molality_si: molality of seawater (S9.12)</b>			
<b>sal_molality_si(s, t, p)</b> $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$m = \frac{S_A}{(1 - S_A) M_S}$ $M_S = 31.4038218 \text{ g mol}^{-1}$	$\frac{\text{mol}}{\text{kg}}$	$M_S$ is the mean molar mass of sea salt with Reference Composition
<b>sal_osm_coeff_si: osmotic coefficient (S9.13)</b>			
<b>sal_osm_coeff_si(s, t, p)</b> $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\phi = 1 + S_A (1 - S_A) \frac{\partial \psi}{\partial S_A}$	1	$\psi$ is the activity potential from (S9.2)
<b>sal_saltenthalpy_si: specific enthalpy of seasalt (S9.14)</b>			
<b>sal_saltenthalpy_si(s, t, p)</b> $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$h_s = \frac{g^S - T g_T^S}{S_A}$	$\frac{\text{J}}{\text{kg}}$	The subscript on $h_s$ is not a derivative. See comments for (S9.8)
<b>sal_salentropy_si: specific entropy of seasalt (S9.15)</b>			
<b>sal_salentropy_si(s, t, p)</b> $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\eta_s = -\frac{g_T^S}{S_A}$	$\frac{\text{J}}{\text{kg K}}$	The subscript of $\eta_s$ is not a derivative. See comments for (S9.8)
<b>sal_saltvolume_si: specific volume of seasalt (S9.16)</b>			
<b>sal_saltvolume_si(s, t, p)</b> $s = S_A = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$v_s = \frac{g_P^S}{S_A}$	$\frac{\text{m}^3}{\text{kg}}$	The subscript of $v_s$ is not a derivative. See comments for (S9.8)

Table S10 (Air\_2): Thermodynamic properties of humid air, depending on the mass fraction of dry air in humid air in  $\text{kg kg}^{-1}$ , absolute temperature in K and mass density in  $\text{kg m}^{-3}$  computed from the Helmholtz potential  $f^{\text{AV}}$  (= air\_f\_si). (See Part I, section 3.4.)

<b>(S10) Air_2 Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_f_si: Helmholtz potential of humid air and its first and second derivatives with respect to the mass fraction of dry air in humid air, temperature, and humid-air density (S10.1)</b>			
air_f_si(l, m, n, a, t, d) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K d = $\rho$ / ( $\text{kg m}^{-3}$ )	$\frac{\partial^{l+m+n}}{\partial A^l \partial T^m \partial \rho^n} f^{\text{AV}}(A, T, \rho)$	$\frac{\text{J m}^{3n}}{\text{K}^m \text{kg}^{n+1}}$	$0 \leq l, m, n$ $l + m + n \leq 2$ $f^{\text{AV}}$ is the full Helmholtz function for moist air including dry air, water vapour and mixed contributions
<b>air_f_cp_si: specific isobaric heat capacity (S10.2)</b>			
air_f_cp_si(a, t, d) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K d = $\rho$ / ( $\text{kg m}^{-3}$ )	$c_p = T \left[ \frac{\rho (f_{T\rho}^{\text{AV}})^2}{2f_{\rho\rho}^{\text{AV}} + \rho f_{\rho\rho}^{\text{AV}}} - f_{TT}^{\text{AV}} \right]$	$\frac{\text{J}}{\text{kg K}}$	$f^{\text{AV}}$ is the full Helmholtz function for moist air and subscripts indicate partial differentiation
<b>air_f_cv_si: specific isochoric heat capacity (S10.3)</b>			
air_f_cv_si(a, t, d) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K d = $\rho$ / ( $\text{kg m}^{-3}$ )	$c_v = -T f_{TT}^{\text{AV}}$	$\frac{\text{J}}{\text{kg K}}$	see comment for (S10.2)
<b>air_f_enthalpy_si: specific enthalpy (S10.4)</b>			
air_f_enthalpy_si(a, t, d) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K d = $\rho$ / ( $\text{kg m}^{-3}$ )	$h = f^{\text{AV}} - T f_T^{\text{AV}} + \rho f_{\rho}^{\text{AV}}$	$\frac{\text{J}}{\text{kg}}$	see comment for (S10.2)
<b>air_f_entropy_si: specific entropy (S10.5)</b>			
air_f_entropy_si(a, t, d) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K d = $\rho$ / ( $\text{kg m}^{-3}$ )	$\eta = -f_T^{\text{AV}}$	$\frac{\text{J}}{\text{kg K}}$	see comment for (S10.2)

Table S10 (Air\_2), continued

<b>(S10) Air_2 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_f_expansion_si: thermal expansion (S10.6)</b>			
air_f_expansion_si (a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\alpha = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right)_{A,P}$	$\frac{1}{\text{K}}$	see comment for (S10.2)
<b>air_f_gibbs_energy_si: specific Gibbs energy (S10.7)</b>			
air_f_gibbs_energy_si (a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$g^{\text{AV}} = f^{\text{AV}} + \rho f_{\rho}^{\text{AV}}$	$\frac{\text{J}}{\text{kg}}$	see comment for (S10.2)
<b>air_f_internal_energy_si: specific internal energy (S10.8)</b>			
air_f_internal_energy_si (a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$u = f^{\text{AV}} - T f_T^{\text{AV}}$	$\frac{\text{J}}{\text{kg}}$	see comment for (S10.2)
<b>air_f_kappa_s_si: isentropic compressibility (S10.9)</b>			
air_f_kappa_s_si(a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\kappa_s = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right)_{A,\eta}$	$\frac{1}{\text{Pa}}$	η is specific entropy
<b>air_f_kappa_t_si: isothermal compressibility (S10.10)</b>			
air_f_kappa_t_si(a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\kappa_T = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right)_{A,T}$	$\frac{1}{\text{Pa}}$	
<b>air_f_lapserate_si: moist (sometimes referred to as "dry")-adiabatic lapse rate (S10.11)</b>			
air_f_lapserate_si(a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\Gamma = \left( \frac{\partial T}{\partial P} \right)_{A,\eta}$	$\frac{\text{K}}{\text{Pa}}$	η is specific entropy

Table S10 (Air\_2), continued

<b>(S10) Air_2 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_f_mix_si</b> : This function implements the cross-component interaction contribution to the specific helmholtz energy of humid air and its derivatives with respect to the mass fraction of dry air in humid air, temperature, and humid-air density (S10.12)			
<b>air_f_mix_si</b> (l, m, n, a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\frac{\partial^{l+m+n} f^{\text{mix}}}{\partial a^l \partial T^m \partial \rho^n}$	$\frac{\text{J m}^{3n}}{\text{K}^m \text{kg}^{n+1}}$	0 ≤ l, m, n l + m + n ≤ 2 f <sup>mix</sup> is the cross-component contribution to the Helmholtz function for moist air.
<b>air_f_pressure_si</b> : pressure (S10.13)			
<b>air_f_pressure_si</b> (a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$\rho^2 f_{\rho}^{\text{AV}}$	Pa	see comment for (S10.2)
<b>air_f_soundspeed_si</b> : sound speed (S10.14)			
<b>air_f_soundspeed_si</b> (a, t, d) a = A / (kg kg <sup>-1</sup> ) t = T / K d = ρ / (kg m <sup>-3</sup> )	$c = \rho \sqrt{f_{\rho\rho}^{\text{AV}} - \frac{(f_{T\rho}^{\text{AV}})^2}{f_{TT}^{\text{AV}}} + 2 \frac{f_{\rho}^{\text{AV}}}{\rho}}$	$\frac{\text{m}}{\text{s}}$	see comment for (S10.2)
<b>chk_iapws10_table()</b> : Check routine for the thermodynamic properties of moist air (S10.15)			
<b>chk_iapws10_table()</b> accepts numerical arguments of 13 and 14 to produce comparisons of tables 13 and 14 of IAPWS-10, the proposed IAPWS guideline on the representation of moist air in the presence of seawater and ice, expected to be officially accepted in 2010. Table 13 deals with the Helmholtz function for humid air at specified values of the mass fraction of dry air in humid air, temperature and humid air density. Table 14 deals with air saturated with water vapour. Successive rows of each table are produced with the published values above and the locally calculated values below. The published results are shown to the number of digits expected to be reproduced by double precision code.			

### Level 3: Implicit Properties

Table S11 (Flu\_3a): This module implements the Gibbs functions for pure liquid water and for water vapour. We use the abbreviations  $f^w$  for  $f^F(T, \rho^w)$  and  $f^v$  for  $f^F(T, \rho^v)$ . The function  $f^F$  is the same in the two cases. (See Part I, sections 4.1, ax4.1.)

#### **(S11) Flu\_3a Module**

Function call	Mathematical equation	Unit	Comments
<b>get_it_ctrl_density(key):</b> parameter retrieval routine for the iterative solver (S11.1)			
get_it_ctrl_density(key)  key is a character string that can take the values IT_STEPS, INIT_LIQ_DENS, INIT_VAP_DENS, TOL_LIQ_DENS, TOL_VAP_DENS, METHOD_LIQ, METHOD_VAP, DENS2_LIQ or DENS2_VAP	This function returns control parameters as set for the iteration method used to compute fluid density from temperature and pressure. The value associated with key is returned in the function name get_it_ctrl_density. See the information on set_it_ctrl_density (below) and comments in the code for further information.		
<b>liq_density_si:</b> density of liquid water (S11.2)			
liq_density_si(t, p) t = T / K p = P / Pa	$\rho^w$ from solving $P = (\rho^w)^2 f_\rho^w$	$\frac{\text{kg}}{\text{m}^3}$	$f^F$ is the Helmholtz function for pure fluid water, liquid or vapour
<b>liq_g_si:</b> Gibbs potential of liquid water (S11.3)			
liq_g_si(m, n, t, p) t = T / K p = P / Pa	$g^w = f^w + \rho^w f_\rho^w \frac{\partial^{n+m}}{\partial T^m \partial P^n} g^w(T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	m, n ≥ 0 m + n ≤ 2



Table S11 (Flu\_3a) (continued)

<b>(S11) Flu_3a Module cont'd</b>			
Function call	Comments		
<b>set_it_ctrl_density(key): parameter setting routine for the iterative solver (S11.4)</b>			
<p>set_it_ctrl_density(key)</p> <p>key is a character string that can take the values            IT_STEPS,            INIT_LIQ_DENS,            INIT_VAP_DENS,            TOL_LIQ_DENS,            TOL_VAP_DENS,            METHOD_LIQ,            METHOD_VAP,            DENS2_LIQ or            DENS2_VAP</p>	<p>This subroutine allows the user to change the default settings for the iterative solver used to determine fluid water density when given temperature and pressure. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string "key" is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_LIQ_DENS and INIT_VAP_DENS allows the initial values of the liquid and vapour densities to be reset, TOL_LIQ_DENS and TOL_VAP_DENS allows the convergence criteria to be changed, METHOD_LIQ and METHOD_VAP allows the user to select a different iterative solver (Newton, Brent and Secant are available). DENS2_LIQ and DENS2_VAP allow the user to reset the second density estimates when using either Brent or Secant methods.</p>		
<b>vap_density_si: density of water vapour (S11.5)</b>			
<p>vap_density_si            (t, p)            t = T / K            p = P / Pa</p>	<p><math>\rho^v</math> from solving  <math>P = (\rho^v)^2 f_\rho^v</math></p>	<p><math>\frac{\text{kg}}{\text{m}^3}</math></p>	<p><math>f^F</math> is the Helmholtz function for pure fluid water, liquid or vapour</p>
<b>vap_g_si: Gibbs potential of water vapour (S11.6)</b>			
<p>vap_g_si            (m, n, t, p)            t = T / K            p = P / Pa</p>	<p><math>g^v = f^v + \rho^v f_\rho^v</math>  <math>\frac{\partial^{m+n}}{\partial T^m \partial P^n} g^v(T, P)</math></p>	<p><math>\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}</math></p>	<p>m, n ≥ 0            m + n ≤ 2</p>

Table S12 (Sea\_3a): Thermodynamic properties of seawater, depending on absolute salinity in  $\text{kg kg}^{-1}$ , absolute temperature in K and absolute pressure in Pa. All quantities here may be determined directly from  $g^{\text{SW}}$ , the Gibbs potential function for seawater which is composed of a pure water component,  $g^{\text{W}}$ , expressed in terms of the IAPWS-95 Helmholtz function and a saline component,  $g^{\text{S}}$  as in module Sal\_2. To use the Helmholtz function in this context, the density of pure water with the specified temperature and pressure must be determined iteratively as in the module Flu\_3a. (See Part I, section 4.2.)

**(S12) Sea\_3a Module**

Function call	Mathematical equation	Unit	Comments
chk_iapws08_table8a, b, c: Check routines for the thermodynamic properties of seawater (S12.1)			
chk_iapws08_table8a, b, c are routines that produce comparisons between published results for seawater and locally calculated results. Table 8a produces comparisons for a variety of thermodynamic properties at $(S, T, P) = (0.035164 \text{ kg/kg}, 273.15 \text{ K}, 101325 \text{ Pa})$ . Each property is compared for the pure water and saline contributions as well as the combined solution. Tables 8b and 8c compare the corresponding results for $(S, T, P) = (0.1 \text{ kg/kg}, 353 \text{ K}, 101325 \text{ Pa})$ and $(0.1 \text{ kg/kg}, 353 \text{ K}, 10^8 \text{ Pa})$ , respectively. The published results are shown to the number of digits expected to be reproduced by double precision code.			
sea_chempot_h2o_si: chemical potential of water in seawater (S12.2)			
sea_chempot_h2o_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\mu = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
sea_chempot_rel_si: relative chemical potential of seawater (S12.3)			
sea_chempot_rel_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\mu = g_{S_A}^{\text{SW}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
sea_cp_si: specific isobaric heat capacity (S12.4)			
sea_cp_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$c_P = -T g_{TT}^{\text{SW}}$	$\frac{\text{kg}}{\text{J}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
sea_density_si: density of seawater (S12.5)			
sea_density_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\rho = \frac{1}{g_P^{\text{SW}}}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)

Table S12 (Sea\_3a), continued

<b>(S12) Sea_3a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_enthalpy_si: specific enthalpy</b> (S12.6)			
sea_enthalpy_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$h = g^{\text{SW}} - T g_T^{\text{SW}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_entropy_si: specific entropy</b> (S12.7)			
sea_entropy_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\eta = -g_T^{\text{SW}}$	$\frac{\text{J}}{\text{kg K}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_g_si: Gibbs function of seawater</b> (S12.8)			
sea_g_si(l, m, n, s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\frac{\partial^{l+n+m}}{\partial S_A^l \partial T^m \partial P^n} g^{\text{SW}}(S_A, T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{kg K}^m}$	$l, m, n \geq 0$ $l + m + n \leq 2$
<b>sea_g_contraction_t_si: haline contraction coefficient</b> (S12.9)			
sea_g_contraction_t_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{T,P}$ $= -\frac{g_{S_A P}^{\text{SW}}}{g_P^{\text{SW}}}$	1	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_g_expansion_t_si: thermal expansion coefficient</b> (S12.10)			
sea_g_expansion_t_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\alpha = \frac{1}{v} \left( \frac{\partial v}{\partial T} \right)_{S_A, P}$ $= \frac{g_{TP}^{\text{SW}}}{g_P^{\text{SW}}}$	$\frac{1}{\text{K}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_gibbs_energy_si: specific Gibbs energy</b> (S12.11)			
sea_gibbs_energy_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$g = g^{\text{SW}}(S_A, T, P)$	$\frac{\text{J}}{\text{kg}}$	Note that derivatives are available through (S12.8)
<b>sea_internal_energy_si: specific internal energy</b> (S12.12)			
sea_internal_energy_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$u = g^{\text{SW}} - T g_T^{\text{SW}} - P g_P^{\text{SW}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)

Table S12 (Sea\_3a), continued

<b>(S12) Sea_3a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_kappa_s_si: isentropic compressibility (S12.13)</b>			
sea_kappa_s_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\kappa_s = -\frac{1}{v} \left( \frac{\partial v}{\partial P} \right)_{S_A, \eta}$ $= \frac{(g_{TP}^{\text{SW}})^2 - g_{TT}^{\text{SW}} g_{PP}^{\text{SW}}}{g_P^{\text{SW}} g_{TT}^{\text{SW}}}$	$\frac{1}{\text{Pa}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_kappa_t_si: isothermal compressibility (S12.14)</b>			
sea_kappa_t_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\kappa_T = -\frac{1}{v} \left( \frac{\partial v}{\partial P} \right)_{S_A, T}$ $= -\frac{g_{PP}^{\text{SW}}}{g_P^{\text{SW}}}$	$\frac{1}{\text{Pa}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_lapserate_si: adiabatic lapse rate (S12.15)</b>			
sea_lapserate_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\Gamma = \left( \frac{\partial T}{\partial P} \right)_{S_A, \eta}$ $= -\frac{g_{TP}^{\text{SW}}}{g_{TT}^{\text{SW}}}$	$\frac{\text{K}}{\text{Pa}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_osm_coeff_si: osmotic coefficient (S12.16)</b>			
sea_osm_coeff_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\phi = -\frac{(g^{\text{S}} - S_A g_{S_A}^{\text{S}})(1 - S_A)}{S_A R_S T}$	1	$g^{\text{S}}$ is the saline part of the Gibbs function, $R_S$ the specific gas constant of sea salt
<b>sea_soundspeed_si: speed of sound (S12.17)</b>			
sea_soundspeed_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$c = g_P^{\text{SW}} \sqrt{\frac{g_{TT}^{\text{SW}}}{(g_{TP}^{\text{SW}})^2 - g_{TT}^{\text{SW}} g_{PP}^{\text{SW}}}}$	$\frac{\text{m}}{\text{s}}$	$g^{\text{SW}}$ is the Gibbs function (S12.8)
<b>sea_temp_maxdensity_si: The temperature of maximum density for specified salinity and pressure (S12.18)</b>			
sea_temp_maxdensity_si(s, p) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$	Iterative solution of $\frac{\partial^2 g^{\text{SW}}}{\partial T \partial P} = 0$	K	A local extremum must be a maximum

Table S13 (Air\_3a): This module implements the Gibbs potential function required for the determination of the thermodynamic properties of humid air. This implementation requires the numerical determination of the `air_density` in order to make use of the IAPWS-95 Helmholtz function for pure water. (See Part I, section 4.4.)

<b>(S13) Air_3a Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_density_si</b> : humid air density as a function of the mass fraction of dry air in humid air, temperature and pressure from numerical iteration of the helmholtz function derivative (S13.1)			
<code>air_density_si(a, t, p)</code> $a = A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\rho^{\text{AV}} = (g_P^{\text{AV}})^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{AV}}$ is the Gibbs function for humid air. Subscript $p$ indicates partial differentiation.
<b>air_g_si</b> : Gibbs function of humid air and its first and second derivatives (S13.2)			
<code>air_g_si(l, m, n, a, t, p)</code> $a = A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\frac{\partial^{l+m+n}}{\partial A^l \partial T^m \partial P^n} g^{\text{AV}}(A, T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	$n, m, l \geq 0$ $n + m + l \leq 2$
<b>get_it_ctrl_airdensity</b> : get the value of a parameter used for the iterative solver (S13.3)			
<code>get_it_ctrl_airdensity(key)</code> <code>key</code> is a character string that can take the values IT_STEPS, INIT_AIR_DENS, TOL_AIR_DENSITY, METHOD_AIR or DENS2_AIR.	This function returns control parameters as set for the iteration method used to compute humid-air density from mass fraction of dry air in humid air, temperature and pressure. The value associated with <code>key</code> is returned in the function name <code>get_it_ctrl_airdensity</code> . See the information on <code>set_it_ctrl_airdensity</code> (below) and comments in the code for further information.		
<b>set_it_ctrl_airdensity</b> : set parameters for iterative solution (S13.4)			
<code>set_it_ctrl_airdensity(key, value)</code> <code>key</code> is a character string that can take the values IT_STEPS, INIT_AIR_DENS, TOL_AIR_DENSITY, METHOD_AIR or DENS2_AIR.  <code>value</code> is a 64 bit variable used to specify a control parameter associated with the choice specified by <code>key</code>	This subroutine allows the user to change the default settings for the iterative solver used to determine humid-air density. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string "key" is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_AIR_DENS allows the initial value of the density estimate to be reset, TOL_AIR_DENSITY allows the convergence criterion to be changed, METHOD_AIR allows the user to select a different iterative solver (Newton, Brent and Secant are available). DENS2_AIR allows the user to reset the second density estimate when using either Brent or Secant methods.		

Table S14 (Flu\_3b): This module contains functions to determine the thermodynamic properties of liquid water and of water vapour determined from the corresponding Gibbs functions, depending on absolute temperature and absolute pressure. The relations between the Gibbs and Helmholtz functions and their derivatives are used to express these quantities in terms of the Helmholtz function once the density is determined from temperature and pressure. (See Part I, sections 4.1, ax4.1.)

<b>(S14) Flu_3b Module</b>			
Function call	Mathematical equation	Unit	Comments
liq_cp_si: specific isobaric heat capacity of liquid water			(S14.1)
liq_cp_si(t, p) t = T / K p = P / Pa	$c_p^w = T \left[ \frac{\rho (f_{T\rho}^w)^2}{2f_\rho^w + \rho f_{\rho\rho}^w} - f_{TT}^w \right]$	$\frac{\text{J}}{\text{kg K}}$	$f^w$ is the Helmholtz function for fluid water with the density chosen to correspond to liquid water. Subscripts indicate partial differentiation.
liq_cv_si: specific isochoric heat capacity of liquid water			(S14.2)
flu_cv_si(t, p) t = T / K p = P / Pa	$c_v^w = -T f_{TT}^w$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S14.1)
liq_enthalpy_si: specific enthalpy of liquid water			(S14.3)
liq_enthalpy_si(t, p) t = T / K p = P / Pa	$h^w = f^w - T f_T^w + \rho f_\rho^w$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.1)
liq_entropy_si: specific entropy of liquid water			(S14.4)
liq_entropy_si(t, p) t = T / K p = P / Pa	$\eta^w = -f_T^w$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S14.1)
liq_expansion_si: thermal expansion coefficient for liquid water			(S14.5)
liq_expansion_si(t, p) t = T / K p = P / Pa	$\alpha^w = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right) \Big _p$	$\frac{1}{\text{K}}$	
liq_gibbs_energy_si: specific Gibbs energy of liquid water			(S14.6)
liq_gibbs_energy_si (t, p) t = T / K p = P / Pa	$g^w = f^w + \rho f_\rho^w$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.1)
liq_internal_energy_si: specific internal energy of liquid water			(S14.7)
liq_internal_energy_si (t, p) t = T / K p = P / Pa	$u^w = f^w - T f_T^w$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.1)

Table S14 (Flu\_3b), continued

<b>(S14) Flu_3b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_kappa_s_si: isentropic compressibility of liquid water</b> (S14.8)			
liq_kappa_s_si(t, p) t = T / K p = P / Pa	$\kappa_s^w = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right) \Big _{\eta}$	$\frac{1}{\text{Pa}}$	$\eta$ is specific entropy
<b>liq_kappa_t_si: isothermal compressibility of liquid water</b> (S14.9)			
liq_kappa_t_si(t, p) t = T / K p = P / Pa	$\kappa_T^w = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right) \Big _T$	$\frac{1}{\text{Pa}}$	
<b>liq_lapserate_si: adiabatic lapse rate for liquid water</b> (S14.10)			
liq_lapserate_si(t, p) t = T / K p = P / Pa	$\Gamma^w = \left( \frac{\partial T}{\partial P} \right) \Big _{\eta}$	$\frac{\text{K}}{\text{Pa}}$	$\eta$ is specific entropy
<b>liq_soundspeed_si: sound speed in liquid water</b> (S14.11)			
liq_soundspeed_si (t, p) t = T / K p = P / Pa	$c^w = \sqrt{\rho^2 \frac{f_{TT}^w f_{\rho\rho}^w - (f_{T\rho}^w)^2}{f_{TT}^w} + 2\rho f_{\rho}^w}$	$\frac{\text{m}}{\text{s}}$	See comments for (S14.1)
<b>vap_cp_si: specific isobaric heat capacity of water vapour</b> (S14.12)			
vap_cp_si(t, p) t = T / K p = P / Pa	$c_p^v = T \left[ \frac{\rho (f_{T\rho}^v)^2}{2f_{\rho}^v + \rho f_{\rho\rho}^v} - f_{TT}^v \right]$	$\frac{\text{J}}{\text{kg K}}$	$f^v$ is the Helmholtz function for fluid water with the density chosen to correspond to water vapour. Subscripts indicate partial differentiation.
<b>vap_cv_si: specific isochoric heat capacity of water vapour</b> (S14.13)			
vap_cv_si(t, p) t = T / K p = P / Pa	$c_v^v = -T f_{TT}^v$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S14.12)
<b>vap_enthalpy_si: specific enthalpy of water vapour</b> (S14.14)			
vap_enthalpy_si(t, p) t = T / K p = P / Pa	$h^v = f^v - T f_T^v + \rho f_{\rho}^v$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.12)
<b>vap_entropy_si: specific entropy of water vapour</b> (S14.15)			
vap_entropy_si(t, p) t = T / K p = P / Pa	$\eta^v = -f_T^v$	$\frac{\text{J}}{\text{kg K}}$	See comments for (S14.12)

Table S14 (Flu\_3b), continued

<b>(S14) Flu_3b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>vap_expansion_si</b> : thermal expansion coefficient for water vapour (S14.16)			
vap_expansion_si(t, p) t = T / K p = P / Pa	$\alpha^v = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right) \Big _p$	$\frac{1}{\text{K}}$	
<b>vap_gibbs_energy_si</b> : specific Gibbs energy of water vapour (S14.17)			
vap_gibbs_energy_si (t, p) t = T / K p = P / Pa	$g^v = f^v + \rho f_\rho^v$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.12)
<b>vap_internal_energy_si</b> : specific internal energy of water vapour (S14.18)			
vap_internal_energy_si (t, p) t = T / K p = P / Pa	$u^v = f^v - T f_T^v$	$\frac{\text{J}}{\text{kg}}$	See comments for (S14.12)
<b>vap_kappa_s_si</b> : isentropic compressibility of water vapour (S14.19)			
vap_kappa_s_si(t, p) t = T / K p = P / Pa	$\kappa_s^v = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right) \Big _\eta$	$\frac{1}{\text{Pa}}$	$\eta$ is specific entropy
<b>vap_kappa_t_si</b> : isothermal compressibility of water vapour (S14.20)			
vap_kappa_t_si(t, p) t = T / K p = P / Pa	$\kappa_T^v = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial P} \right) \Big _T$	$\frac{1}{\text{Pa}}$	
<b>vap_lapserate_si</b> : adiabatic lapse rate for water vapour (S14.21)			
vap_lapserate_si(t, p) t = T / K p = P / Pa	$\Gamma^v = \left( \frac{\partial T}{\partial P} \right) \Big _\eta$	$\frac{\text{K}}{\text{Pa}}$	$\eta$ is specific entropy
<b>vap_soundspeed_si</b> : sound speed in water vapour (S14.22)			
vap_soundspeed_si (t, p) t = T / K p = P / Pa	$c^v = \sqrt{\rho^2 \frac{f_{TT}^v f_{\rho\rho}^v - (f_{T\rho}^v)^2}{f_{TT}^v} + 2\rho f_\rho^v}$	$\frac{\text{m}}{\text{s}}$	See comments for (S14.12)



Table S15 (Sea\_3b): Thermodynamic properties of seawater, depending on absolute salinity in  $\text{kg kg}^{-1}$ , entropy in  $\text{J kg}^{-1} \text{K}^{-1}$  and absolute pressure in Pa. The quantities here are conveniently expressed by using the enthalpy  $h^{\text{SW}}$  as a potential function. (See Part I, sections 4.3, ax4.3.)

<b>(S15) Sea_3b Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_h_si: specific enthalpy as a thermodynamic potential for seawater (S15.1)</b>			
sea_h_si(l, m, n, s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\frac{\partial^{l+m+n}}{\partial S_A^l \partial \eta^m \partial P^n} h^{\text{SW}}(S_A, \eta, P)$	$\frac{\text{J}^{1-m} \text{m}^{3n} \text{K}^m}{\text{kg}^{n-m+1}}$	$l, n, m \geq 0$ $l + n + m \leq 2$
<b>sea_h_contraction_h_si: haline contraction coefficient with respect to potential enthalpy (S15.2)</b>			
sea_h_contraction_h_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\beta^\theta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{h^\theta, P}$ $= \frac{h_{\eta P} h_S^\theta - h_{SP} h_\eta^\theta}{h_P h_\eta^\theta}$	1	$h^\theta$ is the potential enthalpy (S15.9)
<b>sea_h_contraction_t_si: haline contraction coefficient (S15.3)</b>			
sea_h_contraction_t_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\beta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{T, P}$ $= \frac{h_{\eta P} h_{S\eta} - h_{SP} h_{\eta\eta}}{h_P h_{\eta\eta}}$	1	$h$ is the enthalpy of seawater (S15.1)
<b>sea_h_contraction_theta_si: haline contraction coefficient with respect to potential temperature (S15.4)</b>			
sea_h_contraction_theta_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\beta^\theta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{\theta, P}$ $= \frac{h_{\eta P} h_{S\eta}^\theta - h_{SP} h_{\eta\eta}^\theta}{h_P h_{\eta\eta}^\theta}$	1	$h^\theta$ is the potential enthalpy (S15.9)
<b>sea_h_expansion_h_si: thermal expansion coefficient with respect to potential enthalpy (S15.5)</b>			
sea_h_expansion_h_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\alpha^h = \frac{1}{v} \left( \frac{\partial v}{\partial h^\theta} \right)_{S_A, P} = \frac{h_{\eta P}}{h_P h_\eta^\theta}$	$\frac{\text{kg}}{\text{J}}$	$h^\theta$ is the potential enthalpy (S15.9)

Table S15 (Sea\_3b), continued

<b>(S15) Sea_3b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_h_expansion_t_si: thermal expansion coefficient (S15.6)</b>			
sea_h_expansion_t_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\alpha^T = \frac{1}{v} \left( \frac{\partial v}{\partial T} \right)_{S_A, P} = \frac{h_{\eta P}}{h_p h_{\eta \eta}}$	$\frac{1}{\text{K}}$	$h$ is the enthalpy of seawater (S15.1)
<b>sea_h_expansion_theta_si: thermal expansion coefficient with respect to potential temperature (S15.7)</b>			
sea_h_expansion_theta_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$\alpha^\theta = \frac{1}{v} \left( \frac{\partial v}{\partial \theta} \right)_{S_A, P} = \frac{h_{\eta P}}{h_p h_{\eta \eta}^\theta}$	$\frac{1}{\text{K}}$	$h^\theta$ is the potential enthalpy (S15.9)
<b>sea_potdensity_si: potential density (S15.8)</b>			
sea_potdensity_si(s, t, p, pr) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$ $\text{pr} = P_r / \text{Pa}$	$\rho^\theta = \frac{1}{h_p^\theta}$	$\frac{\text{kg}}{\text{m}^3}$	$P_r$ is the absolute reference pressure (usually 101325 Pa, at the sea surface)
<b>sea_potenthalpy_si: potential enthalpy referenced to the sea surface (S15.9)</b>			
sea_potenthalpy_si (s, t, p, pr) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$ $\text{pr} = P_r / \text{Pa}$	$h^\theta = h^{\text{sw}}(S_A, \eta, P_r)$ <p style="text-align: center;">with <math>\eta</math> from</p> $\eta = -g_T^{\text{sw}}(S_A, T, P)$	$\frac{\text{J}}{\text{kg}}$	$P_r = 101325 \text{ Pa}$ is the sea surface pressure
<b>sea_pottemp_si: potential temperature (S15.10)</b>			
sea_pottemp_si(s, t, p, pr) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$ $\text{pr} = P_r / \text{Pa}$	$\theta = h_\eta^{\text{sw}}(S_A, \eta, P_r)$ <p style="text-align: center;">with <math>\eta</math> from</p> $\eta = -g_T^{\text{sw}}(S_A, T, P)$	K	$\theta$ is the absolute potential temperature
<b>sea_temperature_si: absolute temperature (S15.11)</b>			
sea_temperature_si (s, eta, p) $s = S_A / (\text{kg kg}^{-1})$ $\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ $p = P / \text{Pa}$	$T(S_A, \eta, P)$ <p style="text-align: center;">from solving</p> $\eta = -g_T^{\text{sw}}(S_A, T, P)$	K	$T$ is the in-situ temperature of seawater from salinity, entropy and pressure

Table S15 (Sea\_3b), continued

<b>(S15) Sea_3b Module, cont'd</b>	
Function call	Comments
set_it_ctrl_pottemp: set parameters for Newton iteration (S15.12)	
set_it_ctrl_pottemp(key, value)  key: a character string that can take the values IT_STEPS, INIT_THETA or TOL_THETA.  value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine seawater temperature from entropy. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_THETA allows the initial value of the primary temperature estimate to be reset, TOL_THETA allows the convergence criterion to be changed.

Table S16 (Air\_3b): Thermodynamic properties of humid air in terms of the mass fraction of dry air in humid air in  $\text{kg kg}^{-1}$ , the absolute temperature in K and the pressure in Pa. The quantities here are all conveniently expressed in terms of the Gibbs potential function (S16.9). (See Part I, section 4.4.)

<b>(S16) Air_3b Module</b>			
Function call	Mathematical equation	Unit	Comments
air_g_chempot_vap_si: chemical potential of vapour in humid air (S16.1)			
air_g_chempot_vap_si(a, t, p) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K p = P / Pa	$g^{\text{AV}} - A \frac{\partial g^{\text{AV}}}{\partial A}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{AV}}$ is the Gibbs function for moist air (S13.2) and subscripts indicate partial differentiation.
air_g_compressibilityfactor_si: compressibility factor for moist air (S16.2)			
air_g_compressibilityfactor_si(a, t, p) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K p = P / Pa	$\frac{P}{\rho R_{\text{AV}} T}$ $R_{\text{AV}} = R \left[ \frac{1-A}{M^{\text{W}}} + \frac{A}{M^{\text{A}}} \right]$	1	$R_{\text{AV}}$ is the specific gas constant for moist air.
air_g_contraction_si: contraction coefficient of humid air (S16.3)			
air_g_contraction_si(a, t, p) a = A / ( $\text{kg kg}^{-1}$ ) t = T / K p = P / Pa	$\beta = - \frac{g_{\text{AP}}^{\text{AV}}}{g_{\text{P}}^{\text{AV}}}$	$\frac{\text{kg}}{\text{kg}}$	see comments for (S16.1)

Table S16 (Air\_3b), continued

<b>(S16) Air_3b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_g_cp_si: specific isobaric heat capacity of humid air</b> (S16.4)			
air_g_cp_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$c_p = -Tg_{TT}^{AV}$	$\frac{\text{J}}{\text{kg K}}$	see comments for (S16.1)
<b>air_g_cv_si: specific isochoric heat capacity of humid air</b> (S16.5)			
air_g_cv_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$c_v = T \frac{(g_{TP}^{AV})^2 - g_{TT}^{AV} g_{PP}^{AV}}{g_{PP}^{AV}}$	$\frac{\text{J}}{\text{kg K}}$	see comments for (S16.1)
<b>air_g_density_si: density of humid air</b> (S16.6)			
air_g_density_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\rho^{AV} = (g_P^{AV})^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	see comments for (S16.1)
<b>air_g_enthalpy_si: specific enthalpy of humid air</b> (S16.7)			
air_g_enthalpy_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$h = g^{AV} - Tg_T^{AV}$	$\frac{\text{J}}{\text{kg}}$	see comments for (S16.1)
<b>air_g_entropy_si: specific entropy of humid air</b> (S16.8)			
air_g_entropy_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\eta = -g_T^{AV}$	$\frac{\text{J}}{\text{kg K}}$	see comments for (S16.1)
<b>air_g_expansion_si: thermal expansion of humid air</b> (S16.9)			
air_g_expansion_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\alpha = \frac{g_{TP}^{AV}}{g_P^{AV}}$	$\frac{1}{\text{K}}$	see comments for (S16.1)
<b>air_g_gibbs_energy_si: Gibbs energy of humid air</b> (S16.10)			
air_g_gibbs_energy_si (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$g^{AV}$	$\frac{\text{J}}{\text{kg}}$	$g^{AV}$ is the full Gibbs energy including dry air, vapour and mixed contributions. Note derivatives are given by (S13.2).

Table S16 (Air\_3b), continued

<b>(S16) Air_3b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_g_internal_energy_si: specific internal energy of humid air</b> (S16.11)			
<b>air_g_internal_energy_si</b> (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$u = g^{\text{AV}} - T g_T^{\text{AV}} - P g_P^{\text{AV}}$	$\frac{\text{J}}{\text{kg}}$	see comments for (S16.1)
<b>air_g_kappa_s_si: isentropic compressibility of humid air</b> (S16.12)			
<b>air_g_kappa_s_si</b> (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\kappa_S = \frac{(g_{TP}^{\text{AV}})^2 - g_{TT}^{\text{AV}} g_{PP}^{\text{AV}}}{g_P^{\text{AV}} g_{TT}^{\text{AV}}}$	$\frac{1}{\text{Pa}}$	see comments for (S16.1)
<b>air_g_kappa_t_si: isothermal compressibility of humid air</b> (S16.13)			
<b>air_g_kappa_t_si</b> (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\kappa_T = - \frac{g_{PP}^{\text{AV}}}{g_P^{\text{AV}}}$	$\frac{1}{\text{K}}$	see comments for (S16.1)
<b>air_g_lapserate_si: dry-adiabatic lapse rate of humid air</b> (S16.14)			
<b>air_g_lapserate_si</b> (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$\Gamma = - \frac{g_{TP}^{\text{AV}}}{g_{TT}^{\text{AV}}}$	$\frac{\text{K}}{\text{Pa}}$	see comments for (S16.1)
<b>air_g_soundspeed_si: sound speed in humid air</b> (S16.15)			
<b>air_g_soundspeed_si</b> (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$c = g_P^{\text{AV}} \sqrt{\frac{g_{TT}^{\text{AV}}}{(g_{TP}^{\text{AV}})^2 - g_{TT}^{\text{AV}} g_{PP}^{\text{AV}}}}$	$\frac{\text{m}}{\text{s}}$	see comments for (S16.1)
<b>chk_Lemmon_etal(): Check routine for the thermodynamic properties of moist air</b> (S16.16)			
<p><b>chk_Lemmon_etal ()</b> accepts arguments of 1 and 2 to produce comparisons corresponding to the dry-air Helmholtz and Gibbs functions, respectively, based on the results tabulated by Lemmon et al. (2000). In each case, results for pressure, internal energy, entropy, specific heat capacities and sound speed are printed out. Successive rows of each table are printed out with the published values above and the locally calculated values below. The Lemmon et al. results include roundoff errors that make them sub-optimal as check values. Preferred check values are given by <b>chk_iapws10_table</b>, (S10.15).</p>			

Table S17 (Sea\_3c): Thermodynamic properties of seawater, depending on absolute salinity in  $\text{kg kg}^{-1}$ , a local pressure  $P$ , a reference pressure  $P_r$ , each in Pa, plus one other quantity from the four possibilities temperature, potential temperature ( $T$  or  $\theta$ , each in K), enthalpy or potential enthalpy ( $h$  or  $h^\theta$ , each in  $\text{J kg}^{-1}$ ). A character string **key** is used to specify which of these four possibilities is used as an input for a particular application. The quantities here are conveniently expressed by using the enthalpy  $h^{\text{SW}}$  as a potential function. (See Part I, sections 4.3, ax4.3.)

<b>(S17) Sea_3c Module</b>				
Function call	Mathematical equation	Unit	Comments	
<b>sea_eta_contraction_h_si: haline contraction coefficient at constant potential enthalpy</b> (S17.1)				
<b>sea_eta_contraction_h_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\beta^\ominus = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{h^\ominus, P}$ $= \frac{h_{\eta P} h_S^\ominus - h_{SP} h_\eta^\ominus}{h_P h_\eta^\ominus}$	1	$\eta$ is entropy, computed from input x which depends on the value of key as given below	
Options for key and x	key = T $x = T / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, T, P)$		$T$ is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	key = TPOT $x = \theta / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$		$\theta$ is absolute potential temperature
	key = H $x = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$		$h$ is in-situ specific enthalpy
	key = HPOT $x = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$		$h^\theta$ is potential enthalpy
<b>sea_eta_contraction_t_si: haline contraction coefficient (at constant in-situ temperature)</b> (S17.2)				
<b>sea_eta_contraction_t_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\beta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{T, P}$ $= \frac{h_{\eta P} h_{S\eta} - h_{SP} h_{\eta\eta}}{h_P h_{\eta\eta}}$	1	$\eta$ is entropy, computed from input x which depends on the value of key as given below	
Options for key and x	key = T $x = T / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, T, P)$		$T$ is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	key = TPOT $x = \theta / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$		$\theta$ is absolute potential temperature
	key = H $x = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$		$h$ is in-situ specific enthalpy
	key = HPOT $x = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$		$h^\theta$ is potential enthalpy

Table S17 (Sea\_3c), continued

<b>(S17) Sea_3c Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_eta_contraction_theta_si: haline contraction coefficient at constant potential temperature (S17.3)</b>			
<b>sea_eta_contraction_theta_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\beta^\theta = -\frac{1}{v} \left( \frac{\partial v}{\partial S_A} \right)_{\theta, P}$ $= \frac{h_{S\eta}^\theta h_{\eta P} - h_{SP} h_{\eta\eta}^\theta}{h_p h_{\eta\eta}^\theta}$	1	$\eta$ is entropy, computed from input x which depends on the value of key as given below
Options for key and x	<b>key = T</b> $x = T / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, T, P)$	$T$ is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	<b>key = TPOT</b> $x = \theta / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$	$\theta$ is absolute potential temperature
	<b>key = H</b> $x = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
	<b>key = HPOT</b> $x = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy
<b>sea_eta_density_si: density (in-situ) (S17.4)</b>			
<b>sea_eta_density_si</b> (s, h, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\rho = \frac{1}{h_p^{\text{SW}}(S_A, \eta, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$\eta$ is entropy, computed from input h which depends on the value of key as given below
Options for key and h	<b>key = H</b> $h = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
They determine how $\eta$ is determined in column 2 above.	<b>key = HPOT</b> $h = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy
<b>sea_eta_entropy_si: specific entropy (from enthalpy or potential enthalpy) (S17.5)</b>			
<b>sea_eta_entropy_si</b> (s, h, p) $s = S_A / (\text{kg kg}^{-1})$ $h = h / (\text{J kg}^{-1})$ $p = P / \text{Pa}$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$ or $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$\frac{\text{J}}{\text{kg K}}$	Either $(h, P)$ or $(h^\theta, P_r)$ can be used as input parameter pair

Table S17 (Sea\_3c), continued

<b>(S17) Sea_3c Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_eta_expansion_h_si: thermal expansion coefficient with respect to potential enthalpy</b> (S17.6)			
<b>sea_eta_expansion_h_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\alpha^h = \frac{1}{v} \left( \frac{\partial v}{\partial h^\theta} \right)_{S_A, P}$ $= \frac{h_{\eta P}}{h_p h_\eta^\theta}$	$\frac{\text{kg}}{\text{J}}$	$\eta$ is entropy, computed from input x which depends on the value of key as given below
Options for key and x	key = T $x = T / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, T, P)$	$T$ is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	key = TPOT $x = \theta / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$	$\theta$ is absolute potential temperature
	key = H $x = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
	key = HPOT $x = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy
<b>sea_eta_expansion_t_si: thermal expansion coefficient (with respect to in-situ temperature)</b> (S17.7)			
<b>sea_eta_expansion_t_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\alpha^T = \frac{1}{v} \left( \frac{\partial v}{\partial T} \right)_{S_A, P}$ $= \frac{h_{\eta P}}{h_p h_{\eta T}}$	$\frac{1}{\text{K}}$	$\eta$ is entropy, computed from input x which depends on the value of key as given below
Options for key and x	key = T $x = T / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, T, P)$	$T$ is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	key = TPOT $x = \theta / \text{K}$	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$	$\theta$ is absolute potential temperature
	key = H $x = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
	key = HPOT $x = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy



Table S17 (Sea\_3c), continued

<b>(S17) Sea_3c Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_eta_expansion_theta_si: thermal expansion coefficient with respect to potential temperature</b> (S17.8)			
<b>sea_eta_expansion_theta_si</b> (s, x, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\alpha^\theta = \frac{1}{v} \left( \frac{\partial v}{\partial \theta} \right)_{S_A, P}$ $= \frac{h_{\eta p}}{h_p h_{\eta \eta}^\theta}$	$\frac{1}{\text{K}}$	$\eta$ is entropy, computed from input x which depends on the value of key as given below
Options for key and x	key = T x = T / K	$\eta = -g_T^{\text{SW}}(S_A, T, P)$	T is in-situ absolute temperature
They determine how $\eta$ is determined in column 2 above.	key = TPOT x = $\theta$ / K	$\eta = -g_T^{\text{SW}}(S_A, \theta, P_r)$	$\theta$ is absolute potential temperature
	key = H x = h / (J kg <sup>-1</sup> )	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	h is in-situ specific enthalpy
	key = HPOT x = $h^\theta$ / (J kg <sup>-1</sup> )	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy
<b>sea_eta_potdensity_si: potential density (from enthalpy or potential enthalpy)</b> (S17.9)			
<b>sea_eta_potdensity_si</b> (s, h p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\rho = \frac{1}{h_p^{\text{SW}}(S_A, \eta, P_r)}$	$\frac{\text{kg}}{\text{m}^3}$	$\eta$ is entropy, computed from input h which depends on the value of key as given below
Options for key and h	key = H h = h / (J kg <sup>-1</sup> )	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	h is in-situ specific enthalpy
They determine how $\eta$ is determined in column 2 above.	key = HPOT h = $h^\theta$ / (J kg <sup>-1</sup> )	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy

Table S17 (Sea\_3c), continued

<b>(S17) Sea_3c Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_eta_pottemp_si: potential temperature (from enthalpy or potential enthalpy) (S17.10)</b>			
<b>sea_eta_pottemp_si</b> (s, h, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\theta = \frac{1}{h_\eta^{\text{SW}}(S_A, \eta, P_r)}$	K	$\eta$ is entropy, computed from input h which depends on the value of key as given below
Options for key and h	key = H $h = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
They determine how $\eta$ is determined in column 2 above.	key = HPOT $h = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy
<b>sea_eta_temperature_si: temperature from entropy as the thermodynamic potential (as a function of enthalpy or potential enthalpy) (S17.11)</b>			
<b>sea_eta_temperature_si</b> (s, h, p, pr, key) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$T = \frac{1}{h_\eta^{\text{SW}}(S_A, \eta, P)}$	K	$\eta$ is entropy, computed from input h which depends on the value of key as given below
Options for key and h	key = H $h = h / (\text{J kg}^{-1})$	$\eta$ from solving $h = h^{\text{SW}}(S_A, \eta, P)$	$h$ is in-situ specific enthalpy
They determine how $\eta$ is determined in column 2 above.	key = HPOT $h = h^\theta / (\text{J kg}^{-1})$	$\eta$ from solving $h^\theta = h^{\text{SW}}(S_A, \eta, P_r)$	$h^\theta$ is potential enthalpy

Table S17 (Sea\_3c), continued

<b>(S17) Sea_3c Module (cont'd)</b>	
Function call	Comments
<b>set_it_ctrl_entropy: set parameters for Newton iteration</b> (S17.12)	
<p><b>set_it_ctrl_entropy(key, value)</b></p> <p>key: a character string that can take the values IT_STEPS, MODE_ETA, INIT_ETA or TOL_ETA.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>	<p>This subroutine allows the user to specify details regarding the iterative solver used to determine seawater entropy from enthalpy. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, MODE_ETA set to 0 or 1 decides whether (0) the default rather than (1) the value set by INIT_ETA should be used, INIT_ETA allows the initial value of the primary entropy estimate to be reset, TOL_ETA allows the convergence criterion to be changed.</p>

Table S18 (Air\_3c): Thermodynamic properties of humid air determined from the enthalpy as a thermodynamic potential function. (See Part I, section 4.5.)

<b>(S18) Air_3c Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_h_si: this function implements enthalpy as a thermodynamic potential of humid air, depending on the dry-air mass fraction, entropy and pressure</b> (S18.1)			
<p><b>air_h_si(l, m, n, a, eta, p)</b></p> <p><math>a = A / (\text{kg kg}^{-1})</math></p> <p><math>\text{eta} = \eta / (\text{J kg}^{-1} \text{K}^{-1})</math></p> <p><math>p = P / \text{Pa}</math></p>	$\frac{\partial^{l+m+n}}{\partial A^l \partial \eta^m \partial P^n} h^{\text{AV}}(A, \eta, P)$	$\frac{\text{J}^{l-m-n} \text{m}^{3n}}{\text{kg}^{l-m} \text{K}^{-m}}$	<p><math>l, m, n \geq 0</math></p> <p><math>l + m + n \leq 2</math></p> <p><math>h^{\text{AV}}</math> is the Helmholtz potential for moist air.</p> <p><math>\eta</math> is entropy.</p>
<b>air_potdensity_si: potential density of humid air</b> (S18.2)			
<p><b>air_potdensity_si(a, t, p, pr)</b></p> <p><math>a = A / (\text{kg kg}^{-1})</math></p> <p><math>t = T / \text{K}</math></p> <p><math>p = P / \text{Pa}</math></p> <p><math>pr = P_r / \text{Pa}</math></p>	$\rho_\theta = \left( h_p^{\text{AV}}(A, \eta, P_r) \right)^{-1}$ <p style="text-align: center;">with <math>\eta</math> from</p> $\eta = -g_T^{\text{AV}}(A, T, P)$	$\frac{\text{kg}}{\text{m}^3}$	<p><math>g^{\text{AV}}</math> is the Gibbs potential (S13.2).</p> <p><math>\eta</math> is entropy.</p> <p><math>P_r</math> is the reference pressure.</p>

Table S18 (Air\_3c), continued

<b>(S18) Air_3c Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_potenthalpy_si: potential enthalpy of humid air</b> (S18.3)			
<b>air_potenthalpy_si</b> (a, t, p, pr) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa pr = P <sub>r</sub> / Pa	$h^\theta = h^{\text{AV}}(A, \eta, P_r)$ with $\eta$ from $\eta = -g_T^{\text{AV}}(A, T, P)$	$\frac{\text{J}}{\text{kg}}$	See comments for (S18.1) and (S18.2)
<b>air_pottemp_si: absolute potential temperature of humid air in Kelvin</b> (S18.4)			
<b>air_pottemp_si(a, t, p, pr)</b> a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa pr = P <sub>r</sub> / Pa	$T_\theta = h_\eta^{\text{AV}}(A, \eta, P_r)$ with $\eta$ from $\eta = -g_T^{\text{AV}}(A, T, P)$	K	See comments for (S18.1) and (S18.2)
<b>air_temperature_si: absolute temperature of humid air obtained as the solution of air_g_entropy_si(a, t, p) = eta for specified a, p and eta</b> (S18.5)			
<b>air_temperature_si</b> (a, eta, p) a = A / (kg kg <sup>-1</sup> ) eta = $\eta$ / (J kg <sup>-1</sup> K <sup>-1</sup> ) p = P / Pa	$T = h_\eta^{\text{AV}}(A, \eta, P)$ with $\eta$ from $\eta = -g_T^{\text{AV}}(A, T, P)$	K	See comments for (S18.1) and (S18.2)
<b>set_it_ctrl_air_pottemp</b> (S18.6)			
<b>set_it_ctrl_air_pottemp</b> (key, value)  key is a character string that can take the values: IT_STEPS, INIT_THETA or TOL_THETA.  value is a 64 bit variable used to specify a control parameter associated with the choice specified by key	This subroutine sets control parameters for the Newton iteration used to compute potential temperature corresponding to a specified reference pressure. Comments in the subroutine supply detailed information. Default values that should work are used if this routine is not called.		

Table S19 (Sea\_3d): This module provides an iterative inverse solution for absolute salinity given temperature in K, pressure in Pa and in situ density in  $\text{kg m}^{-3}$ . The solution is determined by a Newton iterative scheme. Since the resulting salinity is determined from density, we have referred to it as the "density salinity". If the water sample has the Reference Composition (Millero et al., 2008), then the resulting salinity will be the Reference Salinity. The default upper limit on salinity is  $50 \text{ g kg}^{-1}$ . At atmospheric pressure, it may be extended to  $70 \text{ g kg}^{-1}$  by setting the constant `IsExtension2010` equal to `TRUE`. (See Part I, e4.1.7)

**(S19) Sea\_3d Module**

Function call	Mathematical equation	Unit	Comments
<b>sea_sa_si: absolute salinity</b>			(S19.1)
sea_sa_si(t, p, d) $t = T / \text{K}$ $p = P / \text{Pa}$ $d = \rho / (\text{kg m}^{-3})$	$S_A$ from solving $\rho = \frac{1}{g_P^{SW}(S_A, T, P)}$	$\frac{\text{kg}}{\text{kg}}$	This "density salinity" can estimate the absolute salinity of non-standard seawater, too
<b>set_it_ctrl_salinity: set parameters for Newton iteration</b>			(S19.2)
set_it_ctrl_salinity(key, value)  key: a character string that can take the values IT_STEPS, INIT_SA or TOL_SA.  value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine seawater salinity from density. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string <code>key</code> is used to determine which function to execute: <code>IT_STEPS</code> allows <code>maxit</code> to be reset, <code>INIT_SA</code> allows the initial value of the primary salinity estimate to be reset, <code>TOL_SA</code> allows the convergence criterion to be changed.		

#### Level 4: Phase Equilibrium Properties

Table S20 (Liq\_Vap\_4): Thermodynamic properties in vapour-liquid equilibrium. Equilibrium conditions are determined by calling either `set_liq_vap_eq_at_p` or `set_liq_vap_eq_at_t`. The computed state variables  $T, P^W, P^V, g^W, g^V, \rho^W, \rho^V$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters so that the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.1, ax5.1.)

#### **(S20) Liq\_Vap\_4 Module**

Function call	Mathematical equation	Unit	Comments
<b>chk_iapws95_table8: Check routine for the thermodynamic properties of pure liquid water in equilibrium with pure water vapour. (S20.1)</b>			
chk_iapws95_table8 produces a comparison of locally calculated results with those published in Table 8 of IAPWS-95 at the vapour pressure for specified temperatures. Values of pressure, density, enthalpy and entropy are compared for temperatures of 275 K, 450 K and 625 K. The published results are shown to the number of digits expected to be reproduced by double precision calculations.			
<b>liq_vap_boilingtemperature_si: boiling temperature at specified pressure (S20.2)</b>			
liq_vap_boilingtemperature_si(p) p = P / Pa	$T$	K	$T$ from the state variables
<b>liq_vap_chempot_si: chemical potential (S20.3)</b>			
liq_vap_chempot_si()	$g^V$	$\frac{\text{J}}{\text{kg}}$	$g^V$ from the state variables
<b>liq_vap_density_liq_si: density of liquid water (S20.4)</b>			
liq_vap_density_liq_si()	$\rho^W$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^W$ from the state variables
<b>liq_vap_density_vap_si: density of water vapour (S20.5)</b>			
liq_vap_density_vap_si()	$\rho^V$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^V$ from the state variables
<b>liq_vap_enthalpy_evap_si: specific evaporation enthalpy of water (S20.6)</b>			
liq_vap_enthalpy_evap_si()	$\Delta h^{\text{evap}} = T f_T^F(T, \rho^V) - T f_T^F(T, \rho^W)$	$\frac{\text{J}}{\text{kg}}$	$T, \rho^W$ and $\rho^V$ from the state variables
<b>liq_vap_enthalpy_liq_si: specific enthalpy of liquid water (S20.7)</b>			
liq_vap_enthalpy_liq_si()	$h^W = g^W - T f_T^F(T, \rho^W)$	$\frac{\text{J}}{\text{kg}}$	$T, g^W$ and $\rho^W$ from the state variables

Table S20 (Liq\_Vap\_4), continued

<b>(S20) Liq_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
liq_vap_enthalpy_vap_si: specific enthalpy of water vapour (S20.8)			
liq_vap_enthalpy_vap_si()	$h^V = g^V - T f_T^F(T, \rho^V)$	$\frac{\text{J}}{\text{kg}}$	$T$ , $g^V$ and $\rho^V$ from the state variables
liq_vap_entropy_evap_si: specific evaporation entropy of water (S20.9)			
liq_vap_entropy_evap_si()	$\eta^{\text{evap}} = f_T^F(T, \rho^W) - f_T^F(T, \rho^V)$	$\frac{\text{J}}{\text{kg K}}$	$T$ , $\rho^W$ and $\rho^V$ from the state variables
liq_vap_entropy_liq_si: specific entropy of liquid water (S20.10)			
liq_vap_entropy_liq_si()	$\eta^W = -f_T^F(T, \rho^W)$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $\rho^W$ from the state variables
liq_vap_entropy_vap_si: specific entropy of water vapour (S20.11)			
liq_vap_entropy_vap_si()	$\eta^V = -f_T^F(T, \rho^V)$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $\rho^V$ from the state variables
liq_vap_pressure_liq_si: pressure of the liquid from a previous determination of equilibrium conditions (S20.12)			
liq_vap_pressure_liq_si()	$P^W$	Pa	$P^W$ from the state variables
liq_vap_pressure_vap_si: pressure of the vapour from a previous determination of equilibrium conditions (S20.13)			
liq_vap_pressure_vap_si()	$P^V$	Pa	$P^V$ from the state variables
liq_vap_temperature_si: temperature (S20.14)			
liq_vap_temperature_si()	$T$	K	$T$ from the state variables
liq_vap_vapourpressure_si: vapour pressure at the specified $T$ (S20.15)			
liq_vap_vapourpressure_si(t) t = T / K	$P^V$	Pa	$P^V$ from the state variables
liq_vap_volume_evap_si: specific evaporation volume (S20.16)			
liq_vap_volume_evap_si()	$v^{\text{evap}} = \frac{1}{\rho^V} - \frac{1}{\rho^W}$	$\frac{\text{m}^3}{\text{kg}}$	$\rho^W$ and $\rho^V$ from the state variables

Table S20 (Liq\_Vap\_4), continued

<b>(S20) Liq_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
set_liq_vap_eq_at_p: Set module to determine liquid-vapour equilibrium at given pressure (S20.17)			
set_liq_vap_eq_at_p(p) p = P / Pa	Solve $g^v(T, P) = g^w(T, P)$ with P specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_liq_vap_eq_at_t: Set module to determine liquid-vapour equilibrium at given temperature (S20.18)			
set_liq_vap_eq_at_t(t) t = T / K	Solve $g^v(T, P) = g^w(T, P)$ with T specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_it_ctrl_liq_vap (S20.19)			
set_it_ctrl_liq_vap(key, value) key: a character string that can take the values IT_STEPS, INIT_LIQ_DENS, INIT_VAP_DENS, INIT_TEMP or TOL_VAP_PRESS. value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of water vapour with liquid water. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_LIQ_DENS, INIT_VAP_DENS and INIT_TEMP allow the initial estimates of the liquid density, vapour density and temperature to be reset, TOL_VAP_PRESS allows the convergence criterion to be changed for vapour pressure.		



Table S21 (Ice\_Vap\_4): Thermodynamic properties of the sublimation (i.e. ice-vapour) equilibrium. Equilibrium conditions are determined by calling either `set_ice_vap_eq_at_p` or `set_ice_vap_eq_at_t`. The computed state variables  $T$ ,  $P$ ,  $g^{\text{lh}}$ ,  $g^{\text{v}}$ ,  $\rho^{\text{lh}}$ ,  $\rho^{\text{v}}$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters so that the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.3, ax5.3.)

<b>(S21) Ice_Vap_4 Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_vap_chempot_si: chemical potential</b> (S21.1)			
ice_vap_chempot_si() available after specifying the equilibrium state conditions	$g^{\text{lh}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{lh}}$ is a state variables
<b>ice_vap_density_ice_si: density of ice</b> (S21.2)			
ice_vap_density_ice_si() available after specifying the equilibrium state conditions	$\rho^{\text{lh}}$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^{\text{lh}}$ is a state variable
<b>ice_vap_density_vap_si: density of water vapour</b> (S21.3)			
ice_vap_density_vap_si() available after specifying the equilibrium state conditions	$\rho^{\text{v}}$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^{\text{v}}$ is a state variable
<b>ice_vap_enthalpy_ice_si: specific enthalpy of ice</b> (S21.4)			
ice_vap_enthalpy_ice_si() available after specifying the equilibrium state conditions	$h^{\text{lh}} = g^{\text{lh}} - Tg_T^{\text{lh}}(T, P)$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{lh}}$ , $T$ and $P$ are state variables
<b>ice_vap_enthalpy_subl_si : specific sublimation enthalpy</b> (S21.5)			
ice_vap_enthalpy_subl_si(t) $t = T / \text{K}$	$\Delta h^{\text{subl}} = Tg_T^{\text{lh}}(T, P) - Tf_T^{\text{v}}(T, \rho^{\text{v}})$	$\frac{\text{J}}{\text{kg}}$	$P$ and $\rho^{\text{v}}$ are state variables obtained with the specified $T$
<b>ice_vap_enthalpy_vap_si: specific enthalpy of water vapour</b> (S21.6)			
ice_vap_enthalpy_vap_si() available after specifying the equilibrium state conditions	$h^{\text{v}} = g^{\text{v}} - Tf_T^{\text{v}}(T, \rho^{\text{v}})$	$\frac{\text{J}}{\text{kg}}$	$T$ , $P$ , $g^{\text{v}}$ and $\rho^{\text{v}}$ are state variables
<b>ice_vap_entropy_ice_si: specific entropy of ice</b> (S21.7)			
ice_vap_entropy_ice_si() available after specifying the equilibrium state conditions	$\eta^{\text{lh}} = -g_T^{\text{lh}}(T, P)$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $P$ are state variables
<b>ice_vap_entropy_subl_si : specific sublimation entropy</b> (S21.8)			
ice_vap_entropy_subl_si(t) $t = T / \text{K}$	$\Delta \eta^{\text{subl}} = g_T^{\text{lh}}(T, P) - f_T^{\text{v}}(T, \rho^{\text{v}})$	$\frac{\text{J}}{\text{kg K}}$	$P$ and $\rho^{\text{v}}$ are state variables obtained with the specified $T$

Table S21 (Ice\_Vap\_4), continued

<b>(S21) Ice_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
ice_vap_entropy_liq_si: specific entropy of water (S21.9)			
ice_vap_entropy_vap_si() available after specifying the equilibrium state conditions	$-f_T^F(T, \rho^V)$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $\rho^W$ are state variables
ice_vap_pressure_vap_si: pressure of water vapour (S21.10)			
ice_vap_pressure_vap_si() available after specifying the equilibrium state conditions	$P$	Pa	$P$ is a state variable
ice_vap_sublimationpressure_si: absolute pressure at specified $T$ (S21.11)			
ice_vap_sublimationpressure_si(t) $t = T / \text{K}$	$P$	Pa	$P$ is a state variable obtained with the specified $T$
ice_vap_sublimationtemp_si: absolute temperature at specified $P$ (S21.12)			
ice_vap_sublimationtemp_si(p) $p = P / \text{Pa}$	$T$	K	$T$ is a state variable obtained with the specified $P$
ice_vap_temperature_si: absolute temperature (S21.13)			
ice_vap_temperature_si() available after specifying the equilibrium state conditions	$T$	K	$T$ is a state variable
ice_vap_volume_subl_si : specific sublimation volume (S21.14)			
ice_vap_volume_subl_si(t) $t = T / \text{K}$	$\Delta v^{\text{subl}} = 1 / \rho^V$ $-g_p^{\text{lh}}(T, P)$	$\frac{\text{m}^3}{\text{kg}}$	$P$ and $\rho^V$ are state variables obtained with the specified $T$
set_ice_vap_eq_at_p: Set ice-vapour equilibrium at given pressure (S21.15)			
set_ice_vap_eq_at_p (p) $p = P / \text{Pa}$	Solve $g^{\text{lh}}(T, P) = g^{\text{v}}(T, P)$ with $P$ specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_ice_vap_eq_at_t: Set ice-vapour equilibrium at given temperature (S21.16)			
set_ice_vap_eq_at_t (t) $t = T / \text{K}$	Solve $g^{\text{lh}}(T, P) = g^{\text{v}}(T, P)$ with $T$ specified and store state variables	-	Returns the ErrorReturn value if unsuccessful

Table S21 (Ice\_Vap\_4), continued

<b>(S21) Ice_Vap_4 Module, cont'd</b>	
Function call	Comments
<b>set_it_ctrl_ice_vap</b> : set parameters used by the iterative solver routine	(S21.17)
<b>set_it_ctrl_ice_vap</b> (key, value)  key: a character string that can take the values IT_STEPS, INIT_VAP_DENS, INIT_TEMP or TOL_VAP_PRESS.  value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of water vapour with ice. Default choices that should generally work are set automatically without calling this routine, but users may prefer one of the alternatives available. The character string <b>key</b> is used to determine which function to execute: <b>IT_STEPS</b> allows maxit to be reset, <b>INIT_VAP_DENS</b> and <b>INIT_TEMP</b> allow the initial estimates of the vapour density and temperature to be reset, <b>TOL_VAP_PRESS</b> allows the convergence criterion to be changed for vapour pressure.

Table S22 (Sea\_Vap\_4): Thermodynamic properties of the seawater-vapour equilibrium. Equilibrium conditions are determined by calling one of `set_sea_ice_eq_at_s_p`, `set_sea_ice_eq_at_s_t` or `set_sea_ice_eq_at_t_p`. The computed state variables  $S_A$ ,  $T$ ,  $P$ ,  $\rho^W$ ,  $\rho^V$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters and the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.5, ax5.5.)

**(S22) Sea\_Vap\_4 Module**

Function call	Mathematical equation	Unit	Comments
<b>sea_vap_boilingtemperature_si: boiling temperature (S22.1)</b>			
<code>sea_vap_boilingtemperature_si(s, p)</code> $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$	$T$ from solving $g^V = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	K	This function call determines the equilibrium state.
<b>sea_vap_brinefraction_seavap_si: mass fraction of brine in sea ice (S22.2)</b>			
<code>sea_vap_brinefraction_seavap_si(ssv, t, p)</code> $\text{ssv} = S_{\text{SV}} / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$w = \frac{S_{\text{SV}}}{S_A(T, P)}$ $S_A$ from solving $g^V = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	1	<b>SSV</b> is the sea-vapour salinity i.e. the mass fraction of salt in seawater + vapour. This function call determines the equilibrium state.
<b>sea_vap_brinesalinity_si: mass fraction of salt in sea-ice brine (S22.3)</b>			
<code>sea_vap_brinesalinity_si(t, p)</code> $t = T / \text{K}$ $p = P / \text{Pa}$	$S_A$ from solving $g^V = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	$\frac{\text{kg}}{\text{kg}}$	This function call determines the equilibrium state.
<b>sea_vap_cp_seavap_si: specific isobaric heat capacity of sea vapour (S22.4)</b>			
<code>sea_vap_cp_seavap_si(ssv, t, p)</code> $\text{ssv} = S_{\text{SV}} / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$c_p = -T g_{TT}^{\text{SV}}$	$\frac{\text{J}}{\text{kg K}}$	$g^{\text{SV}}$ is the Gibbs function of sea vapour (S22.16). This function call determines the equilibrium state.
<b>sea_vap_density_sea_si: density of brine (S22.5)</b>			
<code>sea_vap_density_sea_si()</code> available after specifying the equilibrium state conditions	$\rho^{\text{SW}} = \frac{1}{g_P^{\text{SW}}(S_A, T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{SW}}$ is the Gibbs function of seawater

Table S22 (Sea\_Vap\_4), continued

<b>(S22) Sea_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_vap_density_seavap_si: mean density of seawater + vapour (S22.6)</b>			
sea_vap_density_seavap_si (ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\rho^{SV} = \frac{1}{g_P^{SV}(S_{SV}, T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{SV}$ is the Gibbs function of sea vapour (S22.16). This function call determines the equilibrium state.
<b>sea_vap_density_vap_si: density of vapour (S22.7)</b>			
sea_vap_density_vap_si() available after specifying the equilibrium state conditions	$\rho^V$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^V$ is a state variable
<b>sea_vap_enthalpy_evap_si: specific evaporation enthalpy of seawater (S22.8)</b>			
sea_vap_enthalpy_evap_si() available after specifying the equilibrium state conditions	$\Delta h^{\text{evap}} = T(g_T^{\text{sw}} - S_A g_{S_A T}^{\text{sw}} - f_T^V)$	$\frac{\text{J}}{\text{kg}}$	$f^V$ and $g^{\text{sw}}$ are the Helmholtz/Gibbs functions of vapour and seawater
<b>sea_vap_enthalpy_sea_si: specific enthalpy of brine (S22.9)</b>			
sea_vap_enthalpy_sea_si() available after specifying the equilibrium state conditions	$h^{\text{sw}} = g^{\text{sw}} - T g_T^{\text{sw}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{sw}}$ is the Gibbs function of seawater
<b>sea_vap_enthalpy_seavap_si: specific enthalpy of sea vapour (S22.10)</b>			
sea_vap_enthalpy_seavap_si (ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$h^{SV} = g^{SV} - T g_T^{SV}$	$\frac{\text{J}}{\text{kg}}$	$g^{SV}$ is the Gibbs function of sea vapour. This function call determines the equilibrium state.
<b>sea_vap_enthalpy_vap_si: specific enthalpy of vapour (S22.11)</b>			
sea_vap_enthalpy_vap_si() available after specifying the equilibrium state conditions	$h^V = g^V - T g_T^V$	$\frac{\text{J}}{\text{kg}}$	$g^V$ is the Gibbs function of vapour
<b>sea_vap_entropy_sea_si: specific entropy of brine (S22.12)</b>			
sea_vap_entropy_sea_si() available after specifying the equilibrium state conditions	$\eta^{\text{sw}} = -g_T^{\text{sw}}$	$\frac{\text{J}}{\text{kg K}}$	$g^{\text{sw}}$ is the Gibbs function of seawater

Table S22 (Sea\_Vap\_4), continued

<b>(S22) Sea_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_vap_entropy_seavap_si: specific entropy of seawater + vapour (S22.13)</b>			
sea_vap_entropy_seavap_si (ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\eta^{SV} = -g_T^{SV}(S_{SV}, T, P)$	$\frac{\text{J}}{\text{kg K}}$	$g^{SV}$ is the Gibbs function of sea vapour (S22.16). This function call determines the equilibrium state.
<b>sea_vap_entropy_vap_si: specific entropy of vapour (S22.14)</b>			
sea_vap_entropy_vap_si() available after specifying the equilibrium state conditions	$\eta^V = -f_T^V$	$\frac{\text{J}}{\text{kg K}}$	$f^V$ is the Helmholtz function of vapour
<b>sea_vap_expansion_seavap_si: thermal expansion coefficient of sea vapour (S22.15)</b>			
sea_vap_expansion_seavap_si (ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\alpha = \frac{g_{TP}^{SV}}{g_P^{SV}}$	$\frac{1}{\text{K}}$	$g^{SV}$ is the Gibbs function of sea vapour (S22.16). This function call determines the equilibrium state.
<b>sea_vap_g_si: Gibbs function of sea vapour (i.e. seawater + vapour) (S22.16)</b>			
sea_vap_g_si(l, m, n ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\frac{\partial^{l+m+n}}{\partial S_{SV}^l \partial T^m \partial P^n} g^{SV}(S_{SV}, T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{kg K}^m}$	$l, m, n \geq 0$ $l + m + n \leq 2$
<b>sea_vap_kappa_t_seavap_si: isothermal compressibility of sea vapour (S22.17)</b>			
sea_vap_kappa_t_seavap_si (ssv, t, p) ssv = $S_{SV} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\kappa_T = -\frac{g_{PP}^{SV}}{g_P^{SV}}$	$\frac{1}{\text{Pa}}$	$g^{SV}$ is the Gibbs function of sea vapour. This function call determines the equilibrium state.
<b>sea_vap_pressure_si: pressure (S22.18)</b>			
sea_vap_pressure_si() available after specifying the equilibrium state conditions	$P$	Pa	$P$ is a state variable

Table S22 (Sea\_Vap\_4), continued

<b>(S22) Sea_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_vap_salinity_si: brine salinity</b> (S22.19)			
sea_vap_salinity_si() available after specifying the equilibrium state conditions	$S_A$	$\frac{\text{kg}}{\text{kg}}$	$S_A$ is a state variable
<b>sea_vap_temperature_si: temperature</b> (S22.20)			
sea_vap_temperature_si() available after specifying the equilibrium state conditions	$T$	K	$T$ is a state variable
<b>sea_vap_vapourpressure_si: vapour pressure of seater</b> (S22.21)			
sea_vap_vapourpressure_si (s, t) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$	$P$ from solving $g^v = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$	Pa	This function call determines the equilibrium state.
<b>sea_vap_volume_evap_si: specific evaporation volume of seawater</b> (S22.22)			
sea_vap_volume_evap_si() available after specifying the equilibrium state conditions	$\Delta v^{\text{evap}} = g_P^v - g_P^{\text{sw}}$	$\frac{\text{m}^3}{\text{kg}}$	
<b>set_sea_vap_eq_at_s_p: set the equilibrium state</b> (S22.23)			
set_sea_vap_eq_at_s_p (s, p) $s = S_A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$	solve $g^v = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ and store the result	-	This function call determines the equilibrium state. Returns an error if unsuccessful.
<b>set_sea_vap_eq_at_s_t: set the equilibrium state</b> (S22.24)			
set_sea_vap_eq_at_s_t (s, t) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$	solve $g^v = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ and store the result	-	This function call determines the equilibrium state. Returns an error if unsuccessful.
<b>set_sea_vap_eq_at_t_p: set the equilibrium state</b> (S22.25)			
set_sea_vap_eq_at_t_p (t, p) $t = T / \text{K}$ $p = P / \text{Pa}$	solve $g^v = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ and store the result	-	This function call determines the equilibrium state. Returns an error if unsuccessful.

Table S22 (Sea\_Vap\_4), continued

<b>(S22) Sea_Vap_4 Module, cont'd</b>	
Function call	Comments
<b>set_it_ctrl_sea_vap: set parameters for Newton iteration (S22.26)</b>	
<p><b>set_it_ctrl_sea_vap(key, value)</b></p> <p>key: a character string that can take the values            IT_STEPS,            INIT_LIQ_DENS,            INIT_VAP_DENS,            INIT_BRINE_SA,            INIT_TEMP,            TOL_BRINE_SA,            TOL_TEMP or            TOL_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>	<p>This subroutine allows the user to specify details regarding the iterative solvers used to determine the equilibrium of vapour with seawater. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string <b>key</b> is used to determine which function to execute: <b>IT_STEPS</b> allows maxit to be reset, <b>INIT_LIQ_DENS</b>, <b>INIT_VAP_DENS</b>, <b>INIT_BRINE_SA</b>, <b>INIT_TEMP</b> allows the initial value of the liquid density, vapour density, brine salinity or boiling temperature, respectively, estimate to be reset, <b>TOL_BRINE_SA</b>, <b>TOL_TEMP</b>, <b>TOL_PRESS</b> allows the convergence criterion to be changed for brine salinity, boiling temperature, or vapour pressure, respectively.</p>

Table S23 (Ice\_Liq\_4): Thermodynamic properties of the ice-liquid equilibrium. Equilibrium conditions are determined by calling either **set\_ice\_liq\_eq\_at\_p** or **set\_ice\_liq\_eq\_at\_t**. The computed state variables  $T$ ,  $P$ ,  $g^{\text{lh}}$ ,  $g^{\text{w}}$ ,  $\rho^{\text{lh}}$ ,  $\rho^{\text{w}}$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters so that the appropriate **set\_** command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.2, ax5.2.)

<b>(S23) Ice_Liq_4 Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_liq_chempot_si: chemical potential of liquid water (S23.1)</b>			
<p><b>ice_liq_chempot_si()</b>            available after specifying the equilibrium state conditions</p>	$g^{\text{lh}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{lh}}$ is a state variable



Table S23 (Ice\_Liq\_4), continued

<b>(S23) Ice_Liq_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_liq_density_ice_si: density of ice</b> (S23.2)			
ice_liq_density_ice_si() available after specifying the equilibrium state conditions	$\rho^{\text{lh}}$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^{\text{lh}}$ is a state variable
<b>ice_liq_density_liq_si: density of liquid water</b> (S23.3)			
ice_liq_density_liq_si() available after specifying the equilibrium state conditions	$\rho^{\text{w}}$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^{\text{w}}$ are state variables
<b>ice_liq_enthalpy_ice_si: specific enthalpy of ice</b> (S23.4)			
ice_liq_enthalpy_ice_si() available after specifying the equilibrium state conditions	$h^{\text{lh}} = g^{\text{lh}} - Tg_T^{\text{lh}}(T, P)$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{lh}}, T$ and $P$ are state variables
<b>ice_liq_enthalpy_liq_si: specific enthalpy of liquid water</b> (S23.5)			
ice_liq_enthalpy_liq_si() available after specifying the equilibrium state conditions	$h^{\text{w}} = g^{\text{w}} - Tf_T^{\text{F}}(T, \rho^{\text{w}})$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{w}}, T$ and $\rho^{\text{w}}$ are state variables
<b>ice_liq_enthalpy_melt_si: specific melting enthalpy of water</b> (S23.6)			
ice_liq_enthalpy_melt_si() available after specifying the equilibrium state conditions	$\Delta h^{\text{melt}} = Tg_T^{\text{lh}}(T, P)$ $- Tf_T^{\text{F}}(T, \rho^{\text{w}})$	$\frac{\text{J}}{\text{kg}}$	$T, P$ and $\rho^{\text{w}}$ are state variables
<b>ice_liq_entropy_ice_si: specific entropy of ice</b> (S23.7)			
ice_liq_entropy_ice_si() available after specifying the equilibrium state conditions	$\eta^{\text{lh}} = -g_T^{\text{lh}}(T, P)$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $P$ are state variables
<b>ice_liq_entropy_liq_si: specific entropy of liquid water</b> (S23.8)			
ice_liq_entropy_liq_si() available after specifying the equilibrium state conditions	$\eta^{\text{w}} = -f_T^{\text{F}}(T, \rho^{\text{w}})$	$\frac{\text{J}}{\text{kg K}}$	$T$ and $\rho^{\text{w}}$ are state variables
<b>ice_liq_entropy_melt_si: specific melting entropy of water, <math>\Delta\sigma^{\text{melt}}</math></b> (S23.9)			
ice_liq_entropy_melt_si() available after specifying the equilibrium state conditions	$\Delta\eta^{\text{melt}} = g_T^{\text{lh}}(T, P)$ $- f_T^{\text{F}}(T, \rho^{\text{w}})$	$\frac{\text{J}}{\text{kg K}}$	$T_f$ and $P$ are state variables
<b>ice_liq_meltingpressure_si: equilibrium pressure at specified temperature</b> (S23.10)			
ice_liq_meltingpressure_si(t) $t = T / \text{K}$	$P$	Pa	$P$ is a state variable obtained with the specified $T$

Table S23 (Ice\_Liq\_4), continued

<b>(S23) Ice_Liq_4 Module, cont'd</b>			
Function call	Comments		
<b>ice_liq_meltingtemperature_si: equilibrium temperature at specified pressure (S23.11)</b>			
ice_liq_meltingtemperature_si(p) p = P / Pa	$T$	K	$T$ is a state variables obtained with the specified $P$
<b>ice_liq_pressure_liq_si: pressure at previously defined equilibrium (S23.12)</b>			
ice_liq_pressure_liq_si() available after specifying the equilibrium state conditions	$P$	Pa	$T$ is a state variable
<b>ice_liq_temperature_si: Freezing temperature of liquid water at specified pressure (S23.13)</b>			
ice_liq_temperature_si() available after specifying the equilibrium state conditions	$T$	K	$T$ is a state variable obtained with the specified $P$
<b>ice_liq_volume_melt_si: specific melting volume (S23.14)</b>			
ice_liq_volume_melt_si() available after specifying the equilibrium state conditions	$\Delta v^{\text{melt}} = 1/\rho^{\text{W}}$ $-g_p^{\text{lh}}(T, P)$	$\frac{\text{m}^3}{\text{kg}}$	$T, P$ and $\rho^{\text{W}}$ are state variables
<b>set_ice_liq_eq_at_p: set ice-liquid equilibrium at given pressure (S23.15)</b>			
set_ice_liq_eq_at_p(p) p = P / Pa	Solve $g^{\text{lh}}(T, P) = g^{\text{W}}(T, P)$ with $P$ specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
<b>set_ice_liq_eq_at_t: set ice-liquid equilibrium at given temperature (S23.16)</b>			
set_ice_liq_eq_at_t(t) t = T / Pa	Solve $g^{\text{lh}}(T, P) = g^{\text{W}}(T, P)$ with $T$ specified and store state variables	-	Returns the ErrorReturn value if unsuccessful

Table S23 (Ice\_Liq\_4), continued

<b>(S23) Ice_Liq_4 Module, cont'd</b>	
Function call	Comments
set_it_ctrl_ice_liq: set parameters for Newton iteration (S23.17)	
<p>set_it_ctrl_ice_liq (key, value)</p> <p>key: a character string that can take the values IT_STEPS, INIT_LIQ_DENS, INIT_TEMP or TOL_LIQ_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>	<p>This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of liquid water with ice. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_LIQ_DENS and INIT_TEMP allow the initial estimates of the liquid water density and temperature to be reset, TOL_LIQ_PRESS allows the convergence criterion to be changed for liquid water pressure.</p>

Table S24 (Sea\_Liq\_4): Thermodynamic properties of the osmotic water-seawater equilibrium. Equilibrium conditions are determined by calling `set_sea_liq_eq_at_s_t_p`. The computed state variables  $S_A$ ,  $T$ ,  $P^W$ ,  $P^{SW}$ ,  $\rho^W$ ,  $\rho^{SW}$  are saved when one of these commands is called and may be used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, the function `sea_liq_osmoticpressure_si` is called with input parameters and the `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.6, ax5.6.)

<b>(S24) Sea_Liq_4 Module</b>			
Function call	Mathematical equation	Units	Comments
sea_liq_osmoticpressure_si: specifying the equilibrium state (S24.1)			
<p>sea_liq_osmoticpressure_si (s, t, p)</p> <p><math>s = S_A / (\text{kg kg}^{-1})</math> <math>t = T / \text{K}</math> <math>p = P^W / \text{Pa}</math></p>	$\Delta P = P^{SW} - P^W$ <p>from solving</p> $g^W(T, P^W) =$ $g^{SW}(S_A, T, P^{SW})$ $- S_A g_{S_A}^{SW}(S_A, T, P^{SW})$	-	<p><math>P^W</math> is the pressure on pure water, <math>P^{SW}</math> the pressure on seawater</p>

Table S24 (Sea\_Liq\_4), continued

<b>(S24) Sea_Liq_4 Module (cont'd)</b>			
Function call	Mathematical equation	Units	Comments
<b>set_sea_liq_eq_at_s_t_p: set the equilibrium state</b> (S24.2)			
<b>set_sea_liq_eq_at_s_t_p</b> (s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P^W / \text{Pa}$	Solve $g^W(T, P^W) =$ $g^{SW}(S_A, T, P^{SW})$ $- S_A g_{S_A}^{SW}(S_A, T, P^{SW})$ and store the result	-	This function call determines the equilibrium state. Returns an error if unsuccessful
<b>set_it_ctrl_sea_liq: set parameters for Newton iteration</b> (S24.3)			
<b>set_it_ctrl_sea_liq</b> (key, value)  <b>key:</b> a character string that can take the values IT_STEPS, INIT_LIQ_DENS or TOL_LIQ_PRESS.  <b>value:</b> a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solvers used to determine the equilibrium of humid air with seawater. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_LIQ_DENS allows the initial value of the primary pure-water density estimate to be reset, TOL_LIQ_PRESS allows the convergence criterion to be changed for pressure on pure water.		

Table S25 (Sea\_Ice\_4): Thermodynamic properties of the seawater-ice equilibrium (sea ice). Equilibrium conditions are determined by calling one of `set_sea_ice_eq_at_s_p`, `set_sea_ice_eq_at_s_t` or `set_sea_ice_eq_at_t_p`. The computed state variables  $S_A$ ,  $T$ ,  $P$ ,  $\rho^W$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters so that the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.4, ax5.4.)

**(S25) Sea\_Ice\_4 Module**

Function call	Mathematical equation	Unit	Comments
<b>sea_ice_brinefraction_seaice_si: mass fraction of brine in sea ice (S25.1)</b>			
sea_ice_brinefraction_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$w = \frac{S_{SI}}{S_A(T, P)},$ $S_A \text{ from solving}$ $g^{\text{Ih}} = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	1	SSI is the sea-ice salinity i.e. the mass fraction of salt in sea ice. This function call determines the equilibrium state.
<b>sea_ice_brinesalinity_si: mass fraction of salt in sea-ice brine (S25.2)</b>			
sea_ice_brinesalinity_si (t, p) t = $T / \text{K}$ p = $P / \text{Pa}$	$S_A \text{ from solving}$ $g^{\text{Ih}} = g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	$\frac{\text{kg}}{\text{kg}}$	This function call determines the equilibrium state.
<b>sea_ice_cp_seaice_si: isobaric heat capacity of sea ice (S25.3)</b>			
sea_ice_cp_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$c_P = -T g_{TT}^{\text{SI}}$	$\frac{\text{J}}{\text{kg K}}$	$g^{\text{SI}}$ is the Gibbs function of sea ice (S25.18). This function call determines the equilibrium state.
<b>sea_ice_density_ice_si: density of ice (S25.4)</b>			
sea_ice_density_ice_si() available after specifying the equilibrium state conditions	$\rho^{\text{Ih}} = \frac{1}{g_P^{\text{Ih}}(T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{Ih}}$ is the Gibbs function of ice Ih
<b>sea_ice_density_sea_si: density of brine (S25.5)</b>			
sea_ice_density_sea_si() available after specifying the equilibrium state conditions	$\rho^{\text{SW}} = \frac{1}{g_P^{\text{SW}}(S_A, T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{SW}}$ is the Gibbs function of seawater

Table S25 (Sea\_Ice\_4), continued:

<b>(S25) Sea_Ice_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_ice_density_seaice_si: density of sea ice (S25.6)</b>			
sea_ice_density_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\rho^{SI} = \frac{1}{g_P^{SI}(S_{SI}, T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{SI}$ is the Gibbs function of sea ice (S25.18). This function call determines the equilibrium state.
<b>sea_ice_dtfdp_si: Freezing point lowering due to pressure changes (a Clausius-Clapyron relation) (S25.7)</b>			
sea_ice_dtfdp_si(s, p) s = $S_A / (\text{kg kg}^{-1})$ p = $P / \text{Pa}$	$\left. \frac{\partial t_f}{\partial p} \right _{S_A} = \chi_p(S_A, p)$ $= - \frac{g_p^{SW} - S_A g_{S_A p}^{SW} - g_p^{Ih}}{g_t^{SW} - S_A g_{S_A t}^{SW} - g_t^{Ih}}$ determined after calling set_sea_ice_eq_at_s_p	$\frac{\text{K}}{\text{Pa}}$	This function call determines the equilibrium state
<b>sea_ice_dtfds_si: Freezing point lowering due to salinity changes (Raoult's law) (S25.8)</b>			
sea_ice_dtfds_si() s = $S_A / (\text{kg kg}^{-1})$ p = $P / \text{Pa}$	$\left. \frac{\partial t_f}{\partial S_A} \right _p = \chi_s(S_A, p)$ $= \frac{S_A g_{S_A S_A}^{SW}}{g_t^{SW} - S_A g_{S_A t}^{SW} - g_t^{Ih}}$ determined after calling set_sea_ice_eq_at_s_p	$\frac{\text{K}}{\text{kg kg}^{-1}}$	This function call determines the equilibrium state
<b>sea_ice_enthalpy_ice_si: specific enthalpy of ice (S25.9)</b>			
sea_ice_enthalpy_ice_si() available after specifying the equilibrium state conditions	$h^{Ih} = g^{Ih} - T g_T^{Ih}$	$\frac{\text{J}}{\text{kg}}$	$g^{Ih}$ is the Gibbs function of ice Ih
<b>sea_ice_enthalpy_melt_si: specific melting enthalpy of ice in seawater (S25.10)</b>			
sea_ice_enthalpy_melt_si() available after specifying the equilibrium state conditions	$\Delta h^{\text{melt}} = T(g_T^{Ih} - g_T^{SW} + S_A g_{S_A T}^{SW})$	$\frac{\text{J}}{\text{kg}}$	$g^{Ih}$ and $g^{SW}$ are the Gibbs functions of ice Ih and seawater
<b>sea_ice_enthalpy_sea_si: specific enthalpy of brine (S25.11)</b>			
sea_ice_enthalpy_sea_si() available after specifying the equilibrium state conditions	$h^{SW} = g^{SW} - T g_T^{SW}$	$\frac{\text{J}}{\text{kg}}$	$g^{SW}$ is the Gibbs function of seawater

Table S25 (Sea\_Ice\_4), continued:

<b>(S25) Sea_Ice_4 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_ice_enthalpy_seaice_si: specific enthalpy of sea ice</b> (S25.12)			
sea_ice_enthalpy_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$h^{SI} = g^{SI} - Tg_T^{SI}$	$\frac{\text{J}}{\text{kg}}$	$g^{SI}$ is the Gibbs function of sea ice. This function call determines the equilibrium state.
<b>sea_ice_entropy_ice_si: specific entropy of ice</b> (S25.13)			
sea_ice_entropy_ice_si() available after specifying the equilibrium state conditions	$\eta^{lh} = -g_T^{lh}$	$\frac{\text{J}}{\text{kg K}}$	$g^{lh}$ is the Gibbs function of ice
<b>sea_ice_entropy_sea_si: specific entropy of brine</b> (S25.14)			
sea_ice_entropy_sea_si() available after specifying the equilibrium state conditions	$\eta^{sw} = -g_T^{sw}$	$\frac{\text{J}}{\text{kg K}}$	$g^{sw}$ is the Gibbs function of seawater
<b>sea_ice_entropy_seaice_si: specific entropy of sea ice</b> (S25.15)			
sea_ice_entropy_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\eta^{SI} = -g_T^{SI}(S_{SI}, T, P)$	$\frac{\text{J}}{\text{kg K}}$	$g^{SI}$ is the Gibbs function of sea ice (S25.18). This function call determines the equilibrium state.
<b>sea_ice_expansion_seaice_si: thermal expansion coefficient of sea ice</b> (S25.16)			
sea_ice_expansion_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\alpha = \frac{g_{TP}^{SI}}{g_P^{SI}}$	$\frac{1}{\text{K}}$	$g^{SI}$ is the Gibbs function of sea ice (S25.18). This function call determines the equilibrium state.
<b>sea_ice_freezingtemperature_si: freezing temperature of seawater</b> (S25.17)			
sea_ice_freezingtemperature_si (s, p) t = $S_A / (\text{kg kg}^{-1})$ p = $P / \text{Pa}$	$T$ from solving $g^{lh} = g^{sw} - S_A g_{S_A}^{sw}$	K	This function call determines the equilibrium state.
<b>sea_ice_g_si: Gibbs function of sea ice</b> (S25.18)			
sea_ice_g_si(l, m, n ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\frac{\partial^{l+m+n}}{\partial S_{SI}^l \partial T^m \partial P^n} g^{SI}(S_{SI}, T, P)$	$\frac{\text{J}^{1-n} \text{ m}^{3n}}{\text{kg K}^m}$	$l, m, n \geq 0$ $l + m + n \leq 2$

Table S25 (Sea\_Ice\_4), continued:

<b>(S25) Sea_Ice_4 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_ice_kappa_t_seaice_si: isothermal compressibility of sea ice</b> (S25.19)			
sea_ice_kappa_t_seaice_si (ssi, t, p) ssi = $S_{SI} / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\kappa_T = -\frac{g_{PP}^{SI}}{g_P^{SI}}$	$\frac{1}{\text{Pa}}$	This function call determines the equilibrium state.
<b>sea_ice_meltingpressure_si: melting pressure of ice in seawater</b> (S25.20)			
sea_ice_meltingpressure_si (s, t) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$	$P \text{ from solving } g^{\text{lh}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$	Pa	This function call determines the equilibrium state.
<b>sea_ice_pressure_si: pressure</b> (S25.21)			
sea_ice_pressure_si() available after specifying the equilibrium state conditions	$P$	Pa	$P$ is a state variable
<b>sea_ice_salinity_si: brine salinity</b> (S25.22)			
sea_ice_salinity_si() available after specifying the equilibrium state conditions	$S_A$	$\frac{\text{kg}}{\text{kg}}$	$S_A$ is a state variable
<b>sea_ice_temperature_si: temperature</b> (S25.23)			
sea_ice_temperature_si() available after specifying the equilibrium state conditions	$T$	K	$T$ is a state variable
<b>sea_ice_volume_melt_si: specific melting volume of ice in seawater</b> (S25.24)			
sea_ice_volume_melt_si() available after specifying the equilibrium state conditions	$\Delta v^{\text{melt}} = g_P^{\text{sw}} - g_P^{\text{lh}}$	$\frac{\text{m}^3}{\text{kg}}$	



Table S25 (Sea\_Ice\_4), continued:

<b>(S25) Sea_Ice_4 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>set_it_ctrl_sea_ice: set parameters for Newton iteration (S25.25)</b>			
<p>set_it_ctrl_sea_ice(key, value)</p> <p>key: a character string that can take the values IT_STEPS, INIT_LIQ_DENS, INIT_BRINE_SA, INIT_TEMP, TOL_BRINE_SA, TOL_TEMP or TOL_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>			<p>This subroutine allows the user to specify details regarding the iterative solvers used to determine the equilibrium of ice with seawater. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_LIQ_DENS, INIT_BRINE_SA, INIT_TEMP allows the initial value of the liquid density, brine salinity or freezing temperature, respectively, estimate to be reset, TOL_BRINE_SA, TOL_TEMP, TOL_PRESS allows the convergence criterion to be changed for brine salinity, freezing temperature, or melting pressure, respectively.</p>
<b>set_sea_air_eq_at_s_p: specifying the equilibrium state (S25.26)</b>			
<p>set_sea_air_eq_at_s_p(s, p)</p> <p><math>s = S_A / (\text{kg kg}^{-1})</math> <math>p = P / \text{Pa}</math></p>	<p>solve</p> $g^{\text{lh}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ <p>and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>
<b>set_sea_air_eq_at_s_t: specifying the equilibrium state (S25.27)</b>			
<p>set_sea_air_eq_at_s_t(s, t)</p> <p><math>s = S_A / (\text{kg kg}^{-1})</math> <math>t = T / \text{K}</math></p>	<p>solve</p> $g^{\text{lh}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ <p>and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>
<b>set_sea_air_eq_at_t_p: specifying the equilibrium state (S25.28)</b>			
<p>set_sea_air_eq_at_t_p(t, p)</p> <p><math>t = T / \text{K}</math> <math>p = P / \text{Pa}</math></p>	<p>solve</p> $g^{\text{lh}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}$ <p>and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>

Table S26 (Sea\_Air\_4): Thermodynamic properties of the humid air-seawater equilibrium. Equilibrium conditions are determined by calling either `set_sea_air_eq_at_s_a_p` or `set_sea_air_eq_at_s_t_p`. The computed state variables  $A$ ,  $S_A$ ,  $T$ ,  $P$ ,  $\rho^W$ ,  $\rho^{AV}$  are saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters so that the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.11, ax5.11.)

**(S26) Sea\_Air\_4 Module**

Function call	Mathematical equation	Unit	Comments
<b>sea_air_chempot_evap_si</b> : Normalized chemical potential difference (S26.1)			
<b>sea_air_chempot_evap_si</b> (a, s, t, p) $a = A / (\text{kg kg}^{-1})$ $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\frac{(g^{AV} - Ag_A^{AV}) - (g^{SW} - S_A g_{S_A}^{SW})}{R_w T}$	1	A measure of the deviation from air-sea equilibrium (Onsager force)
<b>sea_air_condense_temp_si</b> : condensation temperature of humid air on seawater (S26.2)			
<b>sea_air_condense_temp_si</b> (s, a, p) $s = S_A / (\text{kg kg}^{-1})$ $a = A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$	$T$ from solving $g^{AV} - Ag_A^{AV} = g^{SW} - S_A g_{S_A}^{SW}$	K	On cooling to $T$ , humid air condenses at the sea surface even before dew can form. This function call determines the equilibrium state.
<b>sea_air_density_air_si</b> : density of humid air (S26.3)			
<b>sea_air_density_air_si()</b> available after specifying the equilibrium state conditions	$\rho^{AV} = \frac{1}{g_P^{AV}}$	$\frac{\text{kg}}{\text{m}^3}$	The state variable $\rho^{AV}$ is the density of humid air
<b>sea_air_density_vap_si</b> : density of vapour in humid air (S26.4)			
<b>sea_air_density_vap_si()</b> available after specifying the equilibrium state conditions	$\rho^V = \frac{1 - A}{g_P^{AV}}$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^V$ is the density of vapour in humid air
<b>sea_air_enthalpy_evap_si</b> : specific evaporation enthalpy of seawater (S26.5)			
<b>sea_air_enthalpy_evap_si()</b> available after specifying the equilibrium state conditions	$L^{\text{evap}} = h^{AV} - Ah_A^{AV} - h^{SW} + S_A h_{S_A}^{SW}$	$\frac{\text{J}}{\text{kg}}$	$L^{\text{evap}}$ is the isobaric latent heat of seawater

Table S26 (Sea\_Air\_4), continued:

<b>(S26) Sea_Air_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_air_entropy_air_si: specific entropy of humid air (S26.6)</b>			
sea_air_entropy_air_si (s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\eta^{\text{AV}} = -g_T^{\text{AV}}$	$\frac{\text{J}}{\text{kg K}}$	This function call determines the equilibrium state.
<b>sea_air_massfraction_air_si: mass fraction of dry air in humid air (S26.7)</b>			
sea_air_massfraction_air_si (s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$A \text{ from solving}$ $g^{\text{AV}} - A g_A^{\text{AV}}$ $= g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$	$\frac{\text{kg}}{\text{kg}}$	This function call determines the equilibrium state.
<b>sea_air_vapourpressure_si: partial pressure of vapour in humid air at equilibrium (S26.8)</b>			
sea_air_vapourpressure_si (s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$P x_V^{\text{AV}}$ $x_V^{\text{AV}} \text{ determined from } A^{\text{sat}}$ $\text{by calling}$ $\text{air_molfraction_vap_si}$	Pa	This function call determines the equilibrium state by calling set_sea_air_eq_at_s_t_p.
<b>set_sea_air_eq_at_s_a_p: set the equilibrium state (S26.9)</b>			
set_sea_air_eq_at_s_a_p (s, a, p) $s = S_A / (\text{kg kg}^{-1})$ $a = A / (\text{kg kg}^{-1})$ $p = P / \text{Pa}$	$\text{solve}$ $g^{\text{AV}} - A g_A^{\text{AV}}$ $= g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$ $\text{and store the result}$	-	This function call determines the equilibrium state. Returns an error if unsuccessful
<b>set_sea_air_eq_at_s_t_p: set the equilibrium state (S26.10)</b>			
set_sea_air_eq_at_s_t_p (s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$\text{solve}$ $g^{\text{AV}} - A g_A^{\text{AV}}$ $= g^{\text{SW}} - S_A g_{S_A}^{\text{SW}}$ $\text{and store the result}$	-	This function call determines the equilibrium state. Returns an error if unsuccessful

Table S26 (Sea\_Air\_4), continued:

<b>(S26) Sea_Air_4 Module, cont'd</b>	
Function call	Comments
set_it_ctrl_sea_air: set parameters for Newton iteration (S26.11)	
<p>set_it_ctrl_sea_air(key, value)</p> <p>key: a character string that can take the values IT_STEPS, INIT_AIR, INIT_TEMP, INIT_PRESS, TOL_VAP_PRESS, TOL_TEMP or TOL_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>	<p>This subroutine allows the user to specify details regarding the iterative solvers used to determine the equilibrium of humid air with seawater. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_AIR, INIT_TEMP, INIT_PRESS allows the initial value of the primary mass fraction of dry air in humid air, temperature or pressure, respectively, estimate to be reset, TOL_VAP_PRESS, TOL_TEMP, TOL_PRESS allows the convergence criterion to be changed for vapour pressure, temperature, or pressure, respectively.</p>

Table S27 (Liq\_Ice\_Air\_4): Thermodynamic properties of the equilibrium between liquid water, ice and humid air. Equilibrium conditions are determined by calling one of the subroutines in the two groups discussed below. Each of these routines sets the variables  $T$ ,  $P$ ,  $A$ ,  $\rho^W$  (liquid water density),  $\rho^{AV}$  (humid-air density),  $w^A$  (dry-air fraction of total mass),  $w$  (liquid fraction of the condensed part, including liquid plus solid),  $\eta$  (total entropy) and these are saved for subsequent use in parameter-free function calls. The defined state variables remain available until different conditions are imposed.

For the routines in group (i) (`set_liq_ice_air_eq_at_a`, `set_liq_ice_air_eq_at_t` and `set_liq_ice_air_eq_at_p`), one of the state variables  $A$ ,  $T$  or  $P$  is specified and equality of the chemical potentials then allows complete determination of the other two of  $A$ ,  $T$  and  $P$  and the potential functions. However, in these cases,  $w^A$ ,  $w$  and  $\eta$  are not determined and hence are set to the ErrorReturn value. Note, for example, that by supplying heat to the system, it is possible to melt some ice, thus increasing  $w$  and  $\eta$  without changing  $A$ ,  $T$  or  $P$  will. Similarly, the amount of humid air may be altered at constant  $A$ ,  $T$ ,  $P$  by suitable readjustment of the mass ratios between the three water phases.

To determine these mass ratios, two additional parameters must be specified; group (ii) provides two examples, (`set_liq_ice_air_eq_at_wa_eta_wt` and `set_liq_ice_air_eq_at_wa_wl_wi`) of how this may be done.

For convenience, selected frequently used functions are called with input parameters so that the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.10, ax5.10.)

### (S27) Liq\_Ice\_Air\_4 Module

Function call	Mathematical equation	Unit	Comments
<b>liq_ice_air_airfraction_si: the mass fraction a of dry air in humid air (S27.1)</b>			
<code>liq_ice_air_airfraction_si()</code> Available after equilibrium is set using a group (i) or group (ii) routine	$A$	$\frac{\text{kg}}{\text{kg}}$	State variable for groups (i) and (ii)
<b>liq_ice_air_density_si: density of wet ice air (S27.2)</b>			
<code>liq_ice_air_density_si()</code> Available after equilibrium is set using a group (ii) routine	$\rho = (w^A / A) \rho^{AV} + w^W \rho^W + w^{lh} \rho^{lh}$ $\rho^{lh} = (g_P^{lh})^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$T$ , $P$ , $A$ , $\rho^{AV}$ and $\rho^W$ are state variables of groups (i) and (ii). $w^A$ is a state variable of group (ii). $w^W$ is the liquid fraction (S27.8). $w^{lh}$ is the solid fraction (S27.10). $g^{lh}$ is the Gibbs potential for hexagonal ice I defined at level 1. If a group (i) routine is used then additional constraints must be imposed to determine $w^A$ and $w$ and the mean density.

Table S27 (Liq\_Ice\_Air\_4), continued

<b>(S27) Liq_Ice_Air_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_ice_air_dryairfraction_si</b> : the fraction $a$ of dry air in wet ice air (S27.3)			
liq_ice_air_dryairfraction_si() Available after equilibrium set using group (ii) routine.	$w^A$	$\frac{\text{kg}}{\text{kg}}$	State variable set by group (ii). See final comment under (S27.2).
<b>liq_ice_air_enthalpy_si</b> : specific enthalpy of wet ice air (S27.4)			
liq_ice_air_enthalpy_si() Available after equilibrium set using group (ii) routine.	$h = (w^A / A)h^{AV} + w^W h^W + w^{lh} h^{lh}$	$\frac{\text{J}}{\text{kg}}$	$A$ is a state variable set by groups (i) and (ii). $w^A$ : a state variable of group (ii). $w^W$ : liquid fraction (S27.8) $w^{lh}$ : solid fraction (S27.10) $h^{AV}$ : enthalpy of moist air (S10.4) $h^W$ : enthalpy of liquid water (S7.3) $h^{lh}$ : enthalpy of ice (S8.4) See final comment under (S27.2).
<b>liq_ice_air_entropy_si</b> : specific entropy of wet ice air (S27.5)			
liq_ice_air_entropy_si() Available after equilibrium set using group (ii) routine.	$\eta$	$\frac{\text{J}}{\text{kg K}}$	State variable set by group (ii). See final comment under (S27.2).
<b>liq_ice_air_ifl_si</b> : isentropic freezing level (IFL) from the dry-air fraction, $wa\_si$ and the entropy, $\eta$ of wet air (S27.6)			
liq_ice_air_ifl_si( $wa, \eta$ ) $wa = w^A / (\text{kg kg}^{-1})$ $\eta = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ Specifies group (ii).	$P$ determined with $w = 1$	Pa	State variable set by groups (i) and (ii). See final comment under (S27.2).
<b>liq_ice_air_uml_si</b> : isentropic melting level (IML) from the dry-air fraction, $wa$ and the entropy, $\eta$ of ice air (S27.7)			
liq_ice_air_uml_si( $wa, \eta$ ) $wa = w^A / (\text{kg kg}^{-1})$ $\eta = \eta / (\text{J kg}^{-1} \text{K}^{-1})$ Specifies group (ii).	$P$ determined with $w = 0$	Pa	State variable set by groups (i) and (ii).

Table S27 (Liq\_Ice\_Air\_4), continued

<b>(S27) Liq_Ice_Air_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_ice_air_liquidfraction_si</b> : the fraction of liquid water in wet ice air when water + ice are in equilibrium with humid air (S27.8)			
liq_ice_air_liquidfraction_si() Available after equilibrium set using group (ii) routine.	$w^w = w \left( 1 - \frac{w^A}{A} \right)$	$\frac{\text{kg}}{\text{kg}}$	A is a state variable set by groups (i) and (ii). w and w <sup>A</sup> are state variables set by group (ii). See final comment under (S27.2).
<b>liq_ice_air_pressure_si</b> : pressure of water + ice in equilibrium with humid air (S27.9)			
liq_ice_air_pressure_si() Available after equilibrium set using a group (i) or group (ii) routine.	P	Pa	State variable set by groups (i) and (ii).
<b>liq_ice_air_solidfraction_si</b> : fraction of ice in wet ice air (S27.10)			
liq_ice_air_solidfraction_si() Available after equilibrium set using group (ii) routine.	$w^{\text{th}} = (1 - w) \left( 1 - \frac{w^A}{A} \right)$	$\frac{\text{kg}}{\text{kg}}$	A is a state variable set by groups (i) and (ii). w and w <sup>A</sup> are state variables set by group (ii). See final comment under (S27.2).
<b>liq_ice_air_temperature_si</b> : temperature (S27.11)			
liq_ice_air_temperature_si() Available after equilibrium set using a group (i) or group (ii) routine.	T	K	State variable set by groups (i) and (ii).
<b>liq_ice_air_vapourfraction_si</b> : fraction of vapour in wet ice (S27.12)			
liq_ice_air_vapourfraction_si() Available after equilibrium set using group (ii) routine.	$w^v = w^A \left( \frac{1}{A} - 1 \right)$	$\frac{\text{kg}}{\text{kg}}$	A is a state variable set by groups (i) and (ii). w <sup>A</sup> is a state variable set by group (ii). See final comment under (S27.2).

Table S27 (Liq\_Ice\_Air\_4), continued

<b>(S27) Liq_Ice_Air_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
set_liq_ice_air_eq_at_a: Set module to determine liquid-ice-air equilibrium at given mass fraction of dry air in humid air (S27.13)			
set_liq_ice_air_eq_at_a(a) a = A / (kg kg <sup>-1</sup> )  Specifies group (i).	Solve $g^{AV} - A \left( \frac{\partial g^{AV}}{\partial A} \right)_{T,P} = g^W = g^{lh}$ with the specified value of A and store the results.	-	Returns the ErrorReturn value if unsuccessful. Relative mass fractions of 3 phases are undetermined.
set_liq_ice_air_eq_at_p: Set module to determine liquid-ice-air equilibrium at given pressure (S27.14)			
set_liq_ice_air_eq_at_p(p) p = P / Pa  Specifies group (i).	Solve $g^{AV} - A \left( \frac{\partial g^{AV}}{\partial A} \right)_{T,P} = g^W = g^{lh}$ with the specified value of P and store the results.	-	Returns the ErrorReturn value if unsuccessful. Relative mass fractions of 3 phases are undetermined.
set_ice_air_eq_at_t: Set module to determine liquid-ice-air equilibrium at given temperature (S27.15)			
set_liq_ice_air_eq_at_t(t) t = T / K  Specifies group (i).	Solve $g^{AV} - A \left( \frac{\partial g^{AV}}{\partial A} \right)_{T,P} = g^W = g^{lh}$ with the specified value of T and store the results.	-	Returns the ErrorReturn value if unsuccessful. Relative mass fractions of the 3 phases are left undetermined.



Table S27 (Liq\_Ice\_Air\_4), continued

<b>(S27) Liq_Ice_Air_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>set_liq_ice_air_eq_at_wa_wl_wi:</b> Set module to determine liquid-ice-air equilibrium at given $w^A$ , $w^W$ and $w^{lh}$ (S27.16)			
<b>set_liq_ice_air_equilibrium_at_wa_wl_wi</b> (wa, wl, wi) wa = $w^A / (\text{kg kg}^{-1})$ wl = $w^W / (\text{kg kg}^{-1})$ wi = $w^{lh} / (\text{kg kg}^{-1})$  Specifies group (ii).	Solve $g^{AV} - A \left( \frac{\partial g^{AV}}{\partial A} \right)_{T,P} = g^W = g^{lh}$ and $A = \frac{w^A}{1 - w^W - w^{lh}}$ with the specified values of $w^A$ , $w^W$ and $w^{lh}$ , and store the results.	-	Returns the ErrorReturn value if unsuccessful. Relative mass fractions of the 3 phases are specified.
<b>set_liq_ice_air_equilibrium_at_wa_eta_wt:</b> Set module to liquid-ice-air equilibrium at given $w^A$ , $\eta$ and $w$ (S27.17)			
<b>set_liq_ice_air_equilibrium_at_wa_eta_wt</b> (wa, eta, w) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ w = $w^W / (w^W + w^{lh})$  Specifies group (ii).	Solve $g^{AV} - A \left( \frac{\partial g^{AV}}{\partial A} \right)_{T,P} = g^W = g^{lh}$ and $\eta = w^W \eta^W + w^{lh} \eta^{lh} + w^{AV} \eta^{AV}$ with the specified values of $w^A$ , $\eta$ and $w = w^W / (w^W + w^{lh})$ , and store the results.	-	Returns the ErrorReturn value if unsuccessful. Specification of $w^A$ , $\eta$ and $w$ allows the relative mass fractions of the 3 phases to be determined.
<b>set_it_ctrl_liq_ice_air:</b> set parameters used in the iterative solver routine (S27.18)			
<b>set_it_ctrl_liq_ice_air</b> (key, value)  key: a character string that can take the values IT_STEPS, INIT_AIR, INIT_LIQ, INIT_HUM, INIT_TEMP, INIT_PRESS, TOL_TEMP or TOL_PRESS.  value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of humid air with liquid water and ice. Default choices that should generally work are initialized automatically without calling this function, but users may prefer an alternative so a selection of options is provided. The character string <b>key</b> is used to determine which function to execute: <b>IT_STEPS</b> allows maxit to be reset, <b>INIT_AIR</b> and <b>INIT_LIQ</b> allow the initial values of the densities of air and liquid water to be reset, <b>INIT_HUM</b> allows the dry-air fraction of the humid air to be reset, <b>INIT_PRESS</b> and <b>INIT_TEMP</b> allow the initial estimates of the pressure and temperature to be reset, and <b>TOL_PRESS</b> and <b>TOL_TEMP</b> allow the convergence criteria to be changed for the total pressure and the temperature.		

Table S28 (Sea\_Ice\_Vap\_4): Thermodynamic properties of the seawater-ice-vapour equilibrium (“triple point” of seawater). Equilibrium conditions are determined by calling one of `set_sea_ice_eq_at_p`, `set_sea_ice_eq_at_s` or `set_sea_ice_eq_at_t`. The computed state variables  $S_A$ ,  $T$ ,  $P$ ,  $\rho^W$ ,  $\rho^V$  are determined and saved when one of these commands is called and are then used for subsequent parameter-free function calls. These state variables remain available until different conditions are imposed. For convenience, selected frequently used functions are called with input parameters and the appropriate `set_` command is called internally to compute the equilibrium before determining the requested result. The state variables are also saved for subsequent use in this case. (See Part I, sections 5.7, ax5.7.)

**(S28) Sea\_Ice\_Vap\_4 Module**

Function call	Mathematical equation	Unit	Comments
<b>sea_ice_vap_density_vap_si: vapour density (S28.1)</b>			
sea_ice_vap_density_vap_si() available after specifying the equilibrium state conditions	$\rho^V$	$\frac{\text{kg}}{\text{m}^3}$	$\rho^V$ is a state variable
<b>sea_ice_vap_pressure_si: pressure (S28.2)</b>			
sea_ice_vap_pressure_si() available after specifying the equilibrium state conditions	$P$	Pa	$P$ is a state variable
<b>sea_ice_vap_salinity_si: brine salinity (S28.3)</b>			
sea_ice_vap_salinity_si() available after specifying the equilibrium state conditions	$S_A$	$\frac{\text{kg}}{\text{kg}}$	$S_A$ is a state variable
<b>sea_ice_vap_temperature_si: temperature (S28.4)</b>			
sea_ice_vap_temperature_si() available after specifying the equilibrium state conditions	$T$	K	$T$ is a state variable

Table S28 (Sea\_Ice\_Vap\_4), continued:

<b>(S28) Sea_Ice_Vap_4 Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>set_it_ctrl_sea_ice_vap: set parameters for Newton iteration (S28.5)</b>			
<p><b>set_it_ctrl_sea_ice_vap (key, value)</b></p> <p>key: a character string that can take the values IT_STEPS, INIT_LIQ_DENS, INIT_VAP_DENS, INIT_BRINE_SA, INIT_TEMP, INIT_PRESS, TOL_TEMP or TOL_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>			<p>This subroutine allows the user to specify details regarding the iterative solvers used to determine the equilibrium of humid air with seawater. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string <b>key</b> is used to determine which function to execute: <b>IT_STEPS</b> allows maxit to be reset, <b>INIT_LIQ_DENS</b>, <b>INIT_VAP_DENS</b>, <b>INIT_BRINE_SA</b>, <b>INIT_TEMP</b> allows the initial value of the liquid density, vapour density, brine salinity or temperature, respectively, estimate to be reset, <b>TOL_TEMP</b>, <b>TOL_PRESS</b> allows the convergence criterion to be changed for temperature or pressure, respectively.</p>
<b>set_sea_ice_vap_eq_at_p: set the equilibrium state (S28.6)</b>			
<p><b>set_sea_ice_vap_eq_at_p (p)</b> <math>p = P / \text{Pa}</math></p>	<p>Solve <math>g^{\text{lh}} = g^{\text{v}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}</math> and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>
<b>set_sea_ice_vap_eq_at_s: set the equilibrium state (S28.7)</b>			
<p><b>set_sea_ice_vap_eq_at_s(s)</b> <math>s = S_A / (\text{kg kg}^{-1})</math></p>	<p>Solve <math>g^{\text{lh}} = g^{\text{v}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}</math> and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>
<b>set_sea_ice_vap_eq_at_t: set the equilibrium state (S28.8)</b>			
<p><b>set_sea_ice_vap_eq_at_t (t)</b> <math>t = T / \text{K}</math></p>	<p>Solve <math>g^{\text{lh}} = g^{\text{v}} = g^{\text{sw}} - S_A g_{S_A}^{\text{sw}}</math> and store the result</p>	-	<p>This function call determines the equilibrium state. Returns an error if unsuccessful.</p>

Table S29 (Liq\_Air\_4a): Thermodynamic properties of the liq-air equilibrium. This table deals only with the phase equilibrium properties of pure liquid water with humid air, commonly regarded as "saturated air" or "humidity 100%". The air properties computed here refer to saturated air above the frost point. (See Part I, sections 5.8, ax5.8.)

<b>(S29) Liq_Air_4a Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_air_a_from_rh_cct_si</b> : mass fraction of dry air in humid air from relative humidity, temperature and pressure (S29.2)			
<b>liq_air_a_from_rh_cct_si</b> (rh, t, p) rh = RH / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$A = \frac{1 - RH_{CCT} \times x_V^{AV}(A^{sat}(T, P))}{1 - RH_{CCT} \times x_V^{AV}(A^{sat}(T, P)) \times (1 - M_W / M_A)}$ $RH_{CCT}(A, T, P) = \frac{x_V^{AV}(A)}{x_V^{AV}(A^{sat}(T, P))}$	$\frac{\text{kg}}{\text{kg}}$	$x_V^{AV}$ is the mole fraction of vapour in humid air (level 0). $M_W$ and $M_A$ are the molar masses of water and air.
<b>liq_air_a_from_rh_wmo_si</b> : mass fraction of dry air in humid air from relative humidity, temperature and pressure (S29.1)			
<b>liq_air_a_from_rh_wmo_si</b> (rh, t, p) rh = RH / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$A = \frac{1}{1 + RH_{WMO} \times (1/A^{sat}(T, P) - 1)}$ $RH_{WMO}(A, T, P) = \frac{1/A - 1}{1/A^{sat}(T, P) - 1}$	$\frac{\text{kg}}{\text{kg}}$	$A^{sat}$ is the value of A at equilibrium (S29.2)
<b>liq_air_condensationpressure_si</b> : condensation pressure of humid air (S29.3)			
<b>liq_air_condensationpressure_si</b> (a, t) a = A / (kg kg <sup>-1</sup> ) t = T / K	$P$	Pa	State variable at specified A and T
<b>liq_air_density_air_si</b> : humid air density of saturated air (S29.4)			
<b>liq_air_density_air_si</b> () available after specifying the equilibrium state	$\rho^{AV}$	$\frac{\text{kg}}{\text{m}^3}$	State variable
<b>liq_air_density_liq_si</b> : liquid water density of wet air (S29.5)			
<b>liq_air_density_liq_si</b> () available after specifying the equilibrium state	$\rho^W$	$\frac{\text{kg}}{\text{m}^3}$	State variable
<b>liq_air_density_vap_si</b> : vapour density of saturated air (S29.6)			
<b>liq_air_density_vap_si</b> () available after specifying the equilibrium state	$\rho^V = (1 - A)\rho^{AV}$	$\frac{\text{kg}}{\text{m}^3}$	A and $\rho^{AV}$ are state variables

Table S29 (Liq\_Air\_4a), continued

<b>(S29) Liq_Air_4a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_air_dewpoint_si</b> : dewpoint temperature of humid air (S29.7)			
liq_air_dewpoint_si(a, p) a = A / (kg kg <sup>-1</sup> ) p = P / Pa	$T$	K	State variable at specified A and P of unsaturated air
<b>liq_air_enthalpy_evap_si</b> : specific evaporation heat of saturated air above the frost point (S29.8)			
liq_air_enthalpy_evap_si() available after specifying the equilibrium state	$\Delta h^{\text{evap}} = T(f_T^{\text{W}} + \eta^{\text{AV}} + Ag_{AT}^{\text{AV}})$	$\frac{\text{J}}{\text{kg}}$	A, T, P, $\eta^{\text{AV}}$ , $\rho^{\text{W}}$ and $\rho^{\text{AV}}$ are state variables
<b>liq_air_entropy_air_si</b> : specific entropy of saturated humid air (S29.9)			
liq_air_entropy_air_si() available after specifying the equilibrium state	$\eta^{\text{AV}}$	$\frac{\text{J}}{\text{kg K}}$	$\eta^{\text{AV}}$ is a state variable
<b>liq_air_icl_si</b> : isentropic condensation level ICL of humid air (S29.10)			
liq_air_icl_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$P$	Pa	State variable obtained with fixed A and $\eta$ from A, T, P of unsaturated air
<b>liq_air_ict_si</b> : isentropic condensation temperature ICT of humid air (S29.11)			
liq_air_ict_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$T$	K	State variable obtained with fixed A and $\eta$ from A, T, P of unsaturated air
<b>liq_air_massfraction_air_si</b> : air mass fraction of saturated air (S29.12)			
liq_air_massfraction_air_si(t, p) t = T / K p = P / Pa	$A$	$\frac{\text{kg}}{\text{kg}}$	State variable at specified T and P
<b>liq_air_pressure_si</b> : pressure of saturated air (S29.13)			
liq_air_pressure_si() available after specifying the equilibrium state	$P$	Pa	State variable

Table S29 (Liq\_Air\_4a), continued

<b>(S29) Liq_Air_4a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
liq_air_rh_cct_from_a_si: relative humidity from the mass fraction of dry air in humid air, temperature and pressure (S29.14)			
liq_air_rh_cct_from_a_si (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$RH_{\text{CCT}}(A, T, P) = \frac{x_v^{\text{AV}}(A)}{x_v^{\text{AV}}(A^{\text{sat}}(T, P))}$	1	$x_v^{\text{AV}}$ is the mole fraction of vapour in humid air (defined at level 0). $A^{\text{sat}}$ is the value of A at equilibrium for given T, P (S29.2)
liq_air_rh_wmo_from_a_si: relative humidity from the mass fraction of dry air in humid air, temperature and pressure (S29.15)			
liq_air_rh_wmo_from_a_si (a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$RH_{\text{WMO}}(A, T, P) = \frac{1/A - 1}{1/A^{\text{sat}}(T, P) - 1}$	1	$A^{\text{sat}}$ is the value of A at equilibrium (S29.2)
liq_air_temperature_si: temperature of saturated air (S29.16)			
liq_air_temperature_si() available after specifying the equilibrium state	T	K	State variable
set_it_ctrl_liq_air: set parameters used in the iterative solver routine (S29.17)			
set_it_ctrl_liq_air (key, value)  key: a character string that can take the values IT_STEPS, INIT_HUM, INIT_TEMP, INIT_PRESS, TOL_VAP_PRESS, TOL_TEMP or TOL_PRESS.  value: a real*8 variable used to specify a control parameter associated with the choice specified by key	This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of ice Ih with humid air. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string key is used to determine which function to execute: IT_STEPS allows maxit to be reset, INIT_HUM, INIT_TEMP and INIT_PRESS allow the initial estimates of the mass fraction of dry air in humid air, temperature and pressure to be reset, TOL_VAP_PRESS, TOL_TEMP and TOL_PRESS allow the convergence criteria to be changed for vapour pressure, temperature and total pressure.		

Table S29 (Liq\_Air\_4a), continued

<b>(S29) Liq_Air_4a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
set_liq_air_eq_at_a_eta: sets the saturation equilibrium state between humid air and liquid water (S29.18)			
set_liq_air_eq_at_a_eta (a, eta) a = A / (kg kg <sup>-1</sup> ) eta = η / (J kg <sup>-1</sup> K <sup>-1</sup> )	Solve $g^{AV} - Ag_A^{AV} = g^W$ with A and η specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_liq_air_eq_at_a_p: sets the saturation equilibrium state between humid air and liquid water (S29.19)			
set_liq_air_eq_at_a_p (a, p) a = A / (kg kg <sup>-1</sup> ) p = P / Pa	Solve $g^{AV} - Ag_A^{AV} = g^W$ with A and P specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_liq_air_eq_at_a_t: sets the saturation equilibrium state between humid air and liquid water (S29.20)			
set_liq_air_eq_at_a_t(a, t) a = A / (kg kg <sup>-1</sup> ) t = T / K	Solve $g^{AV} - Ag_A^{AV} = g^W$ with A and T specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_liq_air_eq_at_t_p: sets the saturation equilibrium state between humid air and liquid water (S29.21)			
set_liq_air_equilibrium_to_t_p (t, p) t = T / K p = P / Pa	Solve $g^{AV} - Ag_A^{AV} = g^W$ with T and P specified and store state variables	-	Returns the ErrorReturn value if unsuccessful

Table S30 (Ice\_Air\_4a): Thermodynamic properties of the ice-air equilibrium. This table deals only with the phase equilibrium properties of ice Ih with humid air, commonly regarded as "saturated air" or "humidity 100%". The air properties computed here refer to saturated air below the frost point. (See Part I, sections 5.9, ax5.9.)

<b>(S30) Ice_Air_4a Module</b>			
Function call	Mathematical equation	Unit	Comments
ice_air_a_from_rh_cct_si: mass fraction of dry air in humid air for specified relative humidity, temperature and pressure (S30.2)			
ice_air_a_from_rh_cct_si(rh, t, p) rh = RH / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$A = \frac{1 - RH_{\text{CCT}} \times x_{\text{V}}^{\text{AV}}(A^{\text{sat}}(T, P))}{1 - RH_{\text{CCT}} \times x_{\text{V}}^{\text{AV}}(A^{\text{sat}}(T, P)) \times (1 - M_{\text{W}} / M_{\text{A}})}$ $RH_{\text{CCT}}(A, T, P) = \frac{x_{\text{V}}^{\text{AV}}(A)}{x_{\text{V}}^{\text{AV}}(A^{\text{sat}}(T, P))}$	$\frac{\text{kg}}{\text{kg}}$	$x_{\text{V}}^{\text{AV}}$ is the mole fraction of vapour in humid air. $M_{\text{W}}$ and $M_{\text{A}}$ are the molar masses of water and air.
ice_air_a_from_rh_wmo_si: mass fraction of dry air in humid air for specified relative humidity, temperature and pressure (S30.1)			
ice_air_a_from_rh_wmo_si(rh, t, p) rh = RH / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$A = \frac{1}{1 + RH_{\text{WMO}} \times (1/A^{\text{sat}}(T, P) - 1)}$ $RH_{\text{WMO}}(A, T, P) = \frac{1/A - 1}{1/A^{\text{sat}}(T, P) - 1}$	$\frac{\text{kg}}{\text{kg}}$	$A^{\text{sat}}$ is the value of $A$ at equilibrium (S30.3)
ice_air_condensationpressure_si: condensation pressure of humid air (S30.3)			
ice_air_condensationpressure_si(a, t) a = A / (kg kg <sup>-1</sup> ) t = T / K	$P$	Pa	State variable at specified $A$ and $T$
ice_air_density_air_si: humid air density of saturated air (S30.4)			
ice_air_density_air_si() available after specifying the equilibrium state	$\rho^{\text{AV}}$	$\frac{\text{kg}}{\text{m}^3}$	State variable
ice_air_density_ice_si: ice density of saturated air (S30.5)			
ice_air_density_ice_si() available after specifying the equilibrium state	$\rho^{\text{Ih}} = \frac{1}{g_p^{\text{Ih}}(A, T, P)}$	$\frac{\text{kg}}{\text{m}^3}$	Uses (S8.3) with $A$ , $T$ , $P$ from equilibrium state
ice_air_density_vap_si: vapour density of saturated air (S30.6)			
ice_air_density_vap_si() available after specifying the equilibrium state	$\rho^{\text{V}} = (1 - A)\rho^{\text{AV}}$	$\frac{\text{kg}}{\text{m}^3}$	$A$ and $\rho^{\text{AV}}$ are state variable
ice_air_enthalpy_subl_si: specific sublimation heat of saturated air below the frost point (S30.7)			
ice_air_enthalpy_subl_si() available after specifying the equilibrium state	$\Delta h^{\text{subl}} = T g_T^{\text{Ih}}(T, P) - T f_T^{\text{F}}(T, \rho^{\text{V}})$	$\frac{\text{J}}{\text{kg}}$	$T$ and $P$ are state variables $\rho^{\text{V}}$ from (S30.6)



Table S30 (Ice\_Air\_4a): continued

<b>(S30) Ice_Air_4a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_air_frostpoint_si: frostpoint temperature of humid air (S30.8)</b>			
ice_air_frostpoint_si(a, p) a = A / (kg kg <sup>-1</sup> ) p = P / Pa	$T$	K	State variable at specified A and P
<b>ice_air_icl_si: isentropic condensation level ICL of humid air (S30.9)</b>			
ice_air_icl_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$P$	Pa	State variable obtained with fixed A and $\eta$
<b>ice_air_ict_si: isentropic condensation temperature ICT of humid air (S30.10)</b>			
ice_air_ict_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$T$	K	State variable obtained with fixed A and $\eta$
<b>ice_air_massfraction_air_si: air mass fraction of saturated air (S30.11)</b>			
ice_air_massfraction_air_si(t, p) t = T / K p = P / Pa	$A$	$\frac{\text{kg}}{\text{kg}}$	State variable at specified T and P
<b>ice_air_pressure_si: pressure of saturated air (S30.12)</b>			
ice_air_pressure_si() available after specifying the equilibrium state	$P$	Pa	State variable
<b>ice_air_rh_cct_from_a_si: relative humidity from the mass fraction of dry air in humid air, temperature and pressure (S30.13)</b>			
ice_air_rh_cct_from_a_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$RH_{\text{CCT}}(A, T, P) = \frac{x_v^{\text{AV}}(A)}{x_v^{\text{AV}}(A^{\text{sat}}(T, P))}$	1	$x_v^{\text{AV}}$ is the mole fraction of vapour in humid air (level 0). $A^{\text{sat}}$ is the value of A at equilibrium for given T, P (S29.12)
<b>ice_air_rh_wmo_from_a_si: relative humidity from the mass fraction of dry air in humid air, temperature and pressure (S30.14)</b>			
ice_air_rh_wmo_from_a_si(a, t, p) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa	$RH_{\text{WMO}}(A, T, P) = \frac{1/A - 1}{1/A^{\text{sat}}(T, P) - 1}$	1	$A^{\text{sat}}$ is the value of A at equilibrium (S29.12)

Table S30 (Ice\_Air\_4a), continued

<b>(S30) Ice_Air_4a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
ice_air_sublimationpressure_si: Sublimation pressure of vapour in moist air at equilibrium with ice (S30.15)			
ice_air_sublimationpressure_si(t, p)	$P x_v^{AV}$ $x_v^{AV}$ determined from $A^{sat}$ by calling air_molfraction_vap_si	Pa	This function call sets the equilibrium by calling set_ice_air_eq_at_t_p
ice_air_temperature_si: temperature of saturated air (S30.16)			
ice_air_temperature_si() available after specifying the equilibrium state	$T$	K	State variable
set_ice_air_eq_at_a_eta: set the saturation equilibrium state between humid air and ice (S30.17)			
set_ice_air_eq_at_a_eta(a, eta) a = A / (kg kg <sup>-1</sup> ) eta = η / (J kg <sup>-1</sup> K <sup>-1</sup> )	Solve $g^{AV} - Ag_A^{AV} = g^{lh}$ with A and η specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_ice_air_eq_at_a_p: set the saturation equilibrium state between humid air and ice (S30.18)			
set_ice_air_eqilibrium_to_a_p(a, p) a = A / (kg kg <sup>-1</sup> ) p = P / Pa	Solve $g^{AV} - Ag_A^{AV} = g^{lh}$ with A and P specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_ice_air_eq_at_a_t: set the saturation equilibrium state between humid air and ice (S30.19)			
set_ice_air_eq_at_a_t(a, t) a = A / (kg kg <sup>-1</sup> ) t = T / K	Solve $g^{AV} - Ag_A^{AV} = g^{lh}$ with A and T specified and store state variables	-	Returns the ErrorReturn value if unsuccessful
set_ice_air_eq_at_t_p: set the saturation equilibrium state between humid air and ice (S30.20)			
set_ice_air_eq_to_t_p(t, p) t = T / K p = P / Pa	Solve $g^{AV} - Ag_A^{AV} = g^{lh}$ with T and P specified and store state variables	-	Returns the ErrorReturn value if unsuccessful

Table S30 (Ice\_Air\_4a), continued

<b>(S30) Ice_Air_4a Module, cont'd</b>	
Function call	Comments
set_it_ctrl_ice_air: set parameters used in the iterative solver routine (S30.21)	
<p>set_it_ctrl_ice_air (key, value)</p> <p>key: a character string that can take the values IT_STEPS, INIT_AIR, INIT_TEMP, INIT_PRESS, TOL_VAP_PRESS, TOL_TEMP or TOL_PRESS.</p> <p>value: a real*8 variable used to specify a control parameter associated with the choice specified by key</p>	<p>This subroutine allows the user to specify details regarding the iterative solver used to determine the equilibrium of ice Ih with humid air. Default choices that should generally work are initialized automatically without calling this function, but users may prefer one of the alternatives available. The character string <b>key</b> is used to determine which function to execute: <b>IT_STEPS</b> allows maxit to be reset, <b>INIT_AIR</b>, <b>INIT_TEMP</b> and <b>INIT_PRESS</b> allow the initial estimates of the mass fraction of dry air in humid air, temperature and pressure to be reset, <b>TOL_VAP_PRESS</b>, <b>TOL_TEMP</b> and <b>TOL_PRESS</b> allow the convergence criteria to be changed for vapour pressure, temperature and total pressure.</p>

Table S31 (Liq\_Air\_4b): Thermodynamic properties of the liquid-air equilibrium. This table deals only with the Gibbs function of wet air, i.e., of the composite system of liquid water and humid air in mutual equilibrium ("cloudy air"). Therefore, the air properties computed here refer to saturated air with respect to liquid water. (See Part I, sections 5.8, ax5.8.)

<b>(S31) Liq_Air_4b Module</b>			
Function call	Mathematical equation	Unit	Comments
liq_air_g_si: the Gibbs function of liquid water-air and its first and second derivatives with respect to the mass fraction of dry air in humid air, absolute temperature and absolute pressure computed from its Helmholtz function (S31.1)			
liq_air_g_si(l, m, n, wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$\frac{\partial^{l+m+n} g^{\text{AW}}}{\partial (w^A)^l \partial T^m \partial P^n}$ $g^{\text{AW}} = \frac{w^A}{A} g^{\text{AV}}$ $+ \left(1 - \frac{w^A}{A}\right) g^{\text{Ih}}$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	$l, m, n \geq 0$ $l + m + n \leq 2$ $w^A \text{ is the mass fraction of dry air relative to that of humid air plus ice}$
liq_air_g_cp_si: specific isobaric heat capacity of ice air (S31.2)			
liq_air_g_cp_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$c_p = -T g_{TT}^{\text{AW}}$	$\frac{\text{J}}{\text{kg K}}$	
liq_air_g_density_si: density of ice air (S31.3)			
liq_air_g_density_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$\rho = 1 / g_P^{\text{AW}}$	$\frac{\text{kg}}{\text{m}^3}$	
liq_air_g_enthalpy_si: specific enthalpy of ice air (S31.4)			
liq_air_g_enthalpy_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$h = g^{\text{AW}} - T g_T^{\text{AW}}$	$\frac{\text{J}}{\text{kg}}$	
liq_air_g_entropy_si: specific entropy of ice air (S31.5)			
liq_air_g_entropy_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$\eta = -g_T^{\text{AW}}$	$\frac{\text{J}}{\text{kg K}}$	
liq_air_g_expansion_si: thermal expansion of ice air (S31.6)			
liq_air_g_expansion_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$\alpha = g_{TP}^{\text{AW}} / g_P^{\text{AW}}$	$\frac{1}{\text{K}}$	

Table S31 (Liq\_Air\_4b), continued

<b>(S31) Liq_Air_4b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_air_g_kappa_t_si: isothermal compressibility of ice air</b> (S31.7)			
liq_air_g_kappa_t_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\kappa_T = -g_{PP}^{AW} / g_P^{AW}$	$\frac{1}{\text{Pa}}$	
<b>liq_air_g_lapserate_si: "moist" adiabatic lapse rate of ice air</b> (S31.8)			
liq_air_g_lapserate_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\Gamma = -g_{TP}^{AW} / g_{TT}^{AW}$	$\frac{\text{K}}{\text{Pa}}$	
<b>liq_air_liquidfraction_si: mass fraction of solid water in ice air</b> (S31.9)			
liq_air_liquidfraction_si(wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$w^W = 1 - w^A / A$	$\frac{\text{kg}}{\text{kg}}$	A is determined by (S29.12)
<b>liq_air_vapourfraction_si: mass fraction of vapour in ice air</b> (S31.10)			
liq_air_vapourfraction_si (wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$w^V = (1 / A - 1)w^A$	$\frac{\text{kg}}{\text{kg}}$	A is determined by (S29.12)

Table S32 (Ice\_Air\_4b): Thermodynamic properties of the ice-air equilibrium. This table deals only with the Gibbs function of ice air, i.e., of the composite system of ice and humid air in mutual equilibrium ("icy air", e.g. cirrus clouds). Therefore, the air properties computed here refer to saturated air in equilibrium with respect to ice. (See Part I, section 5.9, ax5.9.)

<b>(S32) Ice_Air_4b Module</b>			
Function call	Mathematical equation	Unit	Comments
ice_air_g_si: the Gibbs function of ice air and its first and second derivatives with respect to the mass fraction of dry air in humid air, absolute temperature and absolute pressure computed from its Helmholtz function (S32.1)			
ice_air_g_si(l, m, n, wa, t, p) wa = $w^A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$g^{\text{AI}} = \frac{w^A}{A} g^{\text{AV}}$ $+ \left(1 - \frac{w^A}{A}\right) g^{\text{th}}$ $\frac{\partial^{l+m+n} g^{\text{AI}}}{\partial (w^A)^l \partial T^m \partial P^n}$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	$l, m, n \geq 0$ $l + m + n \leq 2$ $w^A$ is the mass fraction of dry air relative to the total mass
ice_air_g_cp_si: specific isobaric heat capacity of ice air (S32.2)			
ice_air_g_cp_si(wa, t, p)	$c_p = -T g_{TT}^{\text{AI}}$	$\frac{\text{J}}{\text{kg K}}$	
ice_air_g_density_si: density of ice air (S32.3)			
ice_air_g_density_si(wa, t, p)	$\rho^{\text{AI}} = 1 / g_P^{\text{AI}}$	$\frac{\text{kg}}{\text{m}^3}$	
ice_air_g_enthalpy_si: specific enthalpy of ice air (S32.4)			
ice_air_g_enthalpy_si(wa, t, p)	$h^{\text{AI}} = g^{\text{AI}} - T g_T^{\text{AI}}$	$\frac{\text{J}}{\text{kg}}$	
ice_air_g_entropy_si: specific entropy of ice air (S32.5)			
ice_air_g_entropy_si(wa, t, p)	$\eta^{\text{AI}} = -g_T^{\text{AI}}$	$\frac{\text{J}}{\text{kg K}}$	
ice_air_g_expansion_si: thermal expansion of ice air (S32.6)			
ice_air_g_expansion_si(wa, t, p)	$\alpha = g_{TP}^{\text{AI}} / g_P^{\text{AI}}$	$\frac{1}{\text{K}}$	
ice_air_g_kappa_t_si: isothermal compressibility of ice air (S32.7)			
ice_air_g_kappa_t_si(wa, t, p)	$\kappa_T = -g_{PP}^{\text{AI}} / g_P^{\text{AI}}$	$\frac{1}{\text{Pa}}$	
ice_air_g_lapserate_si: "moist" adiabatic lapse rate of ice air (S32.8)			
ice_air_g_lapserate_si(wa, t, p)	$\Gamma = -g_{TP}^{\text{AI}} / g_{TT}^{\text{AI}}$	$\frac{\text{K}}{\text{Pa}}$	

Table S32 (Ice\_Air\_4b): continued

<b>(S32) Ice_Air_4b Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
ice_air_solidfraction_si: mass fraction of solid water in ice air			(S32.9)
ice_air_solidfraction_si(wa, t, p)	$w^{\text{th}} = 1 - w^{\text{A}} / A$	$\frac{\text{kg}}{\text{kg}}$	A is given by (S30.11)
ice_air_vapourfraction_si: mass fraction of vapour in ice air			(S32.10)
ice_air_vapourfraction_si(wa, t, p)	$w^{\text{v}} = (1/A - 1)w^{\text{A}}$	$\frac{\text{kg}}{\text{kg}}$	A is given by (S30.11)

Table S33 (Liq\_Air\_4c): Thermodynamic properties of the liquid-air equilibrium. This table deals only with the enthalpy of wet air, and its partial derivatives, depending on the mass fraction of dry air in humid air, entropy and pressure. (See Part I, sections 5.8, ax5.8.)

<b>(S33) Liq_Air_4c Module</b>			
Function call	Mathematical equation	Unit	Comments
liq_air_h_si: enthalpy of wet air as a thermodynamic potential, depending on mass fraction of dry air in humid air, entropy and pressure			(S33.1)
liq_air_h_si (l, m, n, wa, eta, p) wa = $w^{\text{A}} / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$h^{\text{AW}} = g^{\text{AW}} - Tg_T^{\text{AW}}$ $\frac{\partial^{l+m+n}}{\partial (w^{\text{A}})^l \partial T^m \partial P^n} g^{\text{AW}}(w^{\text{A}}, T, P)$	$\frac{\text{J}^{1-n} \text{m}^{3n}}{\text{K}^m \text{kg}}$	$l, m, n \geq 0$ $l + m + n \leq 2$ $w^{\text{A}}$ is the mass fraction of dry air relative to that of humid air plus liquid water
liq_air_h_cp_si: specific isobaric heat capacity of wet air			(S33.2)
liq_air_h_cp_si (wa, eta, p) wa = $w^{\text{A}} / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$c_P = \frac{T}{h_{\eta\eta}^{\text{AW}}}$	$\frac{\text{J}}{\text{kg K}}$	T from (S33.6)
liq_air_h_density_si: density of wet air			(S33.3)
liq_air_h_density_si (wa, eta, p) wa = $w^{\text{A}} / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\rho^{\text{AW}} = \frac{1}{h_p^{\text{AW}}}$	$\frac{\text{kg}}{\text{m}^3}$	

Table S33 (Liq\_Air\_4c), continued

<b>(S33) Liq_Air_4c Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>liq_air_h_kappa_s_si: adiabatic compressibility of wet air (S33.4)</b>			
liq_air_h_kappa_s_si (wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\kappa_s = -\frac{h_{PP}^{\text{AW}}}{h_p^{\text{AW}}}$	$\frac{1}{\text{Pa}}$	
<b>liq_air_h_lapserate_si: moist-adiabatic lapse rate (S33.5)</b>			
liq_air_h_lapserate_si (wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\Gamma = h_{TP}^{\text{AW}}$	$\frac{\text{K}}{\text{Pa}}$	
<b>liq_air_h_temperature_si: temperature of wet air (S33.6)</b>			
liq_air_h_temperature_si(wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$T = h_{\eta}^{\text{AW}}$	K	
<b>liq_air_potdensity_si: potential density of wet air (S33.7)</b>			
liq_air_potdensity_si (a, t, p, pr) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$ pr = $P_r / \text{Pa}$	$\rho_{\theta} = \left[ \left( \frac{\partial h^{\text{AW}}(w^A, \eta, P_r)}{\partial P_r} \right)_{w^A, P_r} \right]^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$\eta$ is total entropy determined by (S31.5) $P_r$ is the reference pressure
<b>liq_air_potenthalpy_si: potential enthalpy of wet air (S33.8)</b>			
liq_air_potenthalpy_si (a, t, p, pr) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$ pr = $P_r / \text{Pa}$	$h_{\theta} = h^{\text{AW}}(w^A, \eta, P_r)$	$\frac{\text{J}}{\text{kg}}$	$\eta$ is total entropy determined by (S31.5) $P_r$ is the reference pressure
<b>liq_air_pottemp_si: potential temperature of wet air (S33.9)</b>			
liq_air_pottemp_si (a, t, p, pr) wa = $w^A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$ pr = $P_r / \text{Pa}$	$T = T_0 + \theta$ $= \left( \frac{\partial h^{\text{AW}}(w^A, \eta, P_r)}{\partial \eta} \right)_{w^A, P_r}$	K	$\eta$ is total entropy determined by (S31.5) $T_0 = 273.15 \text{ K}$ $\theta$ is potential temperature in $^{\circ}\text{C}$



Table S34 (Ice\_Air\_4c): Thermodynamic properties of the ice-air equilibrium. This table deals with the enthalpy of ice air, as well as its partial derivatives, depending on mass fraction of dry air in humid air, entropy and pressure. (See Part I, section 5.9, ax5.9.)

<b>(S34) Ice_Air_4c Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_air_h_si: enthalpy of wet air as a thermodynamic potential, depending on mass fraction of dry air in humid air, entropy and pressure (S34.1)</b>			
ice_air_h_si (l, m, n, wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$h^{\text{AI}} = g^{\text{AI}} - Tg_T^{\text{AI}}$ $\frac{\partial^{l+m+n}}{\partial (w^A)^l \partial \eta^m \partial P^n} h^{\text{AI}}(w^A, \eta, P)$	$\frac{\text{J}^{1-n-m} \text{m}^{3n}}{\text{K}^{-m} \text{kg}^{1-m}}$	n, m, l ≥ 0 n + m + l ≤ 2 w <sup>A</sup> is the mass fraction of dry air relative to the total mass
<b>ice_air_h_cp_si: specific isobaric heat capacity of wet air (S34.2)</b>			
ice_air_h_cp_si (wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$c_P = \frac{T}{h_{\eta\eta}^{\text{AI}}}$	$\frac{\text{J}}{\text{kg K}}$	T from (S34.6)
<b>ice_air_h_density_si: density of wet air (S34.3)</b>			
ice_air_h_density_si (wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\rho = (h_P^{\text{AI}})^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	
<b>ice_air_h_kappa_s_si: adiabatic compressibility of wet air (S34.4)</b>			
ice_air_h_kappa_s_si (wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\kappa_S = -\frac{h_{PP}^{\text{AI}}}{h_P^{\text{AI}}}$	$\frac{1}{\text{Pa}}$	
<b>ice_air_h_lapserate_si: moist-adiabatic lapse rate (S34.5)</b>			
ice_air_h_lapserate_si(wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$\Gamma = h_{TP}^{\text{AI}}$	$\frac{\text{K}}{\text{Pa}}$	
<b>ice_air_h_temperature_si: temperature of wet air (S34.6)</b>			
ice_air_h_temperature_si(wa, eta, p) wa = $w^A / (\text{kg kg}^{-1})$ eta = $\eta / (\text{J kg}^{-1} \text{K}^{-1})$ p = $P / \text{Pa}$	$T = h_{\eta}^{\text{AI}}$	$\text{K}$	

Table S34 (Ice\_Air\_4c), continued

<b>(S34) Ice_Air_4c Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>ice_air_potdensity_si: potential density of wet air (S34.7)</b>			
ice_air_potdensity_si(a, t, p, pr) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa pr = P <sub>r</sub> / Pa	$\rho_{\theta} = \left[ \left( \frac{\partial h^{AI}(w^A, \eta, P_r)}{\partial P_r} \right)_{w^A, P_r} \right]^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$\eta$ is total entropy determined by (S32.5) $P_r$ is the reference pressure
<b>ice_air_potenthalpy_si: potential enthalpy of wet air (S34.8)</b>			
ice_air_potenthalpy_si(a, t, p, pr) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa pr = P <sub>r</sub> / Pa	$h_{\theta} = h^{AI}(w^A, \eta, P_r)$	$\frac{\text{J}}{\text{kg}}$	$\eta$ is total entropy determined by (S32.5) $P_r$ is the reference pressure
<b>ice_air_pottemp_si: potential temperature of wet air (S34.9)</b>			
ice_air_pottemp_si(a, t, p, pr) a = A / (kg kg <sup>-1</sup> ) t = T / K p = P / Pa pr = P <sub>r</sub> / Pa	$T = T_0 + \theta$ $= \left( \frac{\partial h^{AI}(w^A, \eta, P_r)}{\partial \eta} \right)_{w^A, P_r}$	K	$\eta$ is total entropy determined by (S32.5) $T_0 = 273.15$ K $\theta$ is potential temperature in °C

Level 5: Approximate relations for initialisation of iterative routines

Table S35 (Flu\_IF97\_5): Implementations of the IF97 industrial formulation approximations to the density and Gibbs potential functions for pure liquid water and water vapour. (See Part I, section 6.)

<b>(S35) Flu_IF97_5 Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>chk_iapws97_table():</b> Check routine for the industrial formulation IF-97 of IAPWS-95 (S35.1)			
chk_iapws97_table accepts arguments of 5 and 15 corresponding to tables 5 and 15 of IAPWS-IF97. Table 5 compares results for $(T, P) = (300 \text{ K}, 3 \text{ MPa}), (300\text{K}, 80 \text{ MPa})$ and $(500 \text{ K}, 3 \text{ MPa})$ and table 15 compares results for $(T, P) = (300 \text{ K}, 3500 \text{ Pa}), (700 \text{ K}, 3500 \text{ Pa})$ and $(700\text{K}, 30 \text{ KPa})$ . Locally calculated values of Gibbs energy, enthalpy, internal energy, entropy, specific heat capacity at constant pressure and sound speed are compared with published results. The published results are shown to the number of digits expected to be reproduced by double precision code.			
<b>fit_liq_density_if97_si:</b> density of liquid water as a function of temperature and pressure, in region 1 (liquid) in IAPWS-IF97 (S35.2)			
fit_liq_density_if97_si(t, p) t = T / K p = P / Pa	$\left[ \frac{\partial g^{\text{IF97(1)}}}{\partial P} \right]^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{IF97(1)}}$ represents the Gibbs potential function corresponding to region 1 in the IF97 industrial formulation
<b>fit_liq_g_if97_si:</b> Gibbs function and its 1st and 2nd derivatives with respect to temperature and pressure, as defined for region 1 (liquid) in IAPWS-IF97 (S35.3)			
fit_liq_g_if97_si(n, m, t, p) t = T / K p = P / Pa	$\frac{\partial^{n+m}}{\partial T^n \partial P^m} g^{\text{IF97(1)}}(T, P)$	$\frac{\text{J m}^{3m}}{\text{K}^n \text{kg}^{m+1}}$	See comment for (S35.2)
<b>fit_vap_density_if97_si:</b> density of water vapour as a function of temperature and pressure, in region 2 (vapour) in IAPWS-IF97 (S35.4)			
fit_vap_density_if97_si(t, p) t = T / K p = P / Pa	$\left[ \frac{\partial g^{\text{IF97(2)}}}{\partial P} \right]^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{IF97(2)}}$ represents the Gibbs potential function corresponding to region 2 in the IF97 industrial formulation
<b>fit_vap_g_if97_si:</b> Gibbs function and its 1st and 2nd derivatives with respect to temperature and pressure, as defined for region 2 (vapour) in IAPWS-IF97 (S35.5)			
fit_vap_g_if97_si(n, m, t, p) t = T / K p = P / Pa	$\frac{\partial^{n+m}}{\partial T^n \partial P^m} g^{\text{IF97(2)}}(T, P)$	$\frac{\text{J m}^{3m}}{\text{K}^n \text{kg}^{m+1}}$	See comment for (S35.4)

Table S36 (Ice\_Flu\_5): Implementation of the correlation functions for the melting and sublimation of ice as proposed in Feistel (2006).

<b>(S36) Ice_Flu_5 Module</b>			
Function call	Mathematical equation	Unit	Comments
fit_ice_liq_pressure_si : Melting pressure as a function of temperature			(S36.1)
fit_ice_liq_pressure_si(t) t = T / K	Determined from a fit of the function ice_liq_pressure_si, as in IAPWS (2008b)	Pa	
fit_ice_liq_temperature_si : the melting temperature of ice as a function of pressure			(S36.2)
fit_ice_liq_temperature_si(p) p = P / Pa	Determined from a fit of the function ice_liq_temperature_si, unpublished	K	
fit_ice_vap_pressure_si : the sublimation pressure as a function of temperature			(S36.3)
fit_ice_vap_pressure_si(t) t = T / K	Determined from a fit of the function ice_vap_pressure_si, as in IAPWS (2008b)	Pa	

Table S37 (Sea\_5a): Thermohaline properties of seawater expressed in common oceanographic units. The conversion fraction of conservative temperature,  $\Theta = h^\theta / c_p^0$ , with respect to potential enthalpy,  $h^\theta$ , is  $c_p^0 = 3991.867\ 957\ 119\ 63\ \text{J kg}^{-1}\ \text{K}^{-1}$ . (See Part I, sections 4.3, ax4.3.)

<b>(S37) Sea_5a Module</b>			
Function call	Mathematical equation	Unit	Comments
sea_alpha_ct_si: thermal expansion coefficient of seawater with respect to conservative temperature, in basic SI units			(S37.1)
sea_alpha_ct_si(s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = T / K p = P / Pa	$\alpha^\Theta = c_p^0 \alpha^h$ $= -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial \Theta} \right)_{S_A, P}$	$\frac{1}{\text{K}}$	$\alpha^h$ is computed from sea_eta_expansion_h_si, (S17.6)

Table S37 (Sea\_5a), continued

<b>(S37) Sea_5a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_alpha_pt0_si</b> : thermal expansion coefficient of seawater with respect to potential temperature, in basic SI units (S37.2)			
<b>sea_alpha_pt0_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\alpha^\theta = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial \theta} \right)_{S_A, P}$	$\frac{1}{\text{K}}$	$\alpha^\theta$ is computed from <b>sea_eta_expansion_theta_si</b> , (S17.8)
<b>sea_alpha_t_si</b> : thermal expansion coefficient of seawater, in basic SI units (S37.3)			
<b>sea_alpha_t_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\alpha^T = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right)_{S_A, P}$	$\frac{1}{\text{K}}$	$\alpha^T$ is computed from <b>sea_g_expansion_t_si</b> , (S12.10)
<b>sea_beta_ct_si</b> : haline contraction coefficient of seawater at fixed conservative temperature, in basic SI units (S37.4)			
<b>sea_beta_ct_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta^\theta = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{h^\theta, P}$	$\frac{\text{kg}}{\text{kg}}$	$\beta^\theta$ is computed from <b>sea_eta_contraction_h_si</b> , (S17.1)
<b>sea_beta_pt0_si</b> : haline contraction coefficient of seawater at fixed potential temperature, in basic SI units (S37.5)			
<b>sea_beta_pt0_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta^\theta = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{\theta, P}$	$\frac{\text{kg}}{\text{kg}}$	$\beta^\theta$ is computed from <b>sea_eta_contraction_theta_si</b> , (S17.3)
<b>sea_beta_t_si</b> : haline contraction coefficient of seawater at fixed in situ temperature, in basic SI units (S37.6)			
<b>sea_beta_t_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta^T = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{T, P}$	$\frac{\text{kg}}{\text{kg}}$	$\beta^T$ is computed from <b>sea_g_contraction_t_si</b> , (S12.9)
<b>sea_cabb_ct_si</b> : cabbeling coefficient with respect to conservative temperature, in basic SI units (S37.7)			
<b>sea_cabb_ct_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$C_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial \Theta} \right)_{S_A, P} + 2 \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \alpha^\theta}{\partial S_A} \right)_{\Theta, P} - \left( \frac{\alpha^\theta}{\beta^\theta} \right)^2 \left( \frac{\partial \beta^\theta}{\partial S_A} \right)_{\Theta, P}$	$\frac{1}{\text{K}^2}$	Third derivatives of the potential function are done by finite differences

Table S37 (Sea\_5a), continued

<b>(S37) Sea_5a Module, cont'd</b>			
Function call	Mathematical equation	Unit	Comments
<b>sea_cabb_pt0_si: cabbeling coefficient with respect to potential temperature, in basic SI units (S37.8)</b>			
sea_cabb_pt0_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$C_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial \theta} \right)_{S_A, P} + 2 \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \alpha^\theta}{\partial S_A} \right)_{\theta, P} - \left( \frac{\alpha^\theta}{\beta^\theta} \right)^2 \left( \frac{\partial \beta^\theta}{\partial S_A} \right)_{\theta, P}$	$\frac{1}{\text{K}^2}$	Third derivatives of the potential function are done by finite differences
<b>sea_ctmp_from_ptmp0_si: conservative temperature from absolute salinity and potential temperature, in basic SI units (S37.9)</b>			
sea_ctmp_from_ptmp0_si(s, t) $s = S_A / (\text{kg kg}^{-1})$ $t = \theta / \text{K}$	$\Theta = 273.15 \text{K} + h(S_A, \theta, P_0) / c_p^0$	K	The reference pressure is $P_0 = 101325 \text{ Pa}$
<b>sea_ptmp0_from_ctmp_si: potential temperature from absolute salinity and conservative temperature, in basic SI units (S37.10)</b>			
sea_ptmp0_from_ctmp_si(s, t) $s = S_A / (\text{kg kg}^{-1})$ $t = \Theta / \text{K}$	$\theta$ determined from $\Theta = 273.15 \text{K} + h(S_A, \theta, P_0) / c_p^0$	K	The reference pressure is $P_0 = 101325 \text{ Pa}$
<b>sea_thrmb_ct_si: thermobaric coefficient with respect to conservative temperature, in basic SI units (S37.11)</b>			
sea_thrmb_ct_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$T_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial P} \right)_{S_A, \Theta} - \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \beta^\theta}{\partial P} \right)_{S_A, \Theta}$	$\frac{1}{\text{K Pa}}$	Third derivatives of the potential function are done by finite differences
<b>sea_thrmb_pt0_si: thermobaric coefficient with respect to potential temperature, in basic SI units (S37.12)</b>			
sea_thrmb_pt0_si(s, t, p) $s = S_A / (\text{kg kg}^{-1})$ $t = T / \text{K}$ $p = P / \text{Pa}$	$T_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial P} \right)_{S_A, \theta} - \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \beta^\theta}{\partial P} \right)_{S_A, \theta}$	$\frac{1}{\text{K Pa}}$	Third derivatives of the potential function are done by finite differences

Table S38 (Air\_5): This module implements properties of moist air that are expressed or accept inputs in units other than basic SI. (See Part I, section 3.4.)

<b>(S38) Air_5 Module</b>			
Function call	Mathematical equation	Unit	Comments
<b>air_lapserate_moist_c100m:</b> Returns the adiabatic lapse rate of humid air in deg c per 100 m altitude change (S38.1)			
air_lapserate_moist_c100m (rh, t, p)  rh = RH <sub>WMO</sub> in % t = t / °C p = P / hPa	$\frac{g \times 100 \times f_{Tp}^{AV}}{\rho (f_{Tp}^{AV})^2 - f_{TT}^{AV} (2f_{\rho}^{AV} + \rho f_{\rho\rho}^{AV})}$	$\frac{^{\circ}\text{C}}{100\text{m}}$	Computed from air_f_lapserate_si which gives the lapse rate with respect to pressure. Multiplying by 100×ρ×g gives the lapse rate per 100 meters.

Table S39 (Liq\_F03\_5): Implementations of the IAPWS-09 Gibbs function for liquid-water (Feistel, 2003) and selected properties based on it. This function is far more efficient than IAPWS-95 but its validity is restricted to the Neptunian range of temperatures and pressures. (See Part I, section 6.)

<b>(S39) Liq_F03_5</b>			
Function call	Mathematical equation	Unit	Comments
<b>chk_iapws09_table6:</b> Check values for the pure liquid water Gibbs function of Feistel (2003) (S39.1)			
chk_iapws09_table6 produces a comparison of locally calculated results for pure liquid water with those published in Table 6 of IAPWS-09 which is based on Feistel's (2003) Gibbs function formulation for pure liquid water in the Neptunian range of <i>T</i> and <i>P</i> . Results are compared successively for ( <i>T</i> , <i>P</i> ) = (273.15 K, 101325 Pa), (273.15 K, 10 <sup>8</sup> Pa) and (313.15 K, 101325 Pa). For each <i>T-P</i> pair, results are compared for the Gibbs function, its first and second derivatives as well as the enthalpy, Helmholtz energy, internal energy, entropy, specific heat at constant pressure and soundspeed. The published results are shown to the number of digits expected to be reproduced by double precision code.			
<b>fit_liq_cp_f03_si:</b> specific isobaric heat capacity cp of liquid water computed from the Gibbs function of Feistel (2003) (S39.2)			
fit_liq_cp_f03_si(t, p) t = T / K p = P / Pa	$c_p = -T g_{TT}^{F03}$	$\frac{\text{J}}{\text{kg K}}$	$g^{F03}$ is the Gibbs function of liquid water (IAPWS 2009) and subscripts represent partial derivatives

Table S39 (Liq\_F03\_5), continued

<b>(S39) Liq_F03_5</b>			
Function call	Mathematical equation	Unit	Comments
fit_liq_density_f03_si: density of pure liquid water computed from the Gibbs function of Feistel (2003) (S39.3)			
fit_liq_density_f03_si(t, p) t = T / K p = P / Pa	$\rho = [g_P^{F03}]^{-1}$	$\frac{\text{kg}}{\text{m}^3}$	See comment for (S39.2)
fit_liq_expansion_f03_si: thermal expansion of liquid water computed from the Gibbs function of Feistel (2003) (S39.4)			
fit_liq_expansion_f03_si (t, p) t = T / K p = P / Pa	$\alpha = \frac{g_{PT}^{F03}}{g_P^{F03}}$	$\frac{1}{\text{K}}$	See comment for (S39.2)
fit_liq_g_f03_si: Gibbs function of pure water as defined in Feistel (2003) (S39.5)			
fit_liq_g_f03_si(n, m, t, p) t = T / K p = P / Pa	$\frac{\partial^{n+m} g^{F03}}{\partial T^n \partial P^m}$	$\frac{\text{J}^{1-m} \text{m}^{3m}}{\text{K}^n \text{kg}}$	n, m ≥ 0 n + m ≤ 2
fit_liq_kappa_t_f03_si: isothermal compressibility of liquid water computed from the Gibbs function of Feistel (2003) (S39.6)			
fit_liq_kappa_t_f03_si (t, p) t = T / K p = P / Pa	$\kappa_T = -\frac{g_{PP}^{F03}}{g_P^{F03}}$	$\frac{1}{\text{Pa}}$	See comment for (S39.2)
fit_liq_soundspeed_f03_si: sound speed of pure liquid water computed from the Gibbs function of Feistel (2003) (S39.7)			
fit_liq_soundspeed_f03_si (t, p) t = T / K p = P / Pa	$c = g_P^{F03} \sqrt{\frac{g_{TT}^{F03}}{(g_{TP}^{F03})^2 - g_{TT}^{F03} g_{PP}^{F03}}}$	$\frac{\text{m}}{\text{s}}$	See comment for (S39.2)



Table S40 (OS2008\_5): Implementation of checking routines for each of the tables in Feistel et al. (2008).

**(S40) OS2008\_5 Module**

Function call	Comments
chk_os2008_table():	Check values for pure fluid water, ice and seawater as tabulated in Feistel et al. (2008). (S40.1)
chk_os2008_table() produces a set of comparison tables between locally calculated results and the results tabulated in Feistel et al. (2008). The input argument refers to table labels in this publication and takes the values 2, 3, A2-A8. Feistel et al. used quadruple precision calculations to ensure mutual consistency between the formulations for pure fluid water, ice Ih and seawater, thus updating the check tables that were published in previous IAPWS Releases. The published results are shown to the accuracy given in Feistel et al. (2008) with a vertical line to indicate the digits expected to be accurately reproduced with double precision (64 bit) calculations.	

Table S41 (GSW\_Library\_5): Thermohaline properties of seawater included in the GSW Library (see [www.teos-10.org](http://www.teos-10.org)). The input and output units and the function names correspond to those in the GSW Library, but the routines use the approaches and routines used in the SIA Library in order to provide a cross-check on the results obtained. In all cases, we use the routine fit\_liq\_g\_f03\_si for the calculations done in this module so that numerical efficiency is improved and results agree to within roundoff error.

**(S41) GSW\_Library\_5 Module**

Function call	Mathematical equation	Unit	Comments
gsw_alpha_ct: thermal expansion coefficient of seawater with respect to conservative temperature, in practical oceanographic units			(S41.1)
gsw_alpha_ct(s, t, p) s = S <sub>A</sub> / (g kg <sup>-1</sup> ) t = t / °C p = p / dbar	$\alpha^\Theta = c_p^0 \alpha^h$ $= -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial \Theta} \right)_{S_A, P}$	$\frac{1}{^\circ\text{C}}$	Computed by calling the corresponding si routine (S37.1)
gsw_alpha_pt0: thermal expansion coefficient of seawater with respect to potential temperature, in practical oceanographic units			(S41.2)
gsw_alpha_pt0(s, t, p) s = S <sub>A</sub> / (g kg <sup>-1</sup> ) t = t / °C p = p / dbar	$\alpha^\theta = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial \theta} \right)_{S_A, P}$	$\frac{1}{^\circ\text{C}}$	Computed by calling the corresponding si routine (S37.2)
gsw_alpha_t: thermal expansion coefficient of seawater with respect to in situ temperature, in practical oceanographic units			(S41.3)
gsw_alpha_t(s, t, p) s = S <sub>A</sub> / (g kg <sup>-1</sup> ) t = t / °C p = p / dbar	$\alpha^T = -\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right)_{S_A, P}$	$\frac{1}{^\circ\text{C}}$	Computed by calling the corresponding si routine (S37.3)

Table S41 (GSW\_Library\_5), continued

<b>(S41) GSW_Library_5 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>gsw_asal_from_psal</b> : Converts from Practical Salinity to Absolute Salinity, possibly allowing for composition anomalies using a lookup table. (S41.4)			
<b>gsw_asal_from_psal</b> (sp, long, lat, p) sp = $S_P$ (Practical Salinity) long = longitude in degrees lat = latitude in degrees p = $P$ / dbar	$S_A = [S_P \times (35.16504/35) + \delta S_A(\text{long, lat, p})] \text{ kg kg}^{-1}$ from McDougall et al (2009).	$\frac{\text{g}}{\text{kg}}$	Note that $P$ , long and lat are required to estimate the effect of composition anomalies on Absolute Salinity. If absent the effect is neglected. $P$ is the sea pressure.
<b>gsw_beta_ct</b> : haline contraction coefficient of seawater at fixed in situ temperature, in practical oceanographic units (S41.5)			
<b>gsw_beta_ct</b> (s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $t / ^\circ\text{C}$ p = $p$ / dbar	$\beta^\Theta = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{h^\Theta, P}$	$\frac{\text{kg}}{\text{g}}$	Computed by calling the corresponding si routine (S37.4)
<b>gsw_beta_pt0_si</b> : haline contraction coefficient of seawater at fixed potential temperature, in practical oceanographic units (S41.6)			
<b>sea_beta_pt0_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta^\theta = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{\theta, P}$	$\frac{\text{kg}}{\text{kg}}$	Computed by calling the corresponding si routine (S37.5)
<b>sea_beta_t_si</b> : haline contraction coefficient of seawater at fixed in situ temperature, in practical oceanographic units (S41.7)			
<b>sea_beta_t_si</b> (s, t, p) s = $S_A / (\text{kg kg}^{-1})$ t = $T / \text{K}$ p = $P / \text{Pa}$	$\beta^T = \frac{1}{\rho} \left( \frac{\partial \rho}{\partial S_A} \right)_{T, P}$	$\frac{\text{kg}}{\text{kg}}$	Computed by calling the corresponding si routine (S37.6)
<b>gsw_cabb_ct</b> : cabbeling coefficient with respect to conservative temperature, in practical oceanographic units (S41.8)			
<b>gsw_cabb_ct</b> (s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $t / ^\circ\text{C}$ p = $p$ / dbar	$C_b^\Theta = \left( \frac{\partial \alpha^\Theta}{\partial \Theta} \right)_{S_A, P} + 2 \frac{\alpha^\Theta}{\beta^\Theta} \left( \frac{\partial \alpha^\Theta}{\partial S_A} \right)_{\Theta, P} - \left( \frac{\alpha^\Theta}{\beta^\Theta} \right)^2 \left( \frac{\partial \beta^\Theta}{\partial S_A} \right)_{\Theta, P}$	$\frac{1}{\text{K}^2}$	Computed by calling the corresponding si routine (S37.7)
<b>gsw_cabb_pt0</b> : cabbeling coefficient with respect to potential temperature, in practical oceanographic units (S41.9)			

<p>gsw_cabb_pt0(s, t, p)  s = <math>S_A / (\text{g kg}^{-1})</math>  t = <math>t / ^\circ\text{C}</math>  p = <math>p / \text{dbar}</math></p>	$C_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial \theta} \right)_{S_A, P} + 2 \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \alpha^\theta}{\partial S_A} \right)_{\theta, P} - \left( \frac{\alpha^\theta}{\beta^\theta} \right)^2 \left( \frac{\partial \beta^\theta}{\partial S_A} \right)_{\theta, P}$	$\frac{1}{\text{K}^2}$	<p>Computed by calling the corresponding si routine (S37.8)</p>
<p><b>gsw_cp: specific isobaric heat capacity cp of seawater computed using (S41.14) (S41.10)</b></p>			
<p>gsw_cp(s, t, p)  s = <math>S_A / (\text{g kg}^{-1})</math>  t = <math>t / ^\circ\text{C}</math>  p = <math>p / \text{dbar}</math></p>	$c_p = -T g_{TT}^{\text{GSW}}$	$\frac{\text{J}}{\text{kg K}}$	<p><math>g^{\text{GSW}}</math> from (S41.15) and subscripts represent partial derivatives</p>

Table S41 (GSW\_Library\_5), continued

<b>(S41) GSW_Library_5 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
gsw_ctmp_from_ptmp0: conservative temperature from absolute salinity and potential temperature referenced to the surface, in standard oceanographic units (S41.11)			
gsw_ctmp_from_ptmp0 (s, t) s = $S_A / (\text{g kg}^{-1})$ t = $\theta / ^\circ\text{C}$	$\Theta = h(S_A, \theta, P_0) / c_P^0$	$^\circ\text{C}$	Enthalpy is computed using (S41.13) and the reference pressure is $P_0 = 0$ dbar
gsw_dens: density of seawater (S41.12)			
gsw_dens(s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$\rho = \frac{1}{g_P^{\text{GSW}}}$	$\frac{\text{kg}}{\text{m}^3}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives
gsw_enthalpy: specific enthalpy based on (S41.14) (S41.13)			
gsw_enthalpy(s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$h = g^{\text{GSW}} - T g_T^{\text{GSW}}$	$\frac{\text{J}}{\text{kg}}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives
gsw_entropy: specific entropy based on (S41.14) (S41.14)			
sea_entropy_si(s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$\eta = -g_T^{\text{GSW}}$	$\frac{\text{J}}{\text{kg K}}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives
gsw_g: Gibbs function of seawater determined using fit_liq_g_f03_si plus sal_g_si (S41.15)			
gsw_g(n, m, t, p) t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$\frac{\partial^{n+m} g^{\text{GSW}}}{\partial T^n \partial P^m}$	$\frac{\text{J}^{1-m} \text{m}^{3m}}{\text{K}^n \text{kg}}$	n, m $\geq 0$ n + m $\leq 2$
gsw_kappa: isentropic compressibility based on (S41.14) (S41.16)			
gsw_kappa(s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$\kappa_s = -\frac{1}{v} \left( \frac{\partial v}{\partial P} \right)_{S_A, \eta}$ $= \frac{(g_{TP}^{\text{GSW}})^2 - g_{TT}^{\text{GSW}} g_{PP}^{\text{GSW}}}{g_P^{\text{GSW}} g_{TT}^{\text{GSW}}}$	$\frac{1}{\text{Pa}}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives
gsw_kappa_t: isothermal compressibility based on (S41.14) (S41.17)			
sea_kappa_t(s, t, p) s = $S_A / (\text{g kg}^{-1})$ t = $T / ^\circ\text{C}$ p = $P / \text{dbar}$	$\kappa_T = -\frac{1}{v} \left( \frac{\partial v}{\partial P} \right)_{S_A, T}$ $= -\frac{g_{PP}^{\text{GSW}}}{g_P^{\text{GSW}}}$	$\frac{1}{\text{Pa}}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives

Table S41 (GSW\_Library\_5), continued

<b>(S41) GSW_Library_5 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
<b>gsw_pden: potential density based on (S41.14) and (S41.18)</b> (S41.18)			
sea_pden(s, t, p, pr) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{dbar}$ $pr = P_r / \text{dbar}$	$\rho^\theta = \frac{1}{g_P^{\text{GSW}}(S_A, \theta, P_r)}$	$\frac{\text{kg}}{\text{m}^3}$	$\theta$ determined from (S41.20) $P_r$ is the reference sea pressure and subscripts represent partial derivatives
<b>gsw_psal_from_asal: Converts from Absolute Salinity to Practical Salinity, possibly allowing for composition anomalies using a lookup table.</b> (S41.19)			
gsw_psal_from_asal (s, long, lat, p) $s = S_A / (\text{g kg}^{-1})$ long = longitude in degrees lat = latitude in degrees $p = P / \text{dbar}$	$S_P = (35/35.16504) \times (S_A - \delta S_A(\text{long, lat, p}))$ from McDougall et al (2009).	1	Note that $P$ , long and lat are required to estimate the effect of composition anomalies on Absolute Salinity. If absent the effect is neglected. $P$ is the sea pressure.
<b>gsw_ptmp: potential temperature</b> (S41.20)			
sea_ptmp(s, t, p, pr) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{Pa}$ $pr = P_r / \text{Pa}$	$\theta$ determined by iterative solution of $\eta = -g_T^{\text{GSW}}(S_A, \theta, P_r)$ with $\eta = -g_T^{\text{GSW}}(S_A, T, P)$	$^\circ\text{C}$	$g_T^{\text{GSW}}$ from (S41.15). Determine $\eta$ and then solve iteratively for $\theta$ .
<b>gsw_ptmp0_from_ctmp: potential temperature from absolute salinity and conservative temperature</b> (S41.21)			
gsw_ptmp0_from_ctmp (s, t) $s = S_A / (\text{g kg}^{-1})$ $t = \Theta / ^\circ\text{C}$	$\theta$ determined from iterative solution of $\Theta = h(S_A, \theta, P_0) / c_p^0$	$^\circ\text{C}$	The reference pressure is $P_0 = 0 \text{ dbar}$
<b>gsw_specvol: specific volume based on (S41.14)</b> (S41.22)			
ice_specific_volume_si (s, t, , p) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{dbar}$	$v = g_P^{\text{GSW}}$	$\frac{\text{m}^3}{\text{kg}}$	$g^{\text{GSW}}$ from (S41.15) and subscripts represent partial derivatives

Table S41 (GSW\_Library\_5), continued

<b>(S41) GSW_Library_5 Module (cont'd)</b>			
Function call	Mathematical equation	Unit	Comments
gsw_svel: speed of sound based on (S41.14)			(S41.23)
sea_svel(s, t, p) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{dbar}$	$c = g_P^{\text{GSW}} \sqrt{\frac{g_{TT}^{\text{GSW}}}{(g_{TP}^{\text{GSW}})^2 - g_{TT}^{\text{GSW}} g_{PP}^{\text{GSW}}}}$	$\frac{\text{m}}{\text{s}}$	$g^{\text{GSW}}$ is the Gibbs function (S41.15)
sea_thrmb_ct: thermobaric coefficient with respect to conservative temperature			(S41.24)
sea_thrmb_ct(s, t, p) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{dbar}$	$T_b^\ominus = \left( \frac{\partial \alpha^\ominus}{\partial P} \right)_{S_A, \ominus}$ $- \frac{\alpha^\ominus}{\beta^\ominus} \left( \frac{\partial \beta^\ominus}{\partial P} \right)_{S_A, \ominus}$	$\frac{1}{\text{K dbar}}$	Computed by calling the corresponding si routine (S41.11)
sea_thrmb_pt0: thermobaric coefficient with respect to potential temperature			(S41.25)
sea_thrmb_pt0(s, t, p) $s = S_A / (\text{g kg}^{-1})$ $t = T / ^\circ\text{C}$ $p = P / \text{dbar}$	$T_b^\theta = \left( \frac{\partial \alpha^\theta}{\partial P} \right)_{S_A, \theta}$ $- \frac{\alpha^\theta}{\beta^\theta} \left( \frac{\partial \beta^\theta}{\partial P} \right)_{S_A, \theta}$	$\frac{1}{\text{K dbar}}$	Computed by calling the corresponding si routine (S41.12)

Table S42 (Convert\_5): A routine to convert between different measures of pressure, temperature and salinity in common use. The routines to convert between Practical Salinity and Absolute Salinity are in Convert\_0 (Table S1)

<b>(S42) Convert_5 Module</b>		
Function call	Mathematical equation	Comments
<b>cnv_pressure: Converts between different pressure units (S42.1)</b>		
<p>cnv_pressure (unit_out, input, unit_in)</p> <p>unit_in = any of the units listed to the right p_in = <math>P / \text{unit\_in}</math> unit_out = any of the units listed to the right</p>	$P/\text{Pa} = 10^6 P/\text{MPa}$ $P/\text{Pa} = 10^3 P/\text{kPa}$ $P/\text{Pa} = 10^2 P/\text{hPa}$ $P/\text{Pa} = 10^4 P^{\text{sea}}/\text{dbar} + 101325$ $P/\text{Pa} = 10^8 P/\text{kbar}$ $P/\text{Pa} = 10^5 P/\text{bar}$ $P/\text{Pa} = 10^2 P/\text{mbar}$ $P/\text{Pa} = P/\text{torr} * 101325 / 760$ $P/\text{Pa} = P/\text{mmHg} * 101325 / 760$ $P/\text{Pa} = P/(1 \text{ atm}) * 98.0665$ $P/\text{Pa} = P/\text{psi} * 6894.8$	<p>Any combination of input and output units can be specified. All units must be specified in capitals.</p> <p>MPa =&gt; "MPA" kPa =&gt; "KPA" hPa =&gt; "HPA" Pa =&gt; "PA" dbar =&gt; "DBAR" kbar =&gt; "KBAR" bar =&gt; "BAR" mbar =&gt; "MBAR" torr =&gt; "TORR" mm Hg =&gt; "MMHG" atm =&gt; "ATM" or "KGF" psi =&gt; "PSI" or "LBF/IN2"</p>
<b>cnv_salinity: Converts between different salinity measures (S42.2)</b>		
<p>cnv_salinity (unit_out, s, unit_in, t, p, long, lat)</p> <p>unit_in = any of the units listed to the right s = <math>S / \text{unit\_in}</math> input = any of the units listed to the right t = <math>T / \text{K}</math> p = <math>P / \text{Pa}</math> long = longitude in degrees lat = latitude in degrees</p>	$S_R/(\text{kg kg}^{-1}) = Cl(\text{‰}) \times 1.80655 \times 0.03516504 / 35$ $S_R/(\text{kg kg}^{-1}) = S_P \times 0.03516504 / 35$ <p><math>S_P</math> is related to conductivity temperature and pressure by the full PSS78 conversion formulae. <math>T</math> and <math>P</math> are required for this conversion; if absent, <math>T = 25 \text{ °C}</math> and <math>P = 101325 \text{ Pa}</math>, are assumed.</p> <p><math>S_A</math> is related to <math>S_P</math> by the conversion routines in Convert_0.</p>	<p>Any combination of input and output units can be specified. All units must be specified in capitals.</p> <p>‰ =&gt; "CL" (chlorinity) sm/m =&gt; "S/M", "SM/M" or "CND" (conductivity) pss78 =&gt; "PSU" or "PSS" kg/kg =&gt; "KG/KG(REF)" for Reference Salinity kg/kg =&gt; "KG/KG(ABS)" for Absolute Salinity</p> <p>Position is required to estimate the effect of composition anomalies on Absolute Salinity. If absent the effect is neglected.</p>

Table S42 (Convert\_5), continued

<b>(S42) Convert_5 Module (cont'd)</b>		
Function call	Mathematical equation	Comments
<b>cnv_temperature: Converts between different temperature units (S42.3)</b>		
<p><b>cnv_temperature</b> (unit_out, input, unit_in)</p> <p>unit_out = any of the units listed to the right but T48 output requires T48 input</p> <p>t_in = T / unit_in</p> <p>unit_in = any of the units listed to the right</p>	<p><math>T/K(T68) = \text{poly1}(T/K(T48))</math>  <math>T/K(T68) = \text{poly2}(T/K(T90))</math>  <math>T/K(T90) = \text{poly3}(T/K(T48))</math>  <math>T/K(T90) = \text{poly4}(T/K(T68))</math></p> <p><math>T/K = T / ^\circ\text{C} + 273.15</math>  <math>T / ^\circ\text{F} = 1.8 T / ^\circ\text{C} + 32</math></p> <p>poly1-4 are distinct polynomials</p>	<p>Outputs on the T48 temperature scale are not supported, but any other combination of input and output units can be specified. All units must be specified in capitals and enclosed in quotes.</p> <p>Valid (and self-evident) unit specifiers are  "K(T48)"  "K(T68)"  "K(T90)"  "DEGC(T48)" "DEGC(T68)"  "DEGC(T90)"  "DEGF(T48)" "DEGF(T68)"  "DEGF(T90)"</p>