SGX documentation

Integrative Observational Platforms
Applied Physics Laboratory
University of Washington

v 1.0
September 2024

Contents

Chapter 1

Introduction

This is the complete user guide for Seaglider SGX autonomous buoyancy-driven underwater vehicle. It includes a reference manual; instructions for pilots responsible for the operation of Seaglider; and details about Seaglider hardware, software, and firmware. This guide is intended to be a living document that will be updated on an ongoing basis to reflect Seaglider developments.

This document is focused on SGX, the latest generation of Seaglider, but is also relevant for previous Seaglider models. Throughout the guide, *Seaglider* refers generically to the glider and *SG* refers to all Seaglider variations prior to SGX, which were built by University of Washington Seaglider Fabrication Center, iRobot, Kongsberg, and Hydroid/Huntington Ingalls Industries. Differences between SG and SGX are highlighted where relevant. The file formats described by this document pertain to firmware released by the APL-UW IOP group and may be different in places than older firmware versions.

1.1 Conventions and definitions

Parameters that are used to control the operations of Seaglider are given in uppercase **BOLD** font, and have leading **$** signs (e.g., **$T_DIVE**). The parameter reference manual is provided at `https://seaglider.pub/parms`. File names that are used in Seaglider command, control, or operations are given in lowercase **bold** font (e.g., **cmdfile**). Computer commands given at a prompt, especially when directly connected to the Seaglider, are given in *italic* font (e.g., *menu hw/batt/read)*.

This guide uses the shorthand *cc* (cubic centimeters) for cm^3.

Density (ρ) is defined as mass per unit volume ($\rho = M/V$), with typical 1000 m ocean values

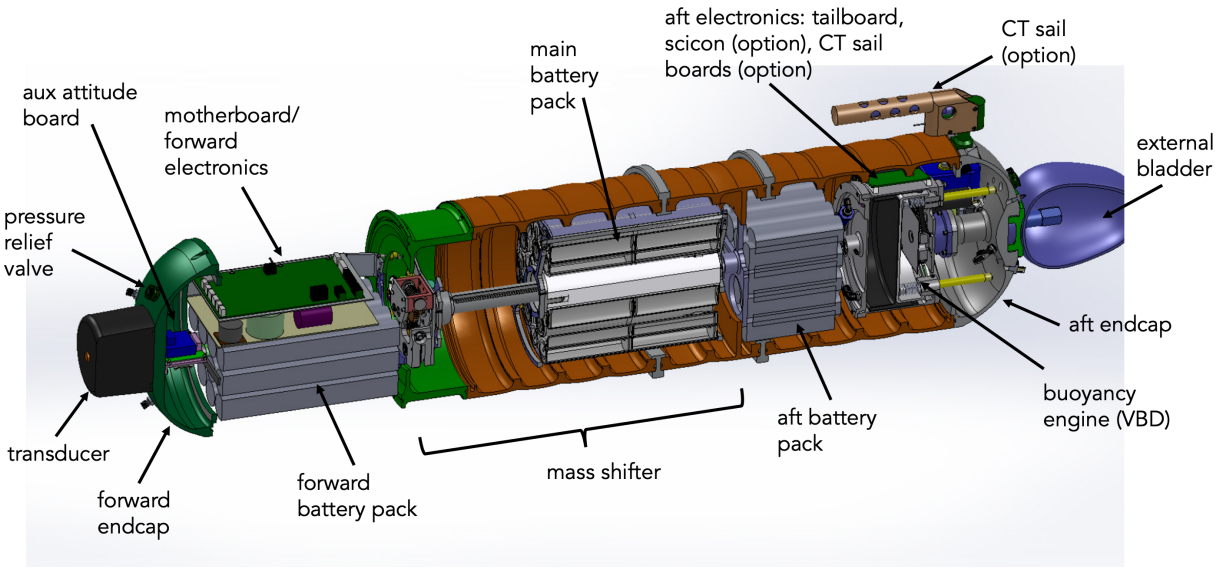

Figure 1.1: SGX cutaway drawing. Not shown components include fairings, rudder and wings, antenna and mast, cables, and hatch-mounted sensors.

of 1027.5 kg/m^3 (expressed in SI units) or 1.0275 g/cm^3 (expressed in cgs units, which are typically used in glider operations). Oceanographers use a shorthand notation for density, σ = ($\rho - 1000$) kg/m^3 – so the typical 1000 m density is 27.5 in σ units. This guide uses the unit σ_T, which means that the density is computed using the *in situ* temperature of the water.

1.2 Components of a Seaglider

1.2.1 Hull, fairings, and batteries

Seaglider has an anodized aluminum hull comprised of several sections that are fastened together and sealed with o-rings. From forward to aft, these sections are: forward endcap, forward battery/electronics hull, forward flange, mass shifter, aft flange, aft battery hull, and aft endcap. (In comparison, the SG hull was comprised of a two-piece forward electronics hull, mass shifter bulkhead and hull section, and aft endcap. The larger size of SGX compared to SG accommodates a larger forward battery pack and an additional aft pack.)

The isopycnal hull of the Seaglider is designed to have the same compressibility as seawater. Most other oceanographic equipment is contained in pressure hulls that maintain a fixed volume at all rated pressures. These pressure hulls acquire positive buoyancy as they are pressurized, which requires compensation (subtraction of displaced volume) to maintain a

prescribed buoyancy. That same compensation has to be recovered by pumping to achieve positive buoyancy. Seaglider's isopycnal hull avoids that need, as the pressure hull does not acquire positive buoyancy from the compression of the surrounding seawater. For dives to 1000 m, this results in about a 5–10% energy savings in the energy budget.

In total, SGX has three 15 V lithium primary battery packs (total mass 19.6 kg; 0.91 kg lithium content), which have an effective capacity of 575 Amp hr. (A 15 V SG has 12.0 kg of battery packs containing 0.56 g of lithium and an effective capacity of 350 Amp hr).

The exterior of the Seaglider consists of fiberglass fairings that add buoyancy, reduce drag, and protect the vehicle. Two removable hatch covers enable access to the connectors on the aft endcap. The glider has a rudder and set of wings that can be easily removed for transit.

1.2.2 Variable buoyancy device (VBD)

The VBD is the buoyancy engine of the Seaglider, located in the aft endcap. It consists of an internal oil reservoir (a rolling diaphragm within the pressure hull; also called a bellofram), an external oil bladder (outside the pressure hull but inside the fairing), and a pumping system to pump low-viscosity hydraulic oil between the internal and external reservoir/bladder. Pumping oil from the internal to external reservoirs changes the glider's volume without changing its mass, thereby changing its density and its buoyancy in seawater. This is the means by which the glider profiles vertically in the water column.

The volume of the internal reservoir is monitored by two linear potentiometers, enabling accurate quantification of the amount of oil moving between the internal reservoir and the bladder. Hydraulic oil is fed into a low-pressure boost pump (note that some older SGs use a high-pressure boost pump, which necessitates a different VBD plumbing configuration) and then to a high-pressure hydraulic axial piston pump (referred to as the main pump): in the modern VBD configuration, the boost pump provides the pressure needed to supply the main pump and is only used when the glider is at depth and trying to ascend. A magnetically latching "Skinner" solenoid valve controls the quantity of oil that is pumped. Three check valves (1 to 5 psi) control the direction and rate of flow within the VBD. A schematic of the hydraulic system is shown in Figure 1.2 .

1.2.3 Mass shifter

Seaglider moves a mass inside its pressure hull to change its pitch and roll attitude. The mass consists of a large, axially asymmetric, battery pack with an 1100 g brass weight mounted on its bottom face. This battery and weight is mounted in an assembly called the

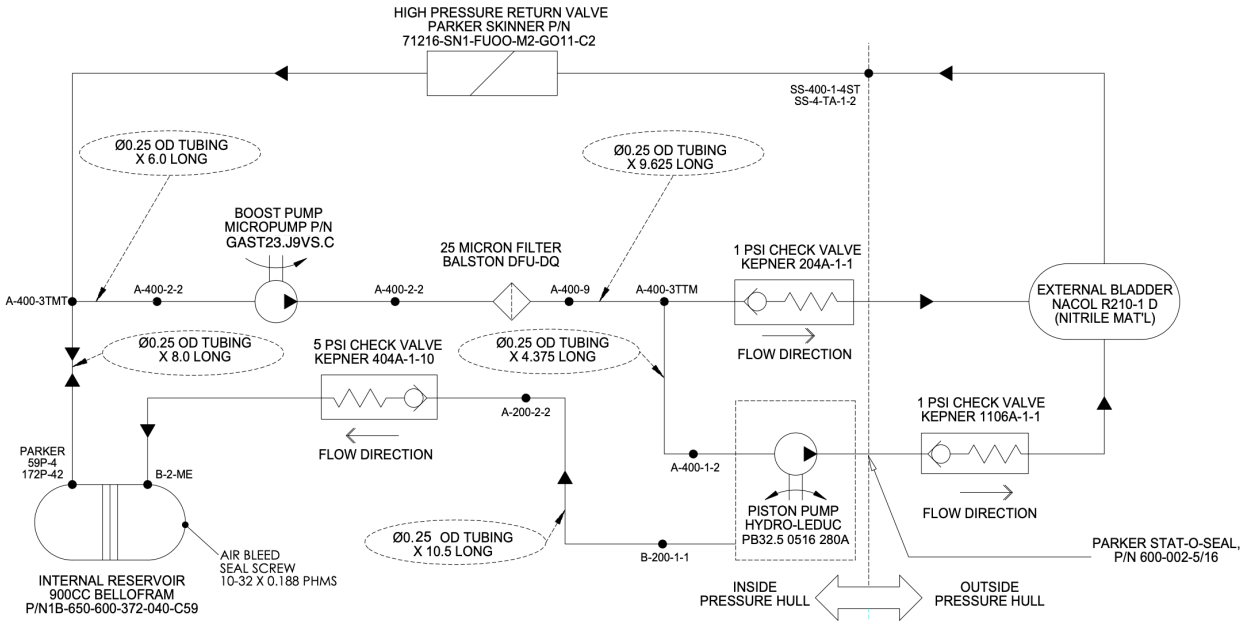

Figure 1.2: Hydraulic diagram for Seaglider VBD system. This configuration is standard on SGX; older SGs may have a different configuration.

mass shifter. The battery pack is translated forward and aft to effect changes in vehicle pitch. It is rotated within the pressure hull, and the axial asymmetry effects vehicle roll when the main battery pack is rolled from side to side.

The mass shifter in SGX has a new pitch-roll arrangement compared to SG in order to reduce backlash and to provide more precise control of attitude. The maximum roll is 80° for SGX compared to 40° for SG.

1.2.4 Electronics

The core components of the Seaglider electronics are the mainboard, mounted in the forward section of the vehicle, and the taiboard, mounted on the aft endcap. The tailboard serves as a junction point for signals and power from the mainboard to aft mounted sensors and systems, including the aft battery. Mainboard and tailboard are connected by three ribbon cables that run most of the length of the vehicle.

Core peripherals connected to the mainboard include the auxiliary attitude board for compass and pressure sensing, Iridium-GPS board, and Applied Acoustic Engineering (AAE) transponder/altimeter board. The aux attitude board is mounted to the forward endcap and provides a 3-axis magnetometer + 3-axis accelerometer + 24-bit ADC for the pressure sensor. The Iridium-GPS board mounts an Iridium 9523 modem module for satellite com-

munications as well as a uBlox GPS module and an RF relay to switch the single element Seaglider antenna between GPS and Iridium systems as needed. The Iridium-GPS board is mounted to the aft endcap and is connected to the mainboard via its own dedicated ribbon cable.

See Appendix A for a detailed description of the mainboard and its connections.

1.2.5 Sensors

Seaglider is able to support a wide range of sensors. At minimum, a conductivity-temperature (CT) sensor and a pressure sensor are required. Additional commonly flown sensors include optical backscatter and fluorescence (e.g., WETLabs triplet), dissolved oxygen (e.g., Aanderaa 4831 optode), irradiance (e.g., Satlantic/Sea-Bird OCR-504), passive acoustics, and ADCP (Nortek Signature1000). Members of the Seaglider community have developed integrations for numerous additional sensors.

Most sensors are mounted to the removable hatch covers on the aft end of the vehicle and are cabled to MCBH bulkhead connectors on the aft endcap, which are wired either directly to the tailboard or to the optional science controller (scicon). Exceptions include the Seabird CT sail (mounted directly to the aft endcap), the Kistler pressure sensor (mounted on the forward endcap on SGX) and the transducer (mounted on the forward endcap).

Chapter 2

Principals of Seaglider control

2.1 Forces controlling glider motion

Seaglider's flight is controlled by systems that change pitch, roll, and buoyancy. Pitch is controlled to obtain upward pitches for climbing and downward pitches for diving and for exposing the antenna at the surface. Roll is controlled to effect bank angles, which cause turns. Pitch and roll are controlled by altering the vehicle's center of mass. Buoyancy is controlled by changing the displaced volume of the vehicle. Seaglider is designed to operate within a few hundred *cc*'s of neutral buoyancy over a seawater density range of about 7 σ_T units for SGX (10 σ_T units for SG).

2.1.1 Static, dynamic, and environmental forces

Gravity

Seaglider achieves static trim by addition of ballast (lead weights) inside the fairings. The position and amount of lead is determined by mission and trim requirements. Sensor payload can also change the ballast.

Seagliders are delivered with ballasting to the user's specification: typically, for a target of -250 cc thrust at 27.5 σ_T. Different ballast may be needed in regions with strong currents or strongly atypical density (e.g., Arctic Ocean or Bay of Bengal, which have very fresh surface layers; Mediterranean and adjoining seas, which are very salty). This is addressed in Section 4.2.4.

Buoyancy

Buoyancy is the unbalanced (positive) upward force on a submerged object arising from the vertical pressure gradient. Submerged objects can alter their buoyancy by changing their density, either by changing their mass or volume. Gliders are fixed-mass, variable-volume devices: they change their buoyancy by changing their displaced volume while keeping their total mass fixed. This is achieved by the VBD, which moves hydraulic oil between a reservoir inside a pressure hull and an external bladder.

> Example 2.1: Suppose a SGX has a mass of M = 72,676 g. In seawater of density 1.0275 g/cm^3, what must the vehicle's volume be for it to be neutrally buoyant at that density? $V = M/\rho$, hence V = 70730.9 cm^3.

> Example 2.2: How much must a glider's volume change to compensate for a density change of 1 σ_T unit? Let $\Delta V = V_2 - V_1$, and let M and V_1 be as given in Example 2.1. Recall that 1 σ_T unit = 1 kg/m^3, or 0.001 g/cm^3, i.e., $\Delta\rho = 0.001 g/cm^3$. Some algebra leads us to the following: $\Delta V = \Delta\rho V_1^2/(M - \Delta\rho V_1)$. Using the values of M and V_1 from Example 2.1, ΔV = 68.9cm^3. This result leads to the following:
>
> **Rule of Thumb**: In SGX it takes about 70 cm^3 of volume change to compensate for 1 σ_T of density change. For SG, it's about 50 cm^3 of volume change per σ_T.

Seaglider uses the variable buoyancy device (VBD) to change its buoyancy. This system acts as described above to change the Seaglider's displacement without changing its mass. Seaglider has about 800 cc of volume change available from its VBD. Here is an example of how that available VBD range is partitioned on a typical mission:

Total VBD available	800 *cc*
Positive buoyancy required to expose antenna	(150 *cc*)
VBD remaining	650 *cc*
Negative thrust in densest water	(250 *cc*)
VBD remaining	400 *cc*
equivalent σ_T units of stratification (SGX)	5.5
equivalent σ_T units of stratification (SG)	8

In this fairly typical example, an SGX could fly normally with 250 cc of negative displacement in the densest water, fully expose the GPS/Iridium (GPSI) antenna, and accommodate 5.5 σ_T units of density difference between the densest water and lightest water.

Lift, drag, and hydrodynamic model

Seaglider gets lift from its body and its wings, which convert the vertical force provided by the VBD into forward motion. Some additional lift comes from the vertical stabilizer while banked (executing turns).

In the Seaglider flight model, drag is partitioned into induced drag and everything else (skin friction, form drag, etc.).

A hydrodynamic model for Seaglider is used by pilots to help with buoyancy trim and is the mechanism by which depth-averaged currents are inferred for a Seaglider dive. The model has three parameters: lift, drag, and induced drag (traditionally called *a*, *b*, and *c*). For our purposes, it is convenient to think of the hydrodynamic model as a black box that produces estimates of the Seaglider's 3-dimensional velocity ($\mathbf{v}_{model}$) as a function of computed buoyancy and observed pitch. $\mathbf{v}_{model}$ can be resolved into horizontal and vertical components. The horizontal component, u_{model}, can be used with the observed compass headings throughout a dive to determine a dead-reckoned glider track through the water. This results in a predicted surfacing position, based on the GPS-determined dive starting point. The difference between this predicted surfacing position and the actual GPS-determined surfacing position is what provides the estimate for depth-averaged current. Similarly, the vertical velocity w_{model} can be compared with $w_{observed} = \frac{dz}{dt}$ (change of glider's depth over time) to adjust the VBD trim and then to estimate vertical velocities in the water column. For details on the Seaglider hydrodynamic model, consult Eriksen et al. (2001).

Stratification

Stratification describes the static stability of the ocean, with denser water below lighter water. Strong stratification means a large change in density between two depths; weak stratification is a small change in density between two depths.

> Example 2.3: Off the Washington coast in winter, the surface water has density ρ = 1.0245 g/cm^3, while the water at 1000 m depth has density ρ = 1.0275 g/cm^3, a difference of 3 σ_T units. By our rule of thumb for SGX (section 2.1) this implies that 210 *cc* of volume change will be required to maintain constant buoyancy forcing through the stratification from 1000 m to the surface (150 *cc* for SG).

Currents

Depth-averaged current over the course of a Seaglider dive influences the distance over the ground covered by the glider. The depth-averaged aspect of the current is important

– the Seaglider can make progress towards a waypoint even in the presence of strong adverse surface currents by diving through deeper waters with more favorable currents. The maximum deep-water depth-averaged current that Seaglider can stem is around 40 cm/s, or 0.8 kts. That is the practical limit and requires driving the Seaglider as hard as it can be driven (in the buoyancy sense). These dives tend to be done with large negative thrust on the dive (-350 *cc*), and vertical velocities of about 18 cm/s. The dives take about three hours between surfacings, or about eight dives per day. The VBD pumps from minimum to maximum on each dive. This regime uses energy at about ten times the rate of a typical dive in a region of weak currents, where the dives are eight hours long (three dives per day) and the VBD stays within about half its full range.

Vertical shear in the currents induces turning moments on the Seaglider body. Large vertical velocities (upwelling or downwelling) can introduce large control changes in buoyancy, and in some cases, cause dives to truncate or abort prematurely.

2.1.2 Control of Seaglider flight

Seaglider's flight is controlled by specification of the positions of the three systems that control pitch, roll and buoyancy. All positions are encoded by linear potentiometers, digitized by 4096-count analog-to-digital (A/D or AD) converters that run from 0 to 4095 counts. Physically attainable limits (referred to as hardware limits) for each system are determined empirically at the time of assembly. A safety margin is added to these physical limits to arrive at a software limit, which is the position (in A/D counts) beyond which the Seaglider operating software will not command that particular system. Associated with each system are the following:

1. A center position, which is intended to be the vehicle neutral for that system, in a particular environment: **$C_PITCH**, **$C_ROLL**, and **$C_VBD**
2. A factor that converts A/D counts to physical displacement, which is based on the mechanical design: **$PITCH_CNV**, **$ROLL_CNV**, and **$VBD_CNV**
3. A gain that relates movement of each system to the effect it has on the Seaglider

Pitch

Pitch is controlled by moving the main battery pack forward and aft along the longitudinal axis of the Seaglider. The motion is accomplished by an electric motor that is geared to drive a worm-gear in such a way that 1 A/D count equals 0.00413 cm of battery mass travel for SGX (**$PITCH_CNV**) (0.003130 cm/count for SG). Seagliders typically respond to movement of the battery pack in the longitudinal axis by pitching

about 15-20° per centimeter of mass travel. This **$PITCH_GAIN** parameter is tuned by the pilot, as it is dependent on the particular sensor suite and trim ballast installed on each Seaglider.

The following are typical pitch ranges and values for SGX:

pitch mass position (glider position)	hardware limit (A/D counts)	software limit (A/D counts)
full forward (pitched down)	115	265 (**$PITCH_MIN**)*
full aft (pitched up)	3700	3550 (**$PITCH_MAX**)*
centered (flat)		2600 (**$C_PITCH**)

*Although **$PITCH_MIN** and **$PITCH_MAX** can be set as software parameters, these are not adjusted while piloting.

Note that A/D counts are always positive, but displacement may be positive or negative relative to a given **$C_PITCH**. Pitch is usually trimmed so as to have about 70% of the pitch travel available for pitching down (pitch mass forward of **$C_PITCH**), and 30% available for pitching up (pitch mass aft of **$C_PITCH)**. This is to ensure a good surface position that is sufficiently pitched down to fully expose the antenna.

Example 2.1.2.5. How many centimeters is the SGX battery pack forward of its center position when the Seaglider is in its surface (fully pitched-down) position? By the values in the table above, the pitch position will be 115 A/D counts when fully forward. Given a **$C_PITCH** of 2600 and a **$PITCH_CNV** of 0.00413 cm/AD count, we have that the pitch position = (115 counts - 2600 counts)*(0.00413 cm/count) = -10.3 cm, or 10.3 cm forward of the pitch center.

Example 2.1.2.6. How many A/D counts will cause 5° of SGX pitch? Assume a **$PITCH_GAIN** of 16°/centimeter of mass travel. Then number of A/D counts that corresponds to 5° of vehicle pitch is 5°· (1 cm/16°) · 1/0.00413 counts/cm ≈ 76 A/D counts.

Example 2.1.2.7. Suppose the pilot made a trim bias correction of **$C_PITCH** to 2375. What was the presumed pitch bias (in degrees)? The previous **$C_PITCH** was 2600 as given in the table above. Then the pitch bias is computed as follows, (2600 A/D counts - 2375 A/D counts) · (0.00413 cm/counts) ·16°/cm = 0.93°. The Seaglider was pitched approximately 1° down.

Roll

Roll is controlled by rotating the main battery pack inside the hull. The pack is axially asymmetric and weighted on its ventral face. An electric motor and gear train rotate the mass such that 1

A/D count is equivalent to 0.054945 degrees of battery mass rotation for SGX (**$ROLL_CNV**; 0.028270 for SG). Seagliders typically respond to the rotation of the battery pack by rolling about 1/4° for every 1° of battery pack rotation (1/2° for SG); this value depends on the amount and distribution of trim lead.

The control strategy is to roll the main battery pack a specified amount (80° for SGX, 40° for SG; **$ROLL_DEG**) in the appropriate direction when a turn is indicated, then roll back to neutral (center) when the desired heading is reached. There is no gain setting for roll as the battery is typically moved as far as it will go to one side or the other during each roll move. Note that the Seaglider turns in the opposite sense from its bank angle on the dive (opposite from upright airplane control), and in the same sense as its bank angle on the climb (same as upright airplane control). Here are some typical roll ranges and values for SGX. Different roll centers are used for the dive and the climb because various asymmetries in form result in different roll trim on dives and climbs.

glider position	hardware limit (A/D counts)	software limit (A/D counts)
full roll to port	50	200 (**$ROLL_MIN**)*
full roll to starboard	3960	3810 (**$ROLL_MAX**)*
centered during dive		2150 **$C_ROLL_DIVE**
centered during climb		2250 **$C_ROLL_CLIMB**

*Although **$ROLL_MIN** and **$ROLL_MAX** can be set as software parameters, these are not adjusted while piloting.

Buoyancy

The Seaglider's VBD controls buoyancy by (a) pumping oil from the internal reservoir into the external bladder to increase the glider's volume (and hence make it more buoyant) and (b) allowing oil to bleed from the external bladder into the internal reservoir to decrease volume (making the glider less buoyant). As the VBD pumps and bleeds, two linear potentiometers report the position of the piston driving the rolling diaphragm that controls the volume of the internal reservoir: larger A/D counts measured by the linear potentiometers imply a larger internal reservoir volume (i.e., a a smaller volume of oil in the external bladder). Small wobbles in the piston cause differences (up to a few hundred A/D counts) between the two linear potentiometer readings, so the average of the two readings is taken. The geometry of the system results in -0.2453 cc of oil moved per A/D count (**$VBD_CNV**; same for SGX and SG).

The position of the VBD at which the glider is neutrally buoyant is designated **$C_VBD**, and is set relative to the densest water to be encountered on a mission (i.e., the water encountered during the deepest part of the dive).

Changing VBD parameters enable the pilot to control the glider's vertical velocity and glide slope.

These are typical VBD ranges and values for SGX:

	hardware limit (A/D counts)	software limit (A/D counts)	volume** (cc)
V_{max}*	105	370 (**$VBD_MIN**)	600
V_{min}*	4060	3960 (**$VBD_MAX**)	-260
Range			860
$C_VBD		2900	

* V_{max} is the maximum displaced volume of the Seaglider and V_{min} is the minimum displaced volume of the Seaglider.
** Volume is the volume of movable oil (i.e., oil that can be pumped between the internal/external bladders) relative to **$C_VBD**. In this example, the range of movable oil (i.e., the range of the glider's volume) is 860 cc.

2.2 Features of glider control

2.2.1 Canonical dive

The Seaglider performs its mission by repeating a canonical dive until either commanded to stop or until an abort condition is reached. Numerous aspects of the canonical dive are under the control of the pilot, through an extensive set of parameters. A few are indicated in the canonical dive that is shown schematically in Figure 2.1. Many more are not shown; consult the Seaglider parameters manual (`https://seaglider.pub/parms`).

2.2.2 Control design

The Seaglider flight control scheme has two guiding principles: maintain constant vertical velocity and minimize the total energy expenditure during a dive. The Seaglider samples its sensors evenly in time, so constant vertical velocity implies the samples are equally spaced in depth. Sample intervals are specified by the pilot either through the **science** file or, for gliders with a science controller (scicon), through the **scicon.sch** file (see scicon manual for details: `https://iop.apl.washington.edu/iopsg/`). Sample intervals may vary by pilot-specified depth bands but are uniform within each specified depth band.

The desired vertical velocity is not specified directly, but is calculated from parameters that describe the target depth of a dive (**$D_TGT**, specified in meters) and the time to complete a dive (**$T_DIVE** (in minutes, time from surface-to-surface discounting pumping time at the bottom of the dive). So, the desired vertical velocity, w_d = (2 * **$D_TGT** * 100 cm/m) / **$T_DIVE** * 60 s/min) cm/s. A typical value for w_d is 10 cm/s. Some simple algebra gives a convenient ratio of **$D_TGT** to **$T_DIVE** = 3: (\$**D_TGT**/\$**T_DIVE**) = $(10cm/s * 60s/min)/(2 * 100cm/m) = 3m/min$. So, to achieve a w_d of 10 cm/s, choose **$D_TGT** that is 3 times the value of **$T_DIVE**. Our typical first shallow test dive uses **$D_TGT** = 45 m and **$T_DIVE** = 15 min. A full-depth dive might use **$D_TGT** = 990

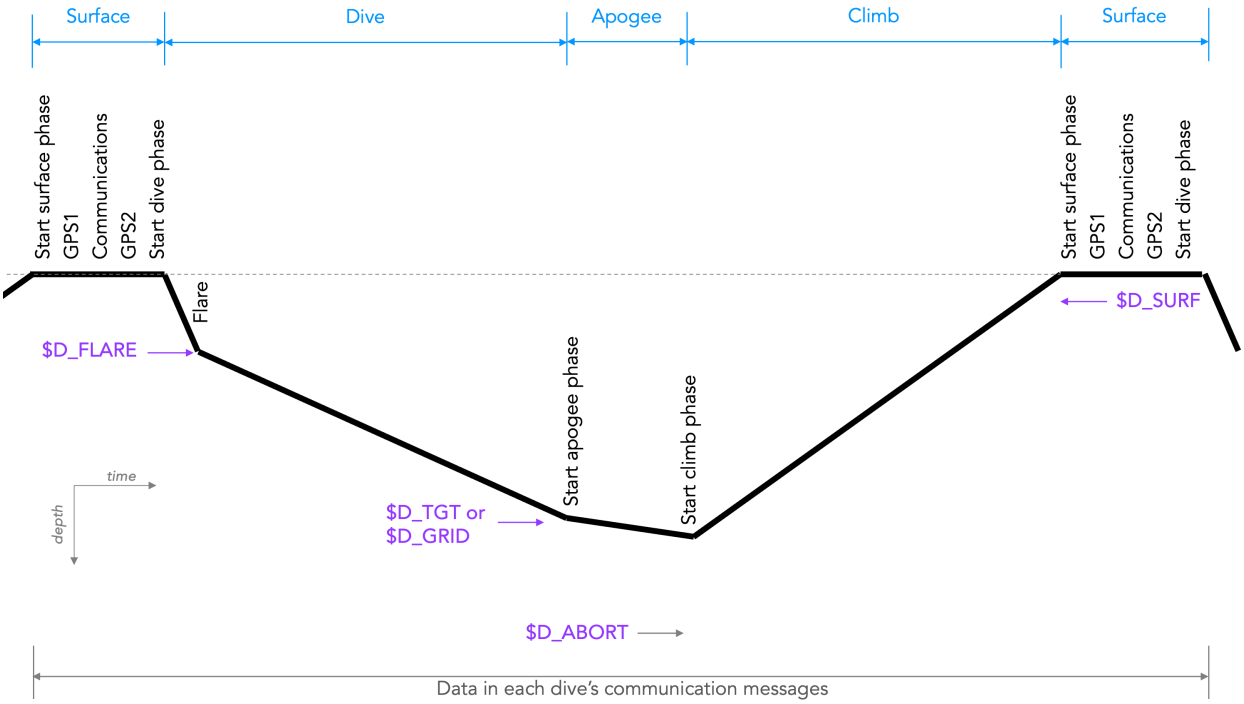

Figure 2.1: Canonical dive: schematic of a Seaglider run, with run phases indicated by the intervals at the top of the figure and the profile data boundaries indicated by the interval at the bottom of the figure. The figure is not to scale in either dimension.

m and **$T_DIVE** = 330 min as a starting place (though the pilot may vary **$T_DIVE** to achieve certain goals.

The buoyancy and pitch used on any individual dive is chosen by the Seaglider operating software to achieve the best results on that dive. The choices are bounded by parameters **$MAX_BUOY**, the maximum negative buoyancy allowed on a dive, and **$GLIDE_SLOPE**, the maximum glide slope allowed on the dive. The choices are also bounded by physical limits, neutral buoyancy (some negative buoyancy is needed to glide) and the stall angle. The software has to choose a buoyancy value between 0 (neutral) and **$MAX_BUOY**, and a desired pitch angle between the stall angle and **$GLIDE_SLOPE**. The choice is determined by the distance to the next waypoint. The pitch angle is chosen to achieve the desired horizontal distance: maximum if the waypoint is distant, minimum if the waypoint is close, or the exact distance, if possible. Once the pitch angle is chosen, the buoyancy is chosen to achieve the desired vertical velocity in the densest water (deepest depth).
The main energy draw on Seaglider is pumping hydraulic oil from the internal reservoir to the external bladder at depth, where the pump has to overcome the seawater pressure acting on the bladder. The pump consumes about 50% of the energy budget of Seaglider. Control during flight is generally designed to minimize the total amount of pumping required on a dive. In particular, no bleeding is allowed on descent (dive) to maintain the desired vertical velocity. Pumping as necessary is allowed on the climb to maintain the desired vertical velocity. Pitch is essentially fixed at all phases of the operation, with the exception of slight pitch maneuvers on the climb to compensate for the changes in mass distribution and buoyancy due to pumping oil from the internal reservoir into the bladder. Details of the control scheme are given in the description of the run phases below.

2.2.3 Run phases

The phases of a Seaglider autonomous run are described below. The launch and recovery phases are generally only performed at the beginning and end of the mission. The surface, dive, apogee and climb phases are meant to be repeated sequentially, once each profile, until the end of the mission. The surface phase is where the GPS positions are acquired, where communication is attempted, and where the navigation calculations for the next dive are made. Various depth, time and functional triggers exist to cause the Seaglider to move from one phase to the next.

Data acquisition is done in the dive, apogee and climb phases of a run. During these phases, the glider continuously collects data from the scientific instrumentation at the rate set specified in the **science** or **scicon.sch** file. Although other actions are performed during these phases, the data collection process is never interrupted.

Another periodic action performed during the profile phases (dive, apogee and climb) is guidance and control (G&C). G&C operations occur at intervals defined in the **science** file (for gliders with scicon, the **science** file primarily controls G&C intervals). Three operations occur during G&C, if necessary, in this order: pitch adjustment, VBD adjustment, and roll adjustment. Only one of

these operations can occur at a given time.

When G&C operations occur, the Seaglider is said to be in active G&C mode. When G&C corrections are not being made, the Seaglider is said to be in passive G&C mode. These definitions of active and passive modes refer to G&C operations only; they do not apply to data acquisition intervals or activities. The Seaglider is acquiring data during all profile phases, whether in active or passive G&C mode.

In passive G&C mode, the Seaglider processor enters a low-power sleep state between data acquisition points. The Seaglider flies in the state specified in the previous active G&C mode.

Launch

The launch phase begins when the field operator has performed a self-test if indicated, initiated the Sea Launch procedure, and all launch dialogue has completed. (Details of the launch procedure will be given in Chapter 4, Mission Execution.) The Seaglider is in its surface position (rolled to neutral, pitched fully forward, and pumped to **$SM_CC** – typically maximum VBD for launch), and enters a normal surface phase: acquires GPS1 (i.e., the first GPS fix) and initiates a communication session via Iridium satellite telephone.

Surface

The surface phase begins at the end of climb-phase data acquisition (or after launch) and consists of the following steps:

1. **Surface maneuver:** The surface position of the Seaglider is pitched fully forward (to the software limit), rolled to neutral (**$C_ROLL_CLIMB**), and pumped to VBD = **$SM_CC**. Note that if the Seaglider surfaces with VBD > **$SM_CC,** no bleeding is done to force VBD = **$SM_CC**. There are several ways to enter the surface maneuver. The Seaglider is in the surface position at launch, after normal completion of a dive (reached **$D_SURF**), in recovery phase, or after **$T_MISSION** minutes have elapsed from the start of the dive without achieving **$D_SURF** in climb phase. The first test in surface phase is to see if the Seaglider depth is less than **$D_SURF**. If so, the Seaglider pitches fully forward and pumps to **$SM_CC**. If not, the Seaglider first pumps VBD to its maximum value, and checks the depth again. If the depth is less than **$D_SURF**, the Seaglider moves the pitch mass to its full forward position. This behavior is designed to try to get the Seaglider to the surface in the event of a **$T_MISSION** timeout.

2. **GPS1:** Once the desired surface position is attained, GPS position 1 (**$GPS1**) is acquired. The GPS receiver is turned on and left on until a satisfactory position is acquired or until **$T_GPS** minutes have elapsed. Once an initial position is acquired, the Seaglider waits an additional **$N_GPS** samples for a GPS position with an HDOP (horizontal dilution of

precision) < 2.0. If one occurs, acquisition stops and that position is accepted. If one has not occurred in **$N_GPS** samples, the last position is accepted.

3. **Communications:** Wireless communication via Iridium begins following acquisition (or timeout) of **$GPS1**. The Seaglider powers up the Iridium phone, waits a specified time for registration with the Iridium system, then attempts a data call to the basestation. If the call is successful, the Seaglider logs into the basestation and transfers files: data, log and engineering files from the Seaglider to the basestation, and command, control, diagnostic and special purpose files from the basestation to the Seaglider. Details of these files and their effects are given in Chapter 3, Command and Control. If all file transfers were not accomplished, the Seaglider waits **$CALL_WAIT** seconds and tries again. It tries up to **$CALL_TRIES** times and, if not successful, continues with the surface phase, marking files as appropriate for later transfer, and incrementing the **$N_NOCOMM** parameter.
4. **Measure surface depth and angle:** After the communications session, the Seaglider computes the average of 10 pressure readings and then the average of 10 pitch angles to obtain a measurement of the Seaglider's surface position. These values are written into the log file for the next dive.
5. **GPS2:** After the surface pressure and pitch angle average is completed, a second GPS fix, **$GPS2**, is acquired. This fix is the most recent position of the glider prior to diving.
6. **Navigation and flight calculations:** Finally are the calculations of the parameters which determine the glider flight path of the next profile: buoyancy, pitch angle, and heading. These computations include the Kalman filter, if enabled (typically not enabled), and the digital bathymetry table lookup, if enabled (typically enabled). Upon completion of the calculations, the surface phase is completed and a new dive phase (and new profile) is started.

Dive

The dive phase begins upon completion of the navigation and flight calculations that conclude the surface phase. Initially, pitch is in the full forward position and the VBD volume is equal to the endpoint of the surface maneuver. At the start of the dive phase, a VBD adjustment (bleed) only is executed during the first G&C operation to get the Seaglider as fast and deep as possible (recall that pitch is still in the maximum forward position). When the Seaglider reaches a prescribed depth, **$D_FLARE**, it goes into a regular G&C operation (pitch, VBD, roll) to move to the desired pitch, VBD position and course computed for the profile.

If the glider speed is too fast compared to the desired vertical velocity on the dive section of the profile (glider too heavy), VBD pumping is not allowed to correct the speed error. This is an energy conservation consideration. As the Seaglider descends into denser water, it will become less negatively buoyant and will slow down. If corrective pumping were allowed on the dive, it is possible that additional bleeding would be required to compensate as the Seaglider reached

denser water. That would then mean more pumping to eventually reach the buoyancy endpoint of the surface maneuver. Excess speed is tolerated on the dive to help minimize total energy expenditure on the profile.

In the dive phase, the Seaglider turns to starboard by banking to port (opposite to upright aircraft flight).

Apogee

When the target depth is reached, the Seaglider enters the apogee phase. The apogee phase is a two-G&C-cycle procedure to smoothly fly from the dive phase to the climb phase without stalling. During the first G&C cycle of this phase, the glider is pitched to an intermediate angle, **$APOGEE_PITCH**, rolled to neutral, and the VBD is pumped to 0 cc. The course adjustment and passive G&C mode are skipped. A second G&C cycle is then executed and the glider is first pitched, then VBD is pumped, to the inverse positions of the dive (pitch = -pitch, VBD = -VBD).

Data sampling continues throughout the apogee phase.

Climb

The climb phase begins at the completion of the second G&C cycle of the apogee phase. The Seaglider is positively buoyant and pitched up, headed for the surface at the same target vertical rate as achieved on the dive phase of the profile. As in the dive phase, data acquisition and G&C continue at the intervals specified in the **science** file.

If the glider speed is too fast on the climb section of the profile (too light), VBD bleeding is not allowed to correct the speed error. This is an energy conservation consideration. There are two reasons: (1) any oil that is bled will need to be pumped again during the surface maneuver; (2) as the glider climbs it will enter less dense water, hence becoming less positively buoyant and slowing down. VBD pumping operations are allowed in the case of the glider being too heavy and slowing down. The **$MAX_BUOY** restriction does not apply to the climb phase. This will not effect the amount of energy used during the profile since the oil will be pumped anyway during the surface maneuver.

In the climb phase, the Seaglider turns to starboard by banking to starboard (as in aircraft flight).

When the Seaglider reaches the depth **$D_SURF**, it enters the passive G&C mode and continues to acquire scientific data at the specified interval for a time equal to **$D_SURF**/($w_{observed}$ m/s) or for a maximum of 50 data points, whichever is shorter. After this period of data acquisition, the Seaglider enters the surface phase.

Recovery

The recovery phase is entered either by command of the pilot (when it is necessary or desirable to keep the Seaglider at the surface) or by an error condition detected by the Seaglider operating software. In the recovery phase, the Seaglider stays on the surface and acquires a series of GPS fixes that are sent to the basestation so that the Seaglider can be recovered.

In recovery, the Seaglider enters a loop of obtaining a GPS fix and communicating every **$T_RSLEEP** minutes. In practice there are about two minutes of overhead in this process, so the actual time between phone calls is closer to **$T_RSLEEP +** 2 minutes. This recovery loop is exited by sending a **$RESUME** directive in the **cmdfile**. The Seaglider will then continue diving.

Chapter 3

Command and Control

3.1 Seaglider

3.1.1 Serial communications

Direct communications with Seaglider are made by connecting one end of the supplied serial communications cable to the serial port on the Seaglider, and the other to a serial port on a computer. Any reasonable terminal emulation program should be able to talk to the glider. Port settings for SGX are 115200 baud, 8N1, and no hardware handshaking. (SG with RevB motherboards use 9600 baud.)

3.1.2 Power on/off

SGX is powered on and off by use of an external magnet, usually mounted at the end of a wand. The magnet is held in the appropriate location (marked on the starboard side just aft of the nose – same location for on and off) for 0.5 seconds. Once the Seaglider is turned on and the serial communication is established, a short start-up banner should appear on the terminal, along with a request to enter a carriage return (CR) within one minute. Common reasons for not being able to connect to the glider include: incorrect baud rate, bad RS232-USB adapter (if using), cable or adapter not seated correctly, magnet not strong enough, not holding magnet in place long enough, holding magnet in place too long (glider shuts back off).

3.1.3 Menus

Interaction with the Seaglider while directly connected is by a text-based multi-level menu system. Menu items are specified either by number or by keyword match. Item numbers and keywords are shown for each entry in the menu system. For example,

```
1 [param ] Parameters and configuration
```

is the first line of the main menu, which means that the Parameters and configuration menu can be accessed by entering *1* or *param*. The menu system is simple to navigate, and multi-level jumps are permitted. For example, from the toplevel menu you can jump directly to the menu item to move the pitch to a specified AD position with *hw/pitch/ad*. This is equivalent to typing *2* (hardware menu), then *1* (pitch menu), then *2* (move to position (AD counts)). You can also specify the desired position via command line arguments *hw/pitch/ad pos=2300*.

The complete menu tree and allowable command line arguments are given in Appendix B.

3.1.4 Extended PicoDOS

Rev E implements its own Extended PicoDOS subsystem. You can enter epdos at any time by typing "pdos" at the menu prompt when connected to the glider. Individual epdos commands can also be executed at the menu prompt by prepending with *!* (e.g., *!dir*).

Pilots can execute epdos commands by uploading a **pdoscmds.bat** file when the Seaglider is operating autonomously. The **pdoscmds.bat** file is created by the pilot only when needed. It contains one or more lines, each of which is a command that the glider runs in order. If a **pdoscmds.bat** file is on the basestation, the glider picks it up near the beginning of its next call, renames the file on the basestation **pdoscmds.bat.nnn.ccc** (where *nnn* is the dive number and *ccc* is the call cycle), and immediately runs the specified commands. Any output from the action will be printed to a file called **pGGGdddd.ccc.pdos**.

Appendix C contains a list of epdos commands for Seaglider.

3.2 Basestation

3.2.1 Function

The Seaglider basestation is the shoreside computer end of the Seaglider system. It has three main responsibilities: (1) It supports a modem (or modems) and dial-up users (Seagliders), (2) it handles one side of the modem-to-modem file transfer protocol that moves files to and from the Seaglider, and (3) it does the data processing necessary to produce scientific and engineering data profiles and to perform simple error-detection and notification.

3.2.2 Configuration

The Seaglider basestation runs on a Linux platform. Various distributions have been used successfully. Core components of the basestation software are the RUDICS daemon for connection

between the Iridium ground-station and the networked basestation, processing scripts for converting and managing data uploaded by the glider, and the visualization and plotting services to enable piloting and data presentation.

Gliders typically connect to the basestation via RUDICS, a telnet-like service, though it is possible to connect via dial-up modem. In either case, Seagliders log in as normal users, and send and receive files from their home directory. Seaglider pilots need access and write permission in the gliders' home directories in order to modify command and control files. Scripts are executed at login and logout that control and log various aspects of the basestation transactions.

3.2.3 Files

The Seaglider pilot interacts with four primary files on the basestation to command and control Seaglider: **cmdfile**, **targets**, **science**, and **pdoscmds.bat**. The basestation utilities *cmdedit, targedit,* and *sciedit* are used to modify **cmdfile**, **targets**, and **science**. The **pdoscmds.bat** file is created and modified by any text editor. Gliders with a scicon have additional files.

File formats are detailed in Appendix D.

Chapter 4

Mission Execution

4.1 Planning

Mission planning is an important part of Seaglider piloting. A basic understanding of Seaglider's operating envelope and its strengths and weaknesses is critical to planning effective science missions. The general idea is to go far by going slow: recall the square-law dependence of drag on velocity. The sections below give the operating limits of the Seaglider.

4.1.1 Environment

Stratification

As discussed in 2.1.1, the range of stratification in which a Seaglider can operate normally is constrained by the total amount of VBD change available and the amount of (negative) buoyancy required for the flight plan. The pilot (or scientist) should determine the likely range of densities to be encountered on a proposed mission, and determine if there is sufficient VBD change available to accommodate that range. Compromises can be made by reducing maximum operating depth (at the expense of duration) or by reducing thrust at apogee (at the expense of horizontal speed).

Currents

The maximum depth-averaged current that Seaglider can stem is 40 cm/s, or 0.8 knots. That performance requires ballasting for 350 cc of negative displacement, specifying vertical velocities of almost 20 cm/s, and diving to 1000 m. Dives last about three hours in that case, and total mission length is of order two months. Remember that it's the depth-averaged current that must be considered. Surface currents can be a problem, especially when doing shallow dives (see below). Plans for crossing strong currents, such as the Kuroshio or Gulf Stream, should be carefully

considered, and contain both return (upstream) plans and bail-out plans.

Bathymetry

The Seaglider is least efficient operating in shallow water and most efficient in deep (up to 1000 m) water. The practical shallow water limit is about 75 m. It is hard to make progress toward a waypoint in water shallower than that for three main reasons: turn radius, pump time, and surface time. Seaglider's turning radius (a few tens of meters at typical 25 cm/s horizontal speeds) is such that a significant portion of a shallow-water dive can be spent turning onto the desired course. Seaglider's pump is optimized for efficiency at pressures equivalent to 1000 m ocean depth, so its rate at shallow-water pressures (about 2cc/s) means that a significant portion of a shallow-water dive can be spent pumping. Finally, the time on the surface can be a significant percentage of the dive time, and if surface currents are adverse the Seaglider can easily lose as much distance toward a waypoint while on the surface as it gains on the dive. The UW's operating guidelines are to operate deeper than 200 m on offshore (deep water) missions and to try to stay deeper than 75 m on coastal or estuarine missions.

SGX and SG are rated to 1000 m depth. Deep-water target depth (**$D_TGT**) is typically 990 m to allow for the apogee maneuver and the discreteness of the sampling times.

Knowledge of the bathymetry of the operating area is important. Seaglider can read a digitized bathymetry map to determine how deep to dive, or can rely on the on-board altimeter to find the bottom and initiate the apogee maneuver appropriately. See section 4.2.3 for detail.

4.1.2 Science requirements

Track

Seaglider can cover about 20 km/day (12 nm/day) through the water in normal flight. It can station-keep within about a factor of two of the dive depth (e.g., 2 km on 1 km dives). The navigation system on Seaglider is waypoint-based, not track-based. The system decides on the most efficient way to reach the next waypoint, but does not attempt to stay on a given track. Track-based navigation can be approximated by using more waypoints along a desired track.

Sampling

Sensor sampling intervals are specified in the **science** file or the **scicon.sch** file for gliders with a scicon. The **science** or **scicon.sch** file also gives the ability to turn off sensors, or only energize them every n-th sample, in a given depth range (or ranges). Scicon provides additional, independent control over science sampling, as well as onboard processing of science sensors. See the scicon manual (`https://iop.apl.washington.edu/iopsg/`) for additional details.

4.1.3 Endurance

Total endurance is dependent on many factors, including depth of dive, specified vertical velocity, type of track, stratification, pump efficiency, sampling, and communications performance. The nominal mission duration for SGX is 12 months. Duration can be extended with periods of loitering (for instance, under ice or at depth).

4.2 Preparation

4.2.1 Refurbishment

This evolution is done in the laboratory between deployments. It requires a complete opening and disassembly of the Seaglider to the major sub-assembly level in order to replace batteries; inspect hydraulics, motors, and gear assemblies; change compact flash or SIM cards as necessary; remove sensors for calibration as necessary; and other operations as necessary. Preparations for deployment following refurbishment are basically the same as those for a new Seaglider.

4.2.2 Calibrations

Sensor calibrations should be performed before and after each deployment, as schedule and budget permit. Compass calibrations can be done in situ during missions and do not have to be performed between deployments.

4.2.3 Bathymetry maps

The Seaglider has the ability to read digital bathymetry maps of an operating area. These maps should be prepared and loaded on the Seaglider's compact flash prior to deployment. It is possible to upload new map files to the glider while underway, but it is easier to do it beforehand. Maps are loaded on the card as files named **bathymap.bbb**, where bbb is an integer number. When in use, a bathymap is loaded into memory. This imposes a practical limit of about 200 kB on each individual map, this multiple maps are needed cover a large area or to provide high resolution. Specify which map to use with **$USE_BATHY**.

4.2.4 Ballasting

This is the procedure for adding or subtracting lead weights to trim the Seaglider for the desired flight characteristics in a given density environment. The initial displaced volume determination is done in the saltwater test tank at the UW School of Oceanography. Further ballasting for particular environments and/or after the vehicle's mass has changed (e.g., from changing batteries for sensors) can be done using the "VBD regression" tool on Basestation3.

4.2.5 Hardware checkout and self-test

The hardware checkout procedure is done in the lab, perhaps prior to closing up the glider, to ensure the Seaglider is functional.

The self-test is done from within the Seaglider operating program, and tests all Seaglider systems, including the GPS and modem. The interactive self-test is the best way to ensure the Seaglider is working properly before proceeding.

4.2.6 Deck dives

These are simulated dives, so named because they have traditionally been done outside on a deck to give the antenna a clear view of the sky. Simulated pressure and pitch observations are generated to allow test dives (**$SIM_W** and **$SIM_PITCH**). This is a valuable way to test the end-to-end data path, since the basestation is involved and has to deal with "real" data files. The "Test Launch!" menu item (in the Launch menu) should be used for deck dives.

4.2.7 Transport

Before packing the Seaglider for transport, use the menu option *hw/misc/travel* to prepare the Seaglider for transport. For short trips, the Seaglider can be transported in its handling cradle. Longer trips, or commercial shipment, requires use of the plastic shipping case. The standard for SGX is a single long shipping case; two-part cases are also available, but require some disassembly of the glider (to the pupae).

4.3 Launch

The actual Seaglider launch procedure is initiated by an operator directly connected to the Seaglider via the serial communications cable. The operator selects the "Sea Launch!" menu option from the Launch menu. A final self-test should be executed prior to sea launch in order to verify that all hardware, electronics, and communications are functioning correctly.

The pilot's responsibilities are to review the results of the self-test file, and to ensure that the proper parameters and **cmdfile** directives are in place when the Seaglider is launched. The Seaglider will transfer the results of the self-test (**.cap** file) and a complete dump of all the parameters (**.prm** file) prior to launch. The pilot should carefully review these files prior to giving a positive answer when the field operator asks for clearance to launch.

Once the pilot has given clearance to launch, the Seaglider goes into a normal surface procedure: acquires GPS1 and initiates a communications session. At that time, the usual file uploads are done, including **cmdfile**, and **science**, **targets**, and **pdoscmds.bat** if they exist. The normal launch

procedure is to place a **$QUIT** directive in the **cmdfile**, which will hold the Seaglider on the surface while surface position and acoustic ranging performance are verified. If the physical launch goes well, and the field team reports satisfactory surface position and acoustic ranging, the pilot can ensure that the proper shallow (test) dive parameters are in place, then replace the **$QUIT** directive in the **cmdfile** with **$RESUME**. That will start the first dive. Once the **cmdfile** has been successfully transferred and the communications session is successfully completed, the **$QUIT** directive should be placed back in the **cmdfile** and the **$RESUME** deleted so the Seaglider will hold at the surface following its first dive, allowing the pilot to review the dive before proceeding with further dives.

4.4 Test and trim dives

4.4.1 Diveplot

Data from the initial test dive should be examined in detail prior to making any parameter changes or letting the Seaglider continue on its second dive. The pilot is responsible for ensuring that the Seaglider is operating properly in all respects, that trim issues (if any) are within the adjustment range of the Seaglider, and that the Seaglider is cleared to continue on its mission. Affirmative answers to these questions will allow the pilot to release the field team for other duties.

It is most efficient to plot the data from the initial dive and inspect the plots. The basestation3 software automatically generates a series of plots and suggested parameter values after each dive that can be used to guide the trim.

4.4.2 Initial revision of trim

Initial adjustments of trim are done in order pitch, VBD, and roll. These adjustments are made by changing the system centers (**$C_PITCH**, **$C_VBD**, **$C_ROLL_DIVE**, and **$C_ROLL_CLIMB**), based on regressions of controlled (or expected) quantities versus observed outcomes.

For pitch, a linear regression is made between observed vehicle pitch, as measured by the inclinometer, and pitch control, the position of the pitch mass in cm relative to **$C_PITCH**. This linear regression will give a pitch bias and a pitch gain. We choose to adjust the **$C_PITCH** so that the pitch bias is zero. We generally only adjust half-way to the calculated **$C_PITCH** after one dive, since the regressions are not always robust enough to be fully trusted after only one dive. It may take another short dive to get **$C_PITCH** reasonably close to neutral pitch.

Once pitch is close, deeper dives can be undertaken, which then will provide more steady-state flight observations. These can be used to adjust **$C_VBD**, if necessary. These adjustments are initially made by comparing the observed Seaglider vertical velocity, $w_{observed} = dp/dt$, with the Seaglider hydrodynamic model predictions of vertical velocity as a function of calculated buoyancy and observed pitch angle. The **$C_VBD** is adjusted by the pilot to make the predicted and

observed vertical velocities more closely agree. Once again, based on experience, adjustments based on regressions of small numbers of dives are done in steps of about half the predicted adjustment.

Roll trim is an ongoing process, and continues throughout the deployment. Turn rate is regressed against roll control; the idea is to trim the Seaglider to fly straight, not to trim it to be flat. Separate roll centers are used for the dive and the climb phases. But environmental conditions can cause the Seaglider to require a lot of turning to stay on course, so the correct value of the roll neutral points can be elusive. Roll can only be effectively trimmed if the compass is calibrated.

4.4.3 Compass calibration

Compass calibration is the process of removing hard and soft iron effects from the compass magnetometer data so that we can isolate just the earth field and calculate heading. Hard iron effects come from materials with their own local magnetic fields (like steel in the battery cell casings). They are fixed in the local coordinate system of the compass, i.e., they move and rotate with the glider in the same way that compass does. Hard iron manifests as an offset in the measured magnetic field. Soft iron comes from materials which produce a magnetic field in the presence of another magnetic field. Soft iron effects are not fixed in the local coordinate system. They distort the measured field through stretching and rotation. Hard iron is usually (but definitely not always) a larger source of error on the glider.

Some form of compass calibration should be applied for every mission.

The glider uses a 3D accelerometer and 3D magnetometer to compute heading. We use the accelerometer to calculate pitch and roll. The magnetometer measures 3 axes of magnetic field in a local coordinate system (x,y,z). We use the compass pitch and roll to rotate that local field into earth coordinates (X,Y,Z). The field we measure is a the sum of earth's field (which is what we want to measure to calculate heading) and hard and soft iron effects.

In the calibration we represent the hard iron as an offset vector (p,q,r) and the soft iron as a 3x3 transformation matrix (a,b,c,d,e,f,g,h,i):

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x-p \\ y-q \\ z-r \end{bmatrix}$$

There are multiple ways for a pilot to compute a compass calibration: the basestation3 visualization computes a per-dive calibration result and plot; the script Magcal.py can be run from the command line, or multi-dive calibration (same as Magcal.py) is available from the "Tools" menu on the basestation3 visualization. Transfer the result from any of these into a file named **tcm2mat.cal**, which has the format

hard0="p q r"

soft0="a b c d e f g h i"

and upload to glider by putting the file into the glider mission or home directory. The updated file will be transferred automatically during the next glider call and will be renamed on the basestation with the dive and dive and call number appended. The parameter **$COMPASS_USE** must be at least 4 for the glider to send back magnetometer data.

4.4.4 Progression of dives

Typical launches progress as follows: two dives to 45 m, then one or two dives to 200 - 250 m, then one or two dives to 500 m, then full depth dives. The exact sequence depends on Seaglider performance, water depth, and other operational considerations, but the idea is to get a progression of longer and longer dives, for better tuning of the Seaglider pitch, VBD, and roll.

Once the glider is diving regularly, the **$RESUME** directive in the **cmdfile** should be replaced with **$GO**. (Keeping the **$RESUME** directive in place means that if the glider goes into recovery mode it will immediately start diving again.)

4.5 Monitoring

Once the initial Seaglider trim is established on a deployment, the pilot mainly monitors the Seaglider performance. This includes how the Seaglider itself is working and how well the Seaglider is accomplishing its scientific mission. On routine deployments, checking the Seaglider status once per day may be sufficient. There may be times when more frequent checking is appropriate.

4.5.1 Seaglider performance

The basics are simple. Here are some of the things to monitor:

- Is the Seaglider communicating as expected? Check the Seaglider's **comm.log** file on the basestation for the latest communication. Do the throughput rates and communication session information seem normal ($>$200 Bps)? Are at least two-thirds of the communications sessions completed successfully with one phone call? Are there many missing or incomplete pieces of data or log files?
- Are the GPS positions current in time and reasonable in position? Are the positions acquired in reasonable amounts of time (~45 s for GPS1, ~15 s for GPS2)? Are the HDOP values generally less than 2.0?
- Are the sensors functional and providing believable profiles?

- Are the Seaglider control systems (pitch, roll and VBD) working normally? Check the motor currents and rates of movement of the device. Is there a trend or a sudden jump? Are there retries or errors?
- Have there been large jumps in internal pressure or humidity?
- Is the glider's volmax changing? Small rate hydraulic leaks manifest themselves as the Seaglider apparently getting heavier. In fact, the Seaglider is actually losing buoyancy for a given amount of oil in the internal reservoir – oil is going somewhere other than into the external bladder.
- Is the specified surface buoyancy sufficient to fully expose the antenna? Check **$_SM_DEPTHo** logfile file. Values of 0.7 m are about normal. Anything over 1.0 m should be reason to adjust **$SM_CC**.

4.5.2 Data completeness and quality

Be sure that all the data for each profile has been transferred successfully to the basestation. There are many ways to do this. One simple way is to compare the size of the processed **p*.dat** file to the file size specified as the first argument in the **$DATA_FILE_SIZE** line in the **p*.log** file. For gliders with scicon, data can be easily viewed in **.dat** or **.eng** files with the scicon directories (one directory per dive and one per climb). If there are missing pieces or timeouts, the **comm.log** file can be an easy way to determine which chunk of data was incompletely transferred. The **resend_dive** command in the epdos command set allows for individual chunks of data or log files to be resent at the next Seaglider surfacing.

4.5.3 Troubleshooting

The contents of the **p*.log** files, the capture file mechanism, and the extended PicoDOS command set (used via the **pdoscmds.bat** file) provide useful tools for troubleshooting.

4.5.4 Resources

Contact the IOP Seaglider group for further help and information (iopsg@uw.edu).

4.6 Recovery

A normal Seaglider recovery is initiated by directing the Seaglider to a specified recovery point. The Seaglider pilot often makes the Seaglider perform relatively shallow (short) dives at the recovery point while awaiting word on the arrival of the recovery team. That way, the Seaglider is relatively close to the surface at all times and there are regular opportunities to place the

Seaglider at the surface. The actual recovery is initiated by putting the **$QUIT** directive in the **cmdfile**. An adjustment to **$T_RSLEEP**, the time (min) to sleep between calls while in recovery, is often made at this time. The **.pagers** or **pagers.yml** file mechanism is generally used to transmit the most recent Seaglider surface position to the recovery team via SMS to an Iridium handset or cell phone.

It is important for the pilot to check the **$INTERNAL_PRESSURE** log file parameter prior to authorizing recovery, to check for possible battery outgassing.

Recoveries have also been accomplished with general locations from Iridium and specific location via acoustic ranging when GPS was not available.

Physical recovery depends on the vessel in use. From low-freeboard vessels, recovery can be directly into the cradle. From larger vessels, a lasso is generally placed around the vertical stabilizer and the Seaglider is lifted aboard with a crane or winch.

Once on deck, the Seaglider is powered off and rinsed with fresh water if available; the antenna, wings and rudder are removed; and covers are placed on the sensors. If a computer and communications cable are available, the Seaglider can be powered on and put in the 'travel' mode as described above.

Appendix A

Main electronics

The Seaglider Rev E motherboard uses an ARM microprocessor as the primary computational engine for glider functions. A second ARM processor is used for motor control functions. An 8-bit microprocessor is used for supervisor and analog input functionality. The supervisor is responsible for:

1. on/off control of the glider (reed switch or shorting plug detection)
2. software watchdog
3. continuous real-time fuel gauging
4. battery voltage monitoring
5. internal sensor monitoring (temperature, humidity, internal pressure)
6. external pressure sampling (when not using aux attitude)

The motor controller has access to an onboard accelerometer and can independently sample the external pressure sensor. These data are provided in motor move records, but are not yet used operationally. The motor controller uses a 16-bit AD converter for motor feedback potentiometers. These values are divided by 16 for compatibility/familiarity with Rev B 12-bit numbers. The only user visible effect of this is that some AD count values are now presented as floating point numbers.

Firmware for the main processor and motor controller are generally user upgradeable. Supervisor firmware is not user upgradeable. Motor controller and supervisor firmware generally changes much less frequently than the main firmware. All of the firmware modules have interdependencies and thus compatibility must be considered when making updates.

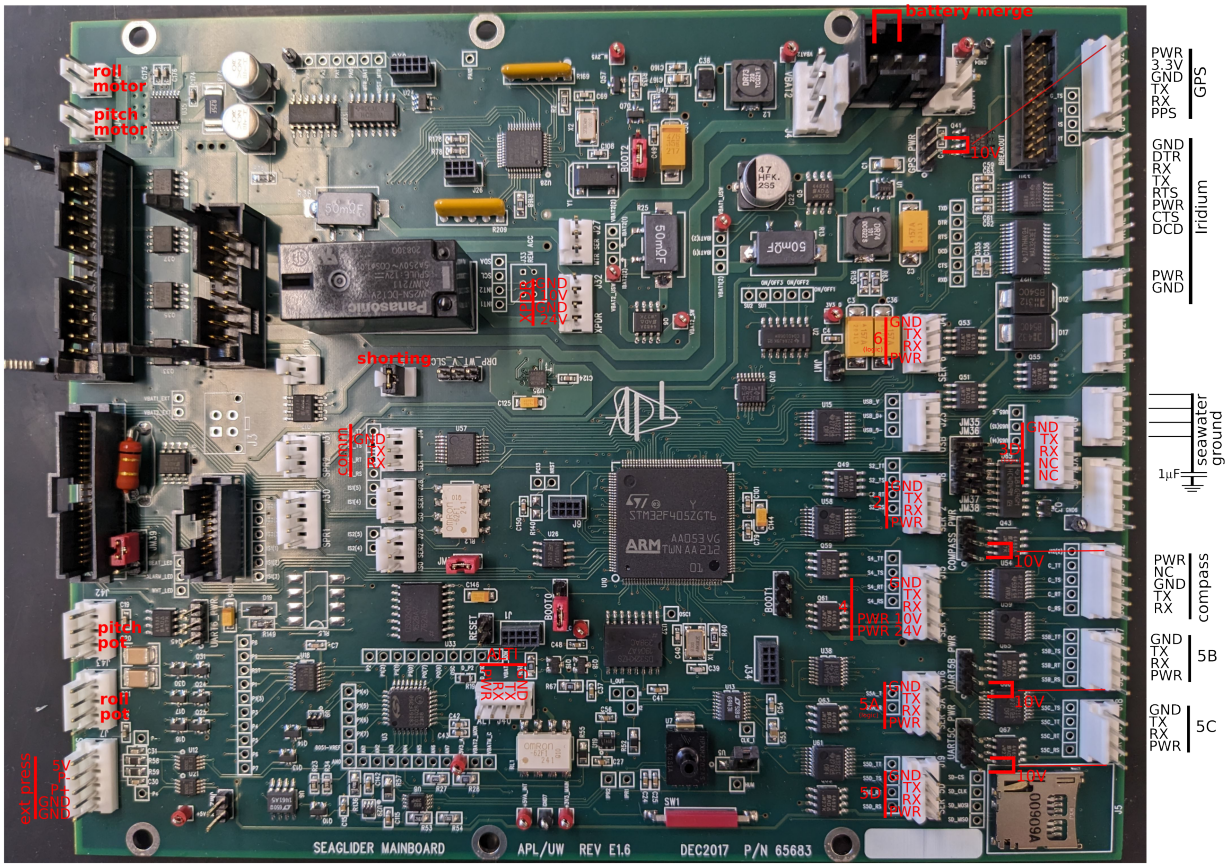

Figure A.1: Seaglider Rev E motherboard connectors.

A.1 Available sensor ports and configuration

Standard sensor port pin out is: 1-GND, 2-TX, 3-RX, 4-power (TX and RX sense is Seaglider TX to sensor and Seaglider RX from sensor). Some ports have jumper selectable voltage: 5B (10 or 24), 5C (10 or 24), and 6 (10/15 or 3.3). Port 4 supports software selection of voltage (10 or 24) on different pins. In a univolt 15V glider, the 10/24 distinction is meaningless, but the jumpers must still be installed. GPS and compass voltage are also jumper selectable (10/15 or 3.3). Always confirm the device voltage input before setting the jumpers and energizing the system. Sending 15V to a 3.3V device will almost certainly damage the device beyond repair.

Three ports (3D, 5A and 6) are always logic level serial (not RS232). Several ports can be made logic level via hardware modification to the motherboard. Many boards ship standard with logic level GPS, Iridium phone and compass ports for example. Check with your service provider if you are unsure about the configuration of your board. Key indicators are the presence or absence

of U27 (Iridium phone), U53 (GPS), and U54 (compass) on the board. If that part is present then the port is configured for RS232. If the corresponding part is absent, then that port is configured for logic level. Connecting a logic level device to an RS232 port will almost certainly damage the device. Likewise, connecting an RS232 device to a logic level port on the motherboard will very likely damage the motherboard.

A.2 Transponder connection

The recommended configuration is to connect the transponder serial to port 5A or 6. The connection must be logic level. The previous default was to connect to J40 (ALTI on silkscreen) which provides a pass through connection via the supervisor. This configuration still works (choose "null port" in the configurator), but will be deprecated in future hardware revisions. When connecting to 5A or 6, do not connect pin 4 (the transponder has always on power and does not use the switched port power). The Rev B pin out is 1-TX, 2-RX, 3-GND. The Rev E pin out (5A, 6, or ALTI) is 1-GND, 2-TX, 3-RX, 4-NC.

A.3 Ribbon cables

There are up to 5 ribbon cables running fore-aft in a Rev E equipped glider. The high density 40-pin cable and the standard density 26-pin (previously "rainbow") cable are always required. The high density 20-pin cable is required if port 5D or any of the spare connectors are used on the tailboard. The 16-pin standard density cable supports the aft mounted battery. A fifth cable supports the aft mounted Iridium-GPS board standard on SGX.

A.4 Aft endcap connections

Most external sensors are mounted in the tail section of SGX and connect to the electronics via a bulkhead connector on the aft endcap. Bulkhead internal wiring is generally either to scicon (it outfitted) or to the tailboard. Pinouts for sensor connections on the tailboard are the same as on the mainboard.

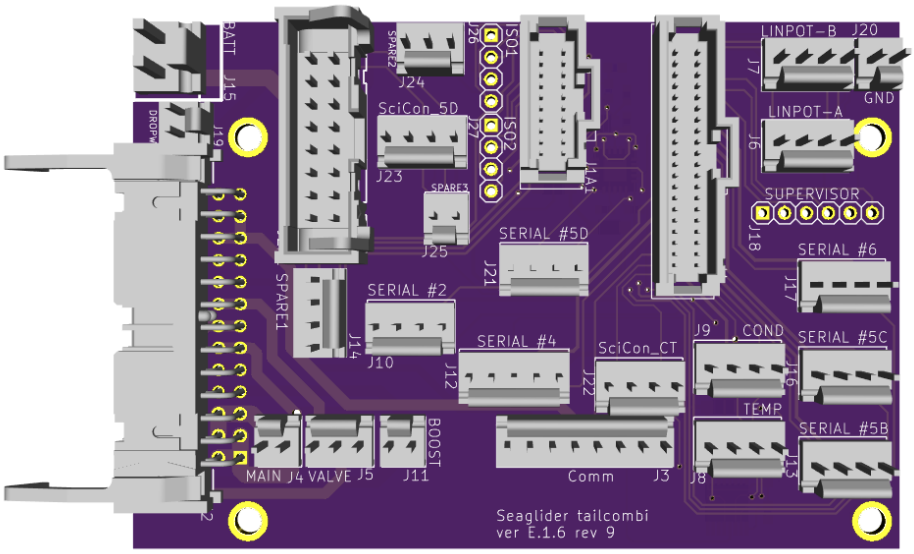

Figure A.2: Seaglider Rev E tailboard for aft mounted electrical connections.

Appendix B

Menu tree

.

```
 [param   ] Parameters and configuration
    **Flight control and mission definition
    [basic   ] Basic mission and glider parameters
    [dive    ] Dive parameters
    [flight  ] Flight parameters
    [surface ] Surface parameters
    [rafos   ] RAFOS parameters
    [password] Set glider login password
    [telnum  ] Set/show basestation phone number
    [altnum  ] Set/show basestation alternate phone number
    **Pitch, roll, VBD
    [pitch   ] Pitch parameters
    [roll    ] Roll parameters
    [vbd     ] VBD parameters
    **Sensors and peripherals
    [config  ] Hardware configuration parameters
        init=int
       [show    ] Show current hardware configuration
       [motherbo] Configure motherboard
           type=int
       [compass ] Configure compass
       [compass2] Configure spare compass
       [phone   ] Configure phone
           flow=int
       [gps     ] Configure GPS
```

```
   [rafos   ] Configure RAFOS
   [network ] Configure network
   [nav0    ] Configure nav[0]
   [nav1    ] Configure nav[1]
   [nav2    ] Configure nav[2]
   [nav3    ] Configure nav[3]
   [xpdr    ] Configure XPDR
   [sensor  ] Configure sensor
       slot=int
   [logger  ] Configure logger sensor
       slot=int
   [sencnf  ] Re-read sensor config
       slot=int
   [logcnf  ] Re-read logger config
       slot=int
   [seradd  ] Add device cnf to hardware library
   [serdel  ] Delete device cnf from hardware library
   [serlib  ] List device cnf files in hardware library
   [logadd  ] Add logger cnf to hardware library
   [logdel  ] Delete logger cnf from hardware library
   [loglib  ] List logger cnf files in hardware library
   [pressure] Configure external pressure sensor
       slot=int
   [param   ] Edit parameters directly
[pressure] Pressure (external) parameters
[intpress] Pressure (internal) parameters
[compass ] Compass paramaters
[altim   ] Altimetry parameters
[seabird ] Seabird CT calibration
[power   ] Power parameters
**Utility
[all     ] Edit all parameters
[validate] Validate parameters
[details ] Show parameter details
[save    ] Save parameters by name to file
[dump    ] Dump parameters to screen
[load    ] Load parameters from file
    file=string
[reset   ] Reset to defaults
    confirm=int
[rebase  ] Rebase parameter memory
```

```
       confirm=int
[hw      ] Hardware tests and monitoring
   **Motors and VBD
   [pitch   ] Pitch control
      [read    ] Current position
          cont=int
          num=int
          chan=int
      [ad      ] Move to position (AD counts)
          pos=float
      [eu      ] Move to position (cm, - for fwd)
          pos=float
      [edit    ] Edit pitch parameters
      [cycle   ] Run pitch duty cycles
      [cyclepr ] Run pitch & roll duty cycles (CTL-Q to quit)
      [selftest] Autonomous selftest
   [roll    ] Roll control
      [read    ] Current position
          cont=int
          num=int
          chan=int
      [ad      ] Move to position (AD counts)
          pos=float
      [eu      ] Move to position (deg, + for stbd)
          pos=float
      [edit    ] Edit roll parameters
      [cycle   ] Run roll duty cycles
      [cyclepr ] Run pitch & roll duty cycles (CTL-Q to quit)
      [selftest] Autonomous selftest
   [vbd     ] VBD control
      [read    ] Current position
          cont=int
          num=int
          chan=int
      [ad      ] Move to position (AD counts)
          pos=float
      [eu      ] Move to position (cc)
          pos=float
      [param   ] Edit vbd parameters
      [open    ] Open valve
      [close   ] Close valve
```

```
   [cycle   ] Cycle valve
   [pump    ] Pump & bleed cycles (pressure chamber or AD)
   [soak    ] Pump and hold at pressure (pressure chamber or AD)
   [selftest] Autonomous selftest
**Fixed Devices
[super   ] Supervisor
   [sample  ] Sample A-D channel
       channel=int
       samples=int
       delay=long
   [readad  ] Read A-D register
       channel=int
       samples=int
       delay=long
   [monitor ] Dump monitored A-D values
       count=int
       pause=long
   [command ] Send command
       command=int
   [show    ] Show gain, fuel, monitoring, heart and watchdog rates
   [setup   ] Setup gain, fuel, monitoring, heart and watchdog rates
       fuel=int
       aux=int
       heart=int
       dog=int
       gain=int
       reed=int
   [readcfg ] Read configuration registers
       register=int
       length=int
   [accum   ] Read accumulators
   [clear   ] Clear accumulators
       command=int
   [minmax  ] Clear min/max values
       command=int
   [fuel    ] Read, latch and clear FG
   [write   ] Write register value
       register=int
       length=int
       value=long
   [read    ] Read register value
```

```
        register=int
        length=int
    [setrtc  ] Set supervisor clock (from uC RTC clock)
        count=int
        compare=int
    [testrtc ] Test setting supervisor clock (from uC RTC clock)
        count=int
        compare=int
    [rtctott8] Set uC RTC from supervisor
    [readrtc ] Read supervisor clock
    [status  ] Read status bytes
    [zerochk ] Check current zero points
    [zeroset ] Set current zero points
        off=long
        low=long
        curr=float
        curr=float
[pressure] Pressure sensor
    [selftest] Basic self-test
        warmup=long
        warmup2=long
    [sealevel] Sealevel test
        sealevel=int
        show=int
        samples=int
    [param   ] Edit pressure parameters
[compass ] Compass
    [selftest] Basic self-test
        warmup=long
        warmup2=long
    [active  ] Toggle active compass
    [dispraw ] Display raw bearing, pitch & roll
        warmup=long
        hdt=int
    [dispcal ] Display calibrated bearing, pitch & roll
        warmup=long
        hdt=int
    [direct  ] Direct comms with unit
        warmup=long
    [capture ] Capture compass serial output
        warmup=long
```

```
   [command ] Send command to SP3003
       command=string
   [reset   ] Reset SP3003
       command=string
   [whirly  ] Whirly calibration - display raw sensor outputs
       warmup=long
       hdt=int
   [whirlraw] Whirly calibration - (SP3003 only) display transducer counts
       warmup=long
       hdt=int
   [cal     ] Calibrate eng file
       t=long
       lat=float
       lon=float
       this=int
       file=string
       n=int
   [prev    ] Calibrate prev dives
       t=long
       lat=float
       lon=float
       this=int
       file=string
       n=int
   [insitu  ] Autocalibrate compass in-situ mode
   [coeff   ] Read calibration coefficients
       show=int
   [edit    ] Edit compass parameters
   [lsm     ] Stream LSM303
       num=int
       cal=int
       delay=long
[gps     ] GPS
   [selftest] GPS self-test (acquire fix)
   [display ] Display lat/long & sat info
   [cycle   ] Cycle GPS
       cycles=int
       sleep=int
       phone=int
       sync=int
       relay=int
```

```
   [direct  ] Direct comm with unit
       reset=int
   [capture ] Capture raw output
       reset=int
   [reset   ] Reset to deployment mode
       flags=int
   [pps     ] Check for presence of PPS signal
       state=int
   [clock   ] Use GPS to set TT8 RTC (uses PPS if available)
       state=int
   [version ] Report GPS version
   [on      ] Power and RF relay on
   [off     ] Power and RF relay off
   [magvar  ] Report magnetic variation
       lat=float
       lon=float
       order=int
[modem   ] Modem
    which=int
    baud=long
    addlf=int
    stroke=int
   [selftest] Self-test (does not make phone call)
   [direct  ] Direct comms
       baud=long
       addlf=int
       stroke=int
   [signal  ] Monitor signal quality
       baud=long
   [sms     ] Send SMS
       baud=long
   [smstest ] SMS selftest
   [inbox   ] Check SMS inbox
   [locate  ] Geolocate phone
       baud=long
   [id      ] Query phone id info
       baud=long
   [remote  ] Remote login and PicoDOS commands (xr and xs enabled for transfer)
   [upload  ] Exercise uploadData
       number=int
       param=int
```

```
   [uploadst] Exercise uploadData self-test results only
       number=int
       param=int
[rafos   ] RAFOS
   [selftest] Selftest
   [setclock] Set RAFOS clock
   [fromtt8 ] Set RAFOS clock from TT8
   [clock   ] Read RAFOS clock
   [settt8  ] Set TT8 clock from RAFOS
   [modemclk] Set modem clock
       which=int
       set=int
       advance=int
       check=int
       pulse=int
   [talkclck] Talk direct to clock
   [gather  ] Gather RAFOS correlations
       which=int
       listen=int
       sleep=int
   [report  ] Report RAFOS correlations from last gather
       which=int
       listen=int
       sleep=int
   [listen  ] Gather, sleep, report RAFOS correlations
       which=int
       listen=int
       sleep=int
   [cycles  ] Run gather/report cycles
       cycles=int
       listen=int
       clock=int
       delay=int
   [control ] Read schedule, test control
       gps=long
   [talkrcvr] Talk direct to receiver
   [schedule] Read RAFOS schedule
       read=int
   [show    ] Show RAFOS schedule
       read=int
   [source  ] Enable/disable source
```

```
        src=int
        state=int
    [modem   ] Direct comms w/nav modem
        baud=long
        addlf=int
        stroke=int
    [modemcmd] Send command to modem
        file=string
        string=string
    [modemfw ] Send firmware to modem
        file=string
        slot=int
        reset=int
        baud=long
    [hits    ] Hits file
        file=string
    [decode  ] Decode micromodem packet
        string=string
[network ] network
    [selftest] selftest
    [direct  ] direct comms
        baud=long
        addlf=int
        stroke=int
    [on      ] on
        interval=long
        offset=long
        lockout-interval=long
        lockout-length=long
        dest=int
        recycle=int
        file=string
    [off     ] off
    [ping    ] ping
    [clock   ] clock
    [control ] test control
        interval=long
        offset=long
        lockout-interval=long
        lockout-length=long
        dest=int
```

```
       recycle=int
       file=string
   [command ] commands
       file=string
       string=string
   [firmware] firmware
       file=string
       slot=int
       reset=int
       baud=long
   [xpdron  ] xpdr mode on
   [xpdroff ] xpdr mode off
   [rebox   ] mid-dive outbox
   [outbox  ] start outbox
[intpress] Internal pressure
   [selftest] Basic self-test
       warmup=long
       warmup2=long
   [read    ] Read internal sensors
   [param   ] Edit internal pressure parameters
[altim   ] Altimeter
   [selftest] Selftest (metadata, no ping)
   [ping    ] Ping the altimeter
       timeout=long
   [config  ] Upload configuration to altimeter
       depth=float
   [count   ] Query the transponder ping count
   [xpdrcfg ] Update XPDR configuration from parameters
   [direct  ] Direct comms with altimeter
   [param   ] Edit altimetry parameters
[sensors ] Sensors
[loggers ] Loggers
**Other
[batt    ] Batteries and fuel gauges
   [read    ] View battery gauges
   [reset   ] Reset battery gauges
       state=string
       value=float
       which=int
       confirm=string
   [backup  ] Backup battery gauges
```

```
   [voltage ] Battery voltage
   [clear   ] Clear voltage minima
[lowlevel] Low-level hardware (IO,A-D,CF)
   **I/O lines
   [io      ] Set and read IO lines
       pin=string
   [input   ] Read IO input lines
       pin=string
   [watchdog] Hang and test the watchdog
   [stroke  ] Repetitively stroke the watchdog
       pause=long
       pulse=long
   **Power control
   [measure ] Measure device power
       state=string
       interval=long
       save=int
   [report  ] Report active devices
   [diag    ] Diagnose power
   [model   ] Manage model currents
       state=string
       nominal=float
       last=float
       saved=float
       save=int
   **uSD card and RAM
   [size    ] Report SD card size
   [free    ] report RAM stats
   **Lowlevel periph
   [terminal] Serial port terminal mode
       string=string
       timeout=long
   [tee     ] Verify tailboard tee
   [checkout] Board checkout
       pause=int
   [count   ] Count periods
[misc    ] Miscellaneous (travel, timeouts, date/time)
   [travel  ] Prepare for travel
   [iotime  ] Change user IO timeout
       timeout=int
   [date    ] Read/set time-of-day
```

```
          state=int
       [zerost  ] Reset self-test counter to zero
       [fault   ] Force a fault - WARNING, CAUSES RESET!!!!
       [dump    ] Dump captured registers
    [develop ] Developer tests
       [lpsleep ] Test low-power sleep
           seconds=long
           ticks=long
           wait=long
           cycles=long
       [file    ] Test file creation/manipulation
       [cap     ] Capture Test
 [modes   ] Test operation modes and files
    [bathy   ] Test bathymetry files
       [verify  ] Verify bathymetry files
       [selftest] Run bathymetry self-test
       [report  ] Report depth toward target
           lat=float
           lon=float
           targetlat=float
           targetlon=float
    [ice     ] Test icemap files
       [verify  ] Verify ice files
       [selftest] Run ice self-test
       [report  ] Lookup ice condition
           lat=float
           lon=float
           date=string
    [recovery] Test recovery
    [surface ] Test surface maneuver
    [sample  ] Test sampling and data file creation
        samples=int
        interval=float
        fuel=int
    [angle   ] Test surface measurements normal
    [upload  ] Exercise uploadData regular operations
        number=int
        param=int
    [uploadst] Exercise uploadData self-test results only
        number=int
        param=int
```

```
   [modes   ] Test active/passive modes
       state=int
       depth=float
       active=int
       num=int
       turns=int
       passive=int
       pitch=float
       buoy=float
[pdos    ] pdos commands
[launch  ] Pre-launch
   [selftest] Perform interactive self test
   [autotest] Perform autonomous self test
   [uploadst] Upload self-test results
       number=int
       param=int
   [quick   ] Quick test (nose-off)
   [nosed   ] Quick test (nose-on)
   [reset   ] Reset dive/run number
   [test    ] Test Launch!
   [sea     ] Sea Launch!
[shutdown] Shutdown
    confirm=int
```

Appendix C

Extended PicoDOS commands

C.1 System commands

menu *menuPath args ...*

> Execute a command from the menu system. menuPath is a / separated list of menu, sub-menu and menu options as described by the [name] strings displayed in the menu system. For example, *menu hw/vbd/ad pos=2000* will move the VBD to 2000 counts. This is equivalent to navigating to the hardware menu (hw), VBD sub-menu (vbd) and choosing the AD option to manually move the VBD control.

quit

> Exit the epdos system and return to the menu system.

pdos [*/f*]

> Exit the glider program. Reboots the glider firmware. The */f* flag forces the exit (skips the confirmation question.

reboot

> Sets **$RELAUNCH** to indicate an epdos commanded reboot and restarts the glider program.

clock [*mm/dd/yyyy HH:MM:SS*]

> Read or optionally set the main real-time clock. This does not set the supervisor RTC (use *hw/super/setrtc* from the menu) or any peripheral clocks.

flash backup *a-f*

Save currently flashed firmware to a file, **backup_x.bin** on microSD card, where *x* is one of *a-f*.

sysclk [*speed*] [*hse*]

Display system clock information. If speed is specified, set CPU operating clock to speed. Specify hse to use the high-speed external oscillator as the system clock (this is almost always what you want). Use with care. An incorrect speed specification could leave the glider in an inoperative state.

scuttle *confirmation*

Scuttle the glider. This pitches the glider full forward and opens the valve.

wipe *confirmation* Wipe the microSD card by filling the entire card with a random sequence of bytes.

C.2 Scripting commands

stroke

Stroke the watchdog.

time *command args* ...

Run *command* with *args*. Report the execution time.

repeat *count command args* ...

Run *command* with *args*, *count* times. *command* can be a batch file.

abort_if *0|1* (only useful inside a batch file) Stop batch file execution if the previous command returned False (0) or True (1). Not all commands return useful values.

delay *milliseconds*

Pause execution for *milliseconds*. Does not enter low power sleep.

sleep *seconds*

Enter low power sleep for *seconds*.

print *arg1* [*arg2*...] [> | >> *dest*]

Print (echo) strings to screen (and capture file) or file *dest*. Arguments are expanded using a similar syntax as logdev and serdev serial strings. See argument expansion below.

C.3 File transfer commands

yr

Receive files via YMODEM.

ys *file*

Send *file* via YMODEM. In online mode (during Iridium call, via **pdoscmds.bat** for example), *file* must be a single file. From the console to a computer connected via serial port, *file* can be glob (e.g., sg*dz.a) to send multiple files.

raw_r *file*

Receive *file* via raw protocol.

raw_s *file*

Send *file* via raw protocol.

C.4 File utility and shell-like commands

gzip *originalFile compressedFile*

Compress a file.

gunzip *compressedFile uncompressedFile*

Uncompress a file.

tar *x*|*t*[*v*][*z*] *tarFile*

Extract (*x*) or display contents (*t*) of *tarFile*.

tar *c*[*v*][*z*] *tarFile glob* [*pathglob*]

Create a tar file. Specify *z* to create a gzip compressed tar file. *v* to operate verbosely. *path* is optional. If specified it will be pre-prended to the the file pattern *glob*. *path* can itself be a *glob*. For example, “tar cvz eng.tgz sg0*dz.a dv00*”, will search all directories matching dv00* for files matching sg0*dz.a and tar them into a single file.

md5 *file*

Compute and display MD5 hash of file.

mkdir *directory*

Create *directory* on the microSD card.

ren *oldName newName*

Rename a file.

mv *glob1* [*glob2* . . .] *directory*

mv is a directory-aware version of rename, similar to the Unix mv command. Multiple glob patterns/filenames can be specified. The destination directory must exist. If there are just two arguments and the second argument is not an existing directory than mv behaves exactly like rename.

cp *file1 file2*

Copy a file.

del [*/v*] *glob1* [*glob2*. . .]

Delete files from microSD card.

rm *glob* [*path*]

Recursively delete files matching *glob*. *path* is optional. If specified, it is the starting point (highest level) in the directory tree for the search for matching files.

dir [*glob1*] [*glob2*...]

Show directory entries for files matching a list of glob patterns. If no patterns are given, shows the entire root directory.

ls [*glob*] [*path*]

Recursively show directory entries for files matching *glob*, with optional starting *path*. If no arguments are given, lists the entire directory tree.

dirfile *file* [*glob*] [*path*]

Same as ls but results are saved to file rather than output to screen.

usage [*glob1*] [*glob2*...]

Count files and sum sizes for files matching a list of *glob* patterns. If not patterns are given, shows the entire root directory.

cat *glob1* [*glob2* . . .] [> | >> *dest*]

Cat (output) files to console or optionally redirected to a file. Use > to overwrite *dest*. Use >> to append to *dest*.

trunc *file length*

Truncate *file* to *length* bytes.

grep */s string1* [*/s string2* . . .] [*/b linesBefore*] [*/a linesAfter*] [*/v*] [*/c*] [*/or*] *glob1* [*glob2*...]

Search files matching a list of glob patterns line-by-line for matching string patterns. Files can be gzipped or plain text. Use */b* and */a* to control how many lines of context are printed before and after matched lines. */c* specifies that only a count of matches will be returned. */v* inverts the search - only non-matching lines are reported (or counted). By default multiple search conditions are and-ed together (i.e., a line in a file matches only if all strings are found on that line). Use */or* to change this to logical or.

xargs *pathspec filespec* [*commands*] [*args*]

Similar to a unix pipeline consisting of find piped to xargs, builds a list of files by separately globbing paths and filenames. Searches for directories matching pathspec, and then within each matched directory searches for files matching filespec. With no additional arguments a list of matched files is printed. If command is specified (possibly with its own arguments) the list of files is appended as additional arguments. Valid commands are any that take a list of files or glob patterns as the final arguments (dir, usage, del, grep, cat (without redirection)). "xargs dv001? sg*kz.a grep /s ,C," for example searches dives 10-19 capture files for the string ,C,.

C.5 Capture system commands

capvec [*subsystem level dest*]

With no arguments, prints the current capture vector information. With arguments, sets debugging verbosity/level (DEBUG, NORMAL, CRITICAL) and destination (BOTH, FILE, SCREEN, NONE) for *subsystem*.

capflush

Flush memory buffered capture contents to disk.

capclose *file* [*capFile*]

Close the current working capture contents (or capFile if specified) to *file* and start a new empty capture file. In online mode (during an Iridium call via **pdoscmds.bat**), the capture file has already been moved to a temporary name to store the output of the commands in **pdoscmds.bat**. To close and reset the "regular" capture file (in which all of the offline results have been stored), specify thisdive.kap as *capFile*.

C.6 Glider control and data file commands

target [*targetName*]

Re-read targets file. With no arguments, displays current targets. With an argument, sets *targetName* as the active target.

science

Re-read and report **science** file contents.

resend_dive [*/l* | */d* | */c*] *diveNumber* [*fragment1*] [*fragment2*...]

Queue log (*/l*), eng (*/d*), capture (*/c*), or all (no specifier) files from *diveNumber* for re-transmission. A list of specific fragments to resend is optional. If not provided, all fragments will be sent.

split *file*

Split *file* into fragments according to **$N_FILEKB**.

resend_dive */f file* [*fragment1*] [*fragment2*. . .]

Queue logger file file for re-transmission. A list of specific fragments to resend is optional. If not provided, all fragments will be sent.

C.7 Low-level filesystem and microSD commands

sdinfo

Report microSD card information (serial number, manufacturer info, capacity, etc.)

sdfree [*/v*]

Report microSD card free space. With the */v* option, report verbose information about the filesystem (open files, etc.)

mem

Display memory (RAM) status.

format *confirmation*

Format the microSD card filesystem.

walkdsk [*/v*]

Recursively walk the entire filesystem to look for potential errors. */v* turns on verbose reporting.

checkdsk [*/v*]

Recursively walk the entire filesystem and attempt to open and close every file.

fsck [*/v*]

Recursively walk the entire filesystem, opening and close every file. Tries to delete any file that cannot be opened.

C.8 Low-level nonvolatile memory commands

initnv *page*

> Query the status of of nonvolatile storage page *page*. Page 0 is parameter memory. Page 1 is utility memory.

dumpnv *page*

> Display low-level information about storage page *page*.

formatnv *page confirmation*

> Format (erases all existing data) *page*. *confirmation* must be *YES*.

readnv *var*

> Read value of utility variable *var*.

writenv *var value*

> Write value to utility variable *var*.

C.9 Exec system commands

eval [*conditionName*]

> If no *conditionName* is specified, evaluate all currently set conditions. If the condition evaluates true, execute the corresponding set number or pdos command.

clear *conditionName*

> Delete *conditionName* from the list of active conditions.

condition *name=conditionName* [*set=setNumber* | *pdos=command*] *expresssion= "conditionalExpression" init="expression"* [*init="expression"* ...]

> Define a new condition and add it to the list of active conditions.

exec [*setNumber*]

> If no *setNumber* is given, show all active conditions. If *setNumber* is given, execute it.

Appendix D

Glider files

There are several categories of files on the basestation:

1. **Control files**: the pilot can make edits/changes to the glider's operational parameters using the control files, which are located on the basestation. Every time the glider calls in, the basestation uploads any control files onto the glider and renames the file on the basestation **<name>.ddd.nnn**, where **ddd** is the dive number and **nnn** is the call cycle. This allows the pilot to see a complete history of the control files that have been sent to the glider. Control files include **cmdfile**, **science**, **targets**, **pdoscmds.bat**, **tcm2mat.cal**, and scicon files **scicon.sch**,**scicon.ins**, **scicon.att** (gliders with scicon only).

2. **Log and Data files**: The glider uploads log and data files it generated on the previous dive(s) to the basestation. These include files with the following extensions: **.cap**, **.log**, **.dat**.

3. **Basestation control files**: Files located on the basestation that control calibration coefficients for converting sensor output into science data, generation of messages and emails containing glider health and location information, instructions for generating plots, instructions on file types to be produced by basestation software, and instructions on where data should be sent for archive or further processing. These include **sg_calib_constants.m**, **sg_plot_constants.m**, **sections.yml**, **.pagers** or **pagers.yml**, **sgGGG.cnf**.

4. **Basestation output files**: Processed science and engineering data files, intermediate products and plots generated on the basestation from the glider data files. These include files with the following extensions: **.asc**, **.eng**, and **.nc**.

5. **Basestation log files**: Files generated on the basestation that record all interactions between the glider and the basestation, the interaction between the basestation and the RUDICS connection, and the status of glider data file processing on the basestation. These

include **comm.log**, **baselog_login_YYMMDDHHMMSS**, **glider_early_gps_YYMMDDHHMMSS**, **baselog_YYMMDDHHMMSS**, **baselog.log**, **processed_dives.cache**, and **alert_message.html.ddd.ccc**.

D.1 cmdfile file

The **cmdfile** file provides the optional input to change values of the Seaglider parameters, and the mandatory directive to maintain or change the Seaglider's state. A complete description of the parameters and command file directives is in the Parameter Reference Manual.

The **cmdfile** file is a line-oriented format. A line in the **cmdfile** has one of two possible formats:

$tag, value

where **$tag** is a parameter and **value** is the new value of that parameter,

or

$directive

where **$directive** is one of the three valid cmdfile directives (**$GO**, **$RESUME**, or **$QUIT)**, as described in the Parameter Reference Manual.

D.2 targets file

The **targets** file provides the input to the Seaglider navigation logic to determine what surface position(s), or target(s), the glider should fly to.

The **targets** file is a line-oriented format. Lines may be comment lines, in which case the character in column 0 of that line must be a **/** and the balance of the line is ignored. All other lines are treated as targets lines. A target line has the following format:

target_name lat=**latitude** lon=**longitude** radius=**radius** goto=**next_target** [escape=**escape_target** depth=**bathymetric_target** finish=**goal_direction**]

target_name is the name of the target. It may be any string that does not contain embedded whitespace. **latitude** and **longitude** are the targets latitude and longitude in DDMM format (negative values being southern or western hemisphere). **radius** is the target radius, expressed in meters. **next_target** is a target identifier indicating the next target the glider should proceed to, once this target has been achieved. The **next_target** must exist as a **target_name** within the same file. A target can refer back to itself.

The **escape_target** is an optional entry that specifies what target to move to in the event of recovery scenarios in which normal park on the surface recovery is not desirable. The **escape_target**

must be a valid named target in the file and can vary for each named target. One possible use is to have the standard targets along a cyclical survey route all point to a single escape target that then points (through **next_target**) to a series of targets which define an entire route to a convenient recovery location.

Targets can also be achieved by crossing a line defined by the **goal_direction**. This finish line concept works by specifying the heading that points in the direction of finish line completion. The finish line is an imaginary line drawn through the target, perpendicular to the specified finish direction. The target is considered achieved when the difference between the bearing to the target and the finish direction is greater than 90 (or less than -90) degrees. For example, a **goal_direction** of 90 specifies a north-south finish line drawn through the target; the target is achieved when the glider is east of the line. A value of 180 specifies an east-west finish line; target is achieved when glider is south of the line. A value of -1 or no specification of finish= means that no finish line will be tested.

Specifying an optional **bathymetric_target** means that the target can be achieved by crossing a bathymetric contour. If the value is positive the target is achieved when crossing that contour from deep to shallow. When negative, target achievement is defined by moving across that contour from shallow to deep. The glider measures its depth for comparison against the target depth either by altimetry or via $T_NO_W.

By default, the glider will search its nonvolatile storage for a recent target name. If that name does not exist, it will select the first listed target as its current target, fly toward that target until it gets within the specified radius, then it will make the next target the current target and repeat the process. The **$HEADING** parameter can override this algorithm.

Example of a simple **targets** file:

```
Shilshole targets (small rectangle)
SE  lat=4743.0 lon=-12224.0 radius=100 goto=NE
NE  lat=4743.5 lon=-12224.0 radius=100 goto=NW
NW  lat=4743.5 lon=-12225.0 radius=100 goto=SW
SW  lat=4743.0 lon=-12225.0 radius=100 goto=SE
```

Edit the **targets** file on the basestation with *targedit*, which provides some bounds checking and verification of changes.

D.3 science file

The **science** file sets the **G&C** intervals (all gliders) as well as the schedule that the installed sensors will be sampled for data (for gliders without scicon only). The schedule and **G&C** intervals are bound to specific ranges of the depth called **depth bins**.

The **science** file is a line oriented format. Lines may comment lines, in which case the character in column 0 of that line must be a **/** and the balance of the line is ignored. All other lines are treated a depth bin lines. A depth bin line has the following format:

> **bottom_depth** *tab* gc=**gc_int** *tab* seconds=**sensor_sampling_interval** *tab* sensors=**sensor_mask** *tab* compass=**compass_sampling_interval** *tab* pressure=**pressure_sampling_interval**

Where *tab* is a single tab character (not spaces). **bottom_depth** is the lowest water depth in this bin. The upper limit of the depth bin is either the previous depth bins **bottom_depth** or the surface. **gc_int** is the **G&C** interval, in seconds, for that depth bin. **sensor_sampling_interval** is the most frequent sensor sampling interval in that depth bin, in seconds. **sensor_mask** is a string of up to six digits, with each digit having a possible value of 0 to 8. The digits position corresponds to the sensor that is installed in that position (per **$SENSORS**). Thus, the first digit represents the first sensor (almost always the CT sensor). Each digit indicates the sampling interval of that sensor as an integer multiplier of **sensor_sampling_interval**. A 0 indicates do not sample at all. A 1 indicates sample every **sensor_sampling_interval**. A 2 indicates sample every other **sensor_sampling_interval**, etc. **compass_sampling_interval** is an integer multiplier of **sensor_sampling_interval** specifying the sample interval of the compass. **pressure_sampling_interval** is an integer multiplier of **sensor_sampling_interval** specifying the sample interval of the pressure sensor.

For gliders running scicon, the **sensor_mask** will be ignored.

D.4 bathymap file

bathymap files provide the glider with geographic (and sometimes temporal) environmental information. A bathymetry map provides the glider with bathymetry data about a given region of the ocean. The glider may carry up to 999 bathymetry maps, but in practice far fewer are on board. These maps are not required for gliders to fly. Files are named **bathymap.xxx** where *xxx* is a number from 001 to 999.

In addition to bathymetry maps, the glider can carry ice maps that indicate a spatially and temporally varying climatology of ice cover. The glider can use this information to make decisions about surfacing.

The **bathymap** files contain a fixed-size header followed by a variable-length data section. The header is defined as follows:

> **row_dim** *ws* **col_dim** *ws* **ll_lon** *ws* **ll_lat** *ws* **delta** *ws* [**start_date** *ws* **end_date** *ws*] *newline*

Where *ws* is white space. **row_dim** is the integer number of rows of data are in the data section and **col_dim** is the integer number of columns. **ll_lon** and **ll_lat** specify the lower left corner of the

map in decimal degrees (DD) format (negative indicates western or southern hemisphere). **delta** is integer distance between grid points (in both dimensions) in meters. **start_date** and **end_date** are optional. They specify the valid dates (as decimal yeardays) for the map. When left unspecified (as is typical with bathymetry files) or when set to 0.0 no date checking is performed.

For a bathymetry map, the data section contains the depth of the bottom at each grid point, expressed in integer meters. The data is stored in column major order.

For an ice map the data section contains ice condition values for the time period between the start and end dates at each grid point. Ice condition values are stored as 2-bit integers packed sequentially together into sixteen equal length periods spanning the dates between start date and end date. Valid condition codes are: 0 = always surface, 1 = possibly ice, 2 = probably ice, 3 = always ice. As an example, for a start date = 0.0 and end date = 365.0, the lowest two bits of the value at any grid point encode the ice condition for the first 23 days of the year. Bits 2 and 3 cover the condition for the next 23 days, etc.

D.5 tcm2mat.cal file

The **tcm2mat.cal** file is used by the glider to calibrate heading values returned from the compass. See Section 4.4.3 for details on how to perform a compass calibration.

The format of this file is:

```
hard0="p q r"
soft0="a b c d e f g h i"
```

If the soft0 line is omitted, only the hard-iron calibration is applied.

Coefficients a–g are components of the 3x3 Poisson matrix used for soft-iron correction. p, q, r are the hard iron correction. For a measured magnetic field vector x, y, z, the field strength values are corrected for hard- and soft-iron and then rotated into earth coordinates to calculate the final heading value. The hard iron correction is

$$\begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} . \tag{D.1}$$

Field values in local coordinates corrected for soft-iron are calculated as

$$\begin{bmatrix} x_{hs} \\ y_{hs} \\ z_{hs} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} . \tag{D.2}$$

Rotation into earth coordinates is done using Euler angles

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos p_c & -\sin p_c \sin r_c & -\sin p_c \cos r_c \\ 0 & \cos r_c & -\sin r_c \end{bmatrix} \begin{bmatrix} x_{hs} \\ y_{hs} \\ z_{hs} \end{bmatrix}. \tag{D.3}$$

The heading value, H, is calculated as

$$H = \tan^{-1}\left(\frac{Y}{X}\right). \tag{D.4}$$

D.6 Data files

D.6.1 Overview

The data file is an ASCII text file that is generated by the Seaglider and transmitted to the basestation for further processing. For gliders not equipped with a science controller (scicon), it serves as the primary conduit for the science data collected by the Seaglider. For gliders with a scicon, the data file only contains data from sensors not managed by scicon; data files generated by scicon are described in the scicon manual (`https://iop.apl.washington.edu/iopsg/`).

Each data file covers one dive of information. The format is designed to minimize transmission size and, while clear text, is not intended for consumption by users.

While on the glider, the file is named **THISDIVE.DAT** during the dive. At the end of the dive, the file is renamed to be **p**<glider_id><dive_number>**.dat**.

The file is divided into a *header* section and a *data* section. The header consists of a series of lines of the format: **tag**:**value**, where **tag** is one of the tag values defined below and **value** is the tag's value.. There may be leading whitespace between the : separator and the actual value.

The header section terminates with the **data** tag. The balance of the file is the data recorded during the dive.

D.6.2 Header tags

version: Contains the version of the software glider that generated this data file. The format for the version is **MAJOR**.**MINOR**, where MAJOR and MINOR are two digit integers.

glider: Contains the glider's individual id value. The value is between 1 and 999, with no leading zeros.

mission: Contains the mission id this glider was on when this file was generated. The value maybe any positive integer, with no leading zeros.

dive: Contains the specific dive the glider was on when it collected the data. The value may be any positive integer, with no leading zeros.

start tag Contains the starting time stamp for the dive specified above, expressed in UTC. The format of the value is (with no leading zeros in any number): **month day year hour min sec** . Note that **year** is a two or three digit number representing years since 1900; **hour**, **min**, and **sec** are two digit numbers representing time in UTC.

columns: The value field for this tag contains a comma-delimited list of strings that indicate by their position what values are recorded in each of the columns in the data section of the file. The first 10 columns are always in a fixed order for any glider (Table D.1). The remaining columns (Table D.2) are determined by what sensors are installed in what sensor ports on the glider.

data: The data tag has no value associated - it simply exists as a delimiter between the header section and data.

Header Tag	Description
rec	Record number of the individual sample
elaps_tms	Time since the start of the dive [ms]
depth	Depth at the start of the sample [cm]
heading	Vehicle heading at the start of the sample [degrees magnetic x 10]
pitch	Vehicle pitch angle at the start of the sample [degress x 10, positive upward]
roll	Vehicle roll at the start of the sample [degrees x 10, positive starboard wing down]
GC_phase	GC phase, encoded as follows: 1 Pitch change 2 VBD change 3 Roll 4 Turning (passive) 5 Roll back (to center) 6 Passive mode (waiting)
mag.x	Magnetic heading in east direction [degrees x 10]
mag.y	Magnetic heading in north direction [degrees x 10]
mag.z	Magnetic heading in upward direction [degrees x 10]

Table D.1: Fixed-order columns that are found in the “columns” tag

Header Tag	Description
TempFreq	Temperature from SBE CT sail (cycle counts of 4MHz in 255 cycles of signal frequency)
CondFreq	Conductivity from SBE CT sail [cycle counts of 4MHz in 255 cycles of signal frequency]
aa4831.O2	Aanderaa optode oxygen concentration [μm x 1000]
aa4831.AirSat	Aanderaa optode oxygen air saturation [percent times 1000]
aa4831.Temp	Aanderaa optode temperature [°C x 1000]
aa4831.CalPhase	Aanderaa optode calibrated phase [degrees x 1000]
aa4831.CalPhase	Aanderaa optode temperature compensated phase [degrees x 1000]
wlbb2fl.BB1ref	Backscatter 1 reference [A/D counts]
wlbb2fl.BB1sig	Backscatter 1 data [A/D counts]
wlbb2fl.BB2ref	Backscatter 2 reference [A/D counts]
wlbb2fl.BB2sig	Backscatter 2 data [A/D counts]
wlbb2fl.FL1ref	Fluorescence 1 reference [A/D counts]
wlbb2fl.FL1sig	Fluorescence 1 data [A/D counts]
wlbb2fl.temp	WETLabs temperature [A/D counts]
qsp.PARuV	QSP2150 photosynthetically active radiation [μV]
O2Freq	Oxygen concentration from SBE43 oxygen sensor (cycle counts of 4Mhz in 255 cycles of signal frequency)

Table D.2: Sensor columns that can be found in the “columns” tag

D.6.3 Data

The data section contains rows of data collected during the dive. Each row contains data from a single sample interval, with contents of each column determined by the **columns** tag. The data itself is encoded in order to create a small data payload: the first row of data contains un-encoded data; all following lines are differences between the actual data and the value of the previous line.

If no value was collected for a specific column of the sample, then that column contains the letter N. The subsequent row’s entry for that column is an un-encoded data value.

D.7 Log file

One log file is generated by the glider per dive. The data contained in the log file are primarily related to the glider’s physical operation and status. The header section is the same as described

above in the the **Data file** description. The end of the header is denoted by the **data:** tag. The first portion of the data is a list of the Seaglider's parameters and their values for that dive; see the Parameter Reference Manual for more information. The other parameters are "pseudo-parameters" - that is, they are reported in the log file with the same format as parameters for processing convenience, but they are not values that the pilot can change directly via the cmdfile.

The second section, beginning with the entry $GPS1, contains information concerning the pre-dive period at the surface. The $GC-labeled lines describe motor actions (pitch, roll, or VBD), one line per motor move. The information listed after the $GC lines are data collected at the end of the dive (surface maneuver data, final temperature reading, etc). Some of this data is from the previous surfacing (before the start of the current dive). Not all gliders will report all of the lines that appear in the tables below because the devices installed vary among Seagliders.

D.7.1 Pre-Dive

The following tables outline the contents of the pre-dive section of the log file. The order of parameters in the pre-dive section of the file is the order they are found here.

(Pseudo) Parameter	Description
$ID	See Parameter Reference
$MISSION	See Parameter Reference
$DIVE	See Parameter Reference
$N_DIVES	See Parameter Reference
$STOP_T	See Parameter Reference
$D_SURF	See Parameter Reference
$D_FLARE	See Parameter Reference
$D_TGT	See Parameter Reference
$D_ABORT	See Parameter Reference
$D_NO_BLEED	See Parameter Reference
$D_BOOST	See Parameter Reference
$T_BOOST	See Parameter Reference
$D_FINISH	See Parameter Reference
$D_PITCH	See Parameter Reference
$D_SAFE	See Parameter Reference
$D_CALL	See Parameter Reference
$SURFACE_URGENCY	See Parameter Reference
$SURFACE_URGENCY_TRY	See Parameter Reference
$SURFACE_URGENCY_FORCE	See Parameter Reference
$T_DIVE	See Parameter Reference
$T_MISSION	See Parameter Reference
$T_ABORT	See Parameter Reference

$T_TURN	See Parameter Reference
$T_TURN_SAMPINT	See Parameter Reference
$T_NO_W	See Parameter Reference
$T_LOITER	See Parameter Reference
$T_EPIRB	See Parameter Reference
$USE_BATHY	See Parameter Reference
$USE_ICE	See Parameter Reference
$ICE_FREEZE_MARGIN	See Parameter Reference
$D_OFFGRID	See Parameter Reference
$CAPTURING	See Parameter Reference
$RELAUNCH	See Parameter Reference
$APOGEE_PITCH	See Parameter Reference
$MAX_BUOY	See Parameter Reference
$GLIDE_SLOPE	See Parameter Reference
$SPEED_FACTOR	See Parameter Reference
$RHO	See Parameter Reference
$MASS	See Parameter Reference
$NAV_MODE	See Parameter Reference
$FERRY_MAX	See Parameter Reference
$KALMAN_USE	See Parameter Reference
$HD_A	See Parameter Reference
$HD_B	See Parameter Reference
$HD_C	See Parameter Reference
$HEADING	See Parameter Reference
$ESCAPE_HEADING	See Parameter Reference
$ESCAPE_HEADING_DELTA	See Parameter Reference
$FIX_MISSING_TIMEOUT	See Parameter Reference
$TGT_DEFAULT_LAT	See Parameter Reference
$TGT_DEFAULT_LON	See Parameter Reference
$TGT_AUTO_DEFAULT	See Parameter Reference
$SM_CC	See Parameter Reference
$N_FILEKB	See Parameter Reference
$FILEMGR	See Parameter Reference
$CALL_NDIVES	See Parameter Reference
$COMM_SEQ	See Parameter Reference
$PROTOCOL	See Parameter Reference
$N_NOCOMM	See Parameter Reference
$N_NOSURFACE	See Parameter Reference
$UPLOAD_DIVES_MAX	See Parameter Reference

$NETBOX	See Parameter Reference
$CALL_TRIES	See Parameter Reference
$CALL_WAIT	See Parameter Reference
$CAPUPLOAD	See Parameter Reference
$CAPMAXSIZE	See Parameter Reference
$T_GPS	See Parameter Reference
$N_GPS	See Parameter Reference
$T_RSLEEP	See Parameter Reference
$STROBE	See Parameter Reference
$RAFOS_PEAK_OFFSET	See Parameter Reference
$RAFOS_CORR_THRESH	See Parameter Reference
$RAFOS_HIT_WINDOW	See Parameter Reference
$RAFOS_MMODEM	See Parameter Reference
$PITCH_MIN	See Parameter Reference
$PITCH_MAX	See Parameter Reference
$C_PITCH	See Parameter Reference
$PITCH_DBAND	See Parameter Reference
$PITCH_CNV	See Parameter Reference
$PITCH_GAIN	See Parameter Reference
$PITCH_TIMEOUT	See Parameter Reference
$PITCH_MAXERRORS	See Parameter Reference
$PITCH_ADJ_GAIN	See Parameter Reference
$PITCH_ADJ_DBAND	See Parameter Reference
$C_PITCH_AUTO_DELTA	See Parameter Reference
$C_PITCH_AUTO_MAX	See Parameter Reference
$PITCH_GAIN_AUTO_DELTA	See Parameter Reference
$PITCH_GAIN_AUTO_MAX	See Parameter Reference
$PITCH_W_GAIN	See Parameter Reference
$PITCH_W_DBAND	See Parameter Reference
$ROLL_MIN	See Parameter Reference
$ROLL_MAX	See Parameter Reference
$ROLL_DEG	See Parameter Reference
$C_ROLL_DIVE	See Parameter Reference
$C_ROLL_CLIMB	See Parameter Reference
$HEAD_ERRBAND	See Parameter Reference
$ROLL_CNV	See Parameter Reference
$ROLL_TIMEOUT	See Parameter Reference
$ROLL_MAXERRORS	See Parameter Reference
$ROLL_ADJ_GAIN	See Parameter Reference

$ROLL_ADJ_DBAND	See Parameter Reference
$VBD_MIN	See Parameter Reference
$VBD_MAX	See Parameter Reference
$C_VBD	See Parameter Reference
$VBD_DBAND	See Parameter Reference
$VBD_CNV	See Parameter Reference
$VBD_LP_IGNORE	See Parameter Reference
$VBD_TIMEOUT	See Parameter Reference
$PITCH_VBD_SHIFT	See Parameter Reference
$UNCOM_BLEED	See Parameter Reference
$VBD_MAXERRORS	See Parameter Reference
$C_VBD_AUTO_DELTA	See Parameter Reference
$C_VBD_AUTO_MAX	See Parameter Reference
$W_ADJ_DBAND	See Parameter Reference
$DBDW	See Parameter Reference
$LOITER_W_DBAND	See Parameter Reference
$LOITER_DBDW	See Parameter Reference
$LOITER_D_TOP	See Parameter Reference
$LOITER_D_BOTTOM	See Parameter Reference
$LOITER_N_DIVE	See Parameter Reference
$CF8_MAXERRORS	See Parameter Reference
$AH0_24V	See Parameter Reference
$AH0_10V	See Parameter Reference
$MINV_24V	See Parameter Reference
$MINV_10V	See Parameter Reference
$MAXI_24V	See Parameter Reference
$MAXI_10V	See Parameter Reference
$FG_AHR_10V	See Parameter Reference
$FG_AHR_24V	See Parameter Reference
$PHONE_SUPPLY	See Parameter Reference
$PRESSURE_YINT	See Parameter Reference
$PRESSURE_SLOPE	See Parameter Reference
$COMPASS_USE	See Parameter Reference
$ALTIM_PING_FIT	See Parameter Reference
$ALTIM_TOP_PING_RANGE	See Parameter Reference
$ALTIM_BOTTOM_TURN_MARGIN	See Parameter Reference
$ALTIM_TOP_TURN_MARGIN	See Parameter Reference
$ALTIM_TOP_MIN_OBSTACLE	See Parameter Reference
$ALTIM_PING_DEPTH	See Parameter Reference

$ALTIM_PING_DELTA	See Parameter Reference
$ALTIM_FREQUENCY	See Parameter Reference
$ALTIM_PULSE	See Parameter Reference
$ALTIM_SENSITIVITY	See Parameter Reference
$XPDR_VALID	See Parameter Reference
$XPDR_INHIBIT	See Parameter Reference
$XPDR_INT	See Parameter Reference
$XPDR_REP	See Parameter Reference
$INT_PRESSURE_SLOPE	See Parameter Reference
$INT_PRESSURE_YINT	See Parameter Reference
$DEEPGLIDER	See Parameter Reference
$DEEPGLIDERMB	See Parameter Reference
$MOTHERBOARD	See Parameter Reference
$DEVICE1	See Parameter Reference
$DEVICE2	See Parameter Reference
$DEVICE3	See Parameter Reference
$DEVICE4	See Parameter Reference
$DEVICE5	See Parameter Reference
$DEVICE6	See Parameter Reference
$LOGGERS	See Parameter Reference
$LOGGERDEVICE1	See Parameter Reference
$LOGGERDEVICE2	See Parameter Reference
$LOGGERDEVICE3	See Parameter Reference
$LOGGERDEVICE4	See Parameter Reference
$COMPASS_DEVICE	See Parameter Reference
$COMPASS2_DEVICE	See Parameter Reference
$PHONE_DEVICE	See Parameter Reference
$GPS_DEVICE	See Parameter Reference
$RAFOS_DEVICE	See Parameter Reference
$NAV_DEVICE	See Parameter Reference
$NAV2_DEVICE	See Parameter Reference
$NETWORK_DEVICE	See Parameter Reference
$PRESSURE_DEVICE	See Parameter Reference
$XPDR_DEVICE	See Parameter Reference
$SIM_W	See Parameter Reference
$SEABIRD_T_G	See Parameter Reference
$SEABIRD_T_H	See Parameter Reference
$SEABIRD_T_I	See Parameter Reference
$SEABIRD_T_J	See Parameter Reference

$SEABIRD_C_G	See Parameter Reference
$SEABIRD_C_H	See Parameter Reference
$SEABIRD_C_I	See Parameter Reference
$SEABIRD_C_J	See Parameter Reference
$OPTIONS	See Parameter Reference
$SC_RECORDABOVE	See Parameter Reference
$SC_PROFILE	See Parameter Reference
$SC_XMITPROFILE	See Parameter Reference
$SC_NDIVE	See Parameter Reference

Table D.3: Logfile pre-dive parameters

$GPS1	The first GPS fix from the start of the current dive. These values are from the first of two GPS fixes prior to the start of the current dive. Format: Date [ddmmyy], Time [hhmmss UTC], Latitude [+/- ddmm.mmm - only minuses are shown, positive north, degrees, minutes, and decimal minutes], Longitude [+/- dddmm.mmm - only minuses are shown, positive east], Time to first fix [seconds], HDOP (Horizontal Dilution Of Precision) - a measure of the strength of the figure used to compute the GPS fix, Total time to acquire fix (see $N_GPS), Magnetic variance [degrees, positive E], Estimated surface drift speed [knots], Estimated drift direction [degrees true], Number of satellites contributing to final fix, Horizontal position error [meters]
$_CALLS	Total number of calls that were made in an attempt to connect on this dive.
$_SM_DEPTHo	Glider measured depth at the end of the dive surface [meters].
$_SM_ANGLEo	Glider measured angle at the end of the dive on the surface [degrees].
$GPS2	These values are from the second GPS fix prior to the start of the current dive. See the "Canonical Dive Profile" for further details on where the GPS fix is taken. Format is as for $GPS1.

$SPEED_LIMITS	The minimum and maximum speed for the glider [m/s]. The maximum/minimum glider speed is based on the minimum/-maximum dive angles and the allowable buoyancy force. The minimal speed is the horizontal velocity corresponding to the maximum dive angle. The maximal speed is obtained as the minimal value of the horizontal speeds corresponding to the stall slope and to the maximum buoyancy force. The glide slope for the maximum buoyancy is approximated by a simple bisection step.
$TGT_NAME	The name of the target of this dive. See the section on **Target File** for more details.
$TGT_LATLONG	The latitude and longitude (in ddmm format) for the target of this dive.
$TGT_RADIUS	The radius for this target of this dive [m].
$KALMAN_CONTROL	A pair of floating point numbers that report the Kalman control vector. The units are meters per second. The first value is the glider speed to the east, the second value is the glider speed to the north.
$KALMAN_X	The first five elements (east-west oriented) components of the Kalman state vector. See Table D.5 for details.
$KALMAN_Y	The second five elements (north-south oriented) components of the Kalamn state vector. See Table D.5 for details.
$MHEAD_RNG_PITCHd_Wd	A seven element vector described in Table D.6.
$D_GRID	The depth which the bathymetry code uses as depth of the ocean for the glider's dive. The depth is extracted from the currently active bathymetry map and is based on the glider's starting position and the direction of the target [m].
$IRON	The hard-iron matrix currently used by the glider for compass calibration.
$MAGCAL	The soft-iron and hard-iron matrices estimated by the glider's automatic compass calibration procedure.
$OSC	Frequency of the system clock.

Table D.4: Logfile pre-dive pseudo-parameters

D.7.2 Dive

The dive section contains data that is related to the glider's performance during the actual dive. The section is denoted by the pseudo-parameter **$GCHEAD** and finished with the pseudo-

Element Number	Description
0	East position at time t|k due to mean current
1	East position at time t|k due to diurnal current
2	East position at time t|k due to semi-diurnal current
3	East position at time t|k due to glider speed through water
4	Y displacement from present position to predicted position due to mean, diurnal and semidiurnal components of the model
5	North position at time t|k due to mean current
6	North position at time t|k due to diurnal current
7	North position at time t|k due to semi-diurnal current
8	North position at time t|k due to glider speed through water
9	Y displacement from present position to predicted position due to mean, diurnal and semidiurnal components of the model

Table D.5: Kalman state vector pseudo-parameter description

Element Number	Description
0	Desired Heading (magnetic) for the glider to steer during the dive.
1	Distance (in meters) to the target being steered for.
2	Desired pitch angle for the glider to assume during the dive.
3	Desired vertical velocity for the glider (meters/second * 100) the dive.
4	Desired glide slope
5	dB/dw derivative (gain) for buoyancy adjustments (override with $DBDW parameter)
6	pressure sensor noise (RMS m)

Table D.6: $MHEAD_RNG_PITCHd_Wd pseudo-parameter description

parameter **$FINISH**. In between are a series of pseudo-parameter **$GC** lines, which report data that occurs during a GC phase.

The Dive section of the log file is bounded by the **$GCHEAD** pseudo-parameter and the **$FINISH** pseudo-parameter. In between is a series of **$GC** pseudo-parameters.

$GCHEAD: This parameter declares the column headers for the **$GC** lines that follow, enabling processing tools to key off the columns' meaning (as defined in the table below) rather then relying on the position.

$GC: Each **$GC** pseudo-parameter, or line, corresponds to one **G&C** phase. (See the **Canonical Dive Profile**.) The fields of the G&C line match the headers specified in Table D.7.

The individual entries in each **$GC** line are effectively divided into two separate groups. The first half of the line – all the entries through, but not including, the **end_secs** column – are written at the start of the G&C phase. The data from with **end_secs** to the end of the line is written out after the G&C phase is completed. In between the start and end of each line, it is possible for the glider to take one or more data readings that are reported in the data file. Correspondingly, the G&C phase takes a variable amount of time depending on the motor activity that happens.

$FINISH: The **$FINISH** pseudo-parameter has two values: Depth of glider at the first sample taken after reaching $D_SURF (or $D_FINISH, if enabled) [m], Density of water at the first sample taken after reaching $D_SURF (or $D_FINISH, if enabled) [g/cc].

D.7.3 Post-dive

The post-dive section provides data that either summarizes measures over the entire dive or is relevant to the end of the dive. Gliders will only report parameters relevant to its installed sensors.

(Pseudo) parameter	Description
$SM_CCo	A 7-element vector containing data about the surface maneuver pump: Time from the start of the dive to when the Surface Maneuver (SM) pump was started [s], Time for the SM pump [s], Average current for the VBD during the SM pump [A], Number of retries during the SM pump, Number of errors during the SM pump, Final position of the VBD after the SM pump [AD counts], Final position of the VBD after the SM pump [cc].
$SM_GC	G&C data from the surface maneuver pump (see Table D.7): depth, vbd_secs, pitch_secs, roll_secs, vbd_i, pitch_i, roll_i, vbd_ad, vbd_pot1_ad, vbd_pot2_ad,pitch_ad, roll_ad, vbd_errors, pitch_errors, roll_errors, vbd_volts, pitch_volts, roll_volts.
$SUPER	Supervisor status.

$IRIDIUM_FIX	Data from Iridium fix: latitude, longitude, date (ddmmyy format), time (hhmmss format).
$TCM_TEMP	Last temperature reading taken from the compass [degC].
$XPDR_PINGS	Data from transponder pings: number of pings, interrogate frequency [kHz], reply frequency [kHz].
$SC_FREEKB	Free space on scicon [kbytes].
$HUMID	Humidity inside the pressure hull [percent].
$TEMP	Temperature inside the pressure hull [degC].
$INTERNAL_PRESSURE	Pressure inside the pressure hull [PSIA].
$24V_AH	A two element vector. First element is the minimum measured battery voltage (measured during active phase) on the battery packs [V]. The second element is the total amp-hours consumed on the battery packs since the last reset of the battery meters (usually when new batteries are installed).
$10V_AH	Same as $24V_AH but measured differently, so value may be slightly different.
$FG_AHR_24Vo	Cumulative A-hr from the motors consumed as tracked by the supervisor fuel gauge reflects the fuel gauge state at the start of the dive.
$FG_AHR_10Vo	Same as $FG_AHR_24Vo but for the sensors and electronics.
$DEVICES	An pseudo-parameter that provides the column headers for the next two pseudo-parameters. Each of these columns refers to one of the tracked power states for the glider and generally refers to the energy consumed by a given device over the course of the dive. Some devices have multiple power states associated with them. See Table D.9 for details.
$DEVICE_SECS	Reports the time each device listed in $DEVICES was powered on during the dive [s].
$DEVICE_MAMPS	Reports the maximum current drawn by each device listed in $DEVICES [mA].

$SENSORS	Similar to $DEVICES, in simply providing titles for the numbers listed in the following two columns ($SENSOR_SECS and $SENSOR_MAMPS). Each title represents one of the sensors installed on the Seaglider, as described here. A sensor is defined as a device that is controlled via the **science** file (see Section D.3 for more details on the **science** file). The exact column order is dependent on the hardware configuration of the glider. See Table D.10 for details of the possible column headers. See the section on **Seaglider Configuration** for more details on glider configuration. For gliders with scicon installed, the 7th element in $SENSORS will be *SciCon* and the sensors that are controlled and logged by the scicon will not be included in $SENSORS.
$SENSOR_SECS	Reports time each sensor listed in $SENSORS was powered on during the dive [s].
$SENSOR_MAMPS	Reports the maximum current drawn by each sensor listed in $SENSORS during the dive [mA].
$DATA_FILE_SIZE	A two element pseudo-parameter that reports the total size of the data file [bytes] and the number of data samples taken during the dive.
$SDSIZE	A two element pseudo-parameter that reports the total size and available free space on the compact flash card [bytes].
$ERRORS	A pseudo-parameter that reports important errors from the current dive, where each value represents the number of each error type: pitchErrors, rollErrors, vbdErrors, pitchRetries, rollRetries, vbdRetries, GPS_line_timeouts, compass_timeouts, pressure_timeouts, sensor_timeouts[0], sensor_timeouts[1], sensor_timeouts[2], sensor_timeouts[3], sensor_timeouts[4], sensor_timeouts[5], logger_timeouts[0], logger_timeouts[1], logger_timeouts[2], logger_timeouts[3]. (_line_timeouts represents the number of times the GPS did not return data from the $GPRMC (position and time) or $GPGGA (fix data) within the 2 second timeout. Note: that these two sentences do not require that the GPS unit actually have a position fix to get a response back; the GPS unit simply needs to be on and functioning to respond.)
$CURRENT	Depth-averaged current speed [m/s], direction [degrees], and validity [0 or 1].

$IMPLIED_C_PITCH	A vector based on fit to pitch control vs pitch data: estimated $C_PITCH [AD counts], estimated $PITCH_GAIN, number of points in pitch regression.
$GPS	These values are from the most recent GPS fix, which corresponds to the end of the current dive. The format is as for $GPS1 and $GPS2.

Table D.8: Logfile post-dive parameters

Errors and retries

Errors and retries have specific meanings in the glider terminology depending on the hardware context. For motors (Pitch, Roll and VBD), a retry is declared when the observed rate of movement is less then a threshold value, as set by a parameter. A motor error is declared when a motor fails to achieve its commanded end position, as determined by the glider's internal control requirements. For the operations related to the flash card (refereed to as file operations), a retry is declared for each time the given particular file operation is retried after having failed. A file operation error is declared when a given file operations has been retried three times - at which point the file operation is terminated.

D.8 Capture file

The capture file contains the verbose output that the glider created during a dive. Capture files are a potential great source of information on the glider's performance, especially in error analysis and debugging, but generally need not be viewed a pilot. For more information on the use of capture files, please see the section on **Capture Files** in the users manual.

D.8.1 Format

The format of the capture file is not as hard and fast as other file formats. As a general rule, the file is comprised of single lines of the following form:

time,**service**,**outputlevel**,**text**

Where **time** is the time since the start of the dive expressed in seconds, the **service** and **outputlevel** are described in the **Seaglider Extended PicoDOS Reference Manual** under the **capvec** command. The **text** portion is a free form sequence of characters that conveys output that is logically associated with the specified **service** and **outputlevel**.

Column Name	Description
st_secs	Elapsed time from the start of the dive to the start of the GC [s]
flags	Bitmask value that represents the action that was occurring during a given GC phase:
vbd_ctl	Position of the VBD [cc]
pitch_ctl	Position of the pitch mass [cm]
roll_ctl	Position of the roll mass [degrees]
vbd_ad_start	Position of the VBD at the start of the move [AD counts]
vbd_pot1_ad_start	Position of linpot 1 at the start of the move [AD counts]
vbd_pot2_ad_start	Position of linpot 2 at the start of the move [AD counts]
pitch_ad_start	Position of the pitch mass at the start of the move [AD counts]
roll_ad_start	Position of the rol mass at the start of the move [AD counts]
depth	Depth at start of GC [m]
ob_vertv	Observed vertical velocity [cm/s])
data_pts	Number of data points collected thus far in the dive
end_secs	Elapsed time from the start of the dive to the end of the GC [s]
vbd_secs	Number of seconds the VBD was running
pitch_secs	Number of seconds the pitch motor was running
roll_secs	Number of seconds the roll motor was running
vbd_i	Average current used by the VBD [amps]
pitch_i	Average current used by the pitch motor [amps]
roll_i	Average current used by the roll motor [amps]
vbd_ad	Position of the VBD [AD counts]
vbd_pot1_ad	Position of linpot 1 [AD counts]
vbd_pot2_ad	Position of linpot 2 [AD counts]
pitch_ad	Position of the pitch motor [AD counts]
roll_ad	Position of the roll motor [AD counts]
vbd_errors	Number of VBD errors (timeouts) during the motor move
pitch_errors	Number of pitch errors (timeouts) during the motor move
roll_errors	Number of roll errors (timeouts) during the motor move
vbd_volts	minimum voltage recorded during VBD portion of move [V]
pitch_volts	minimum voltage recorded during pitch portion of move [V]
roll_volts	minimum voltage recorded during roll portion of move [V]

Table D.7: $GCHEAD pseudo-parameter description

Element	Description
Pitch_motor	All use of the pitch motor.
Roll_motor	All use of the roll motor.
VBD_pump	All use of the VBD pump.
Iridium	All use of the phone.
Transponder_ping	Use of the transponder during an active ping.
GPS	All use of the GPS for fix acquisition
Core	All use for main processor at normal operation
Fast	All use for main processor running fast
Slow	All use for main processor running slow
LPSleep	Use of the motherboard under low-power sleep
Compass	Use of the compass
Compass2	Use of the second compass, if installed
RAFOS	Use of the RAFOS receiver
Transponder	Total use of the transponder (including ping time)
Network	Use of the network acoustic modem

Table D.9: $DEVICES pseudo-parameter

D.8.2 Format

The GPS file contains at least on pseudo-parameter and possibly two others, in the same format as the described in Section D.7 on Logfile parameters. The one that is always present is the **$GPS** pseudo-parameter.

If the Seaglider is in Recovery, then there are two additional pseudo-parameters written to the GPS file - **$EOP_CODE** and the **$RECOV_CODE**. The **$EOP_CODE**, which stands for End Of Phase, is an internal state variable the Seaglider uses to indicate how a particular phase of the dive completed. If the Seaglider entered recovery due to a problem as part of the dive policy, this code will indicate what the problem was. See the description of the **Canonical Dive Profile** (Section 2.2.1) for details on the specific phases. Table D.11 describes the possible EOP codes. where the table lists specific parameters, please refer to that parameter's documentation for a complete description of the impact on the gliders behaviour.

The **$RECOV_CODE** pseudo-parameter describes the reason the Seaglider is in recovery. The recovery code generally contains information on non-dive policy related errors for a Seaglider entering recovery, although there are a few cases that overlap with the **$EOP_CODE**. Table D.12 describes the possible Recovery Codes.

Sensor Value	Description
SBE_CT	Seabird CT sensor. By convention, this is configured as the first device.
SBE_O2	Seabird O2 sensor. By convention, this is configured as the second device.
WL_BB2FL	Wetlabs BB2FL combination scattering meter and fluorometer
AA4831	Aanderaa 4831 optode oxygen sensor
QSP2150	QSP2150 PAR sensor
nil	Indicates no sensor is installed in this position

Table D.10: $SENSORS pseudo-parameter

EOP Code	Description
CONTROL_FINISHED_OK	No problem
TARGET_DEPTH_EXCEEDED	Glider exceeded $D_TGT
SURFACE_DEPTH_REACHED	Glider reached $D_SURF
FINISH_DEPTH_REACHED	Glider reached $D_FINISH
ABORT_DEPTH_EXCEEDED	Glider exceeded $D_ABORT
FLARE_DEPTH_REACHED	Glider reached $D_FLARE
NO_VERTICAL_VELOCITY	Glider's vertical velocity
HALF_MISSION_TIME_EXCEEDED	Glider did not reached apogee before $T_MISSION/2 minutes
UNCOMMANDED_BLEED_DETECTED	VBD bleed occurred without being commanded by the glider
MOTOR_MAX_ERRORS_EXCEEDED	Maximum number of motor errors exceeded (see $PITCH_MAXERRORS and $ROLL_MAXERRORS)
CF8_MAX_ERRORS_EXCEEDED	Maximum number of motor errors exceeded (see $CF8_MAXERRORS)
BOTTOM_OBSTACLE_DETECTED	If altimeter in use for bottom detection, bottom detected
SURFACE_OBSTACLE_DETECTED	If altimeter in use for surface detection, surface detected
ABORT_TIME_EXCEEDED	Time greater than $T_ABORT
LOITER_COMPLETE	Loiter complete
POWER_ERROR_DETECTED	Power error detected
SENSOR_ERROR_DETECTED	Sensor error detected

Table D.11: $EOP_CODE pseudo-parameters

Recovery Code	Description
NO_RECOVERY_REASON	There was no recovery reason specified by the Seaglider. Normally, this is not seen and represents on internal error in the Seaglider software.
QUIT_COMMAND	The pilot placed a **$QUIT** command in the cmd-file, sending the glider into recovery.
EXHAUSTED_24V_PACK	The recorded power draw has exceeded pilot-specified capacity of **$AH0_24V**.
EXHAUSTED_10V_PACK	The recorded power draw has exceeded pilot specified capacity of **$AH0_10V**.
WATCHDOG_RESET	The Seaglider has rebooted due to a watch dog time out (**$T_WATCHDOG**) and the **$RE-LAUNCH** was set to 0.
END_OF_SCENARIO_RUNS	The Seaglider has completed it scenario runs. This recovery code is only seen when the Seaglider is in testing (scenario) mode.
GPS_DEAD	The GPS unit failed to turn on when the Seaglider commanded it to do so.
NO_TARGETS	There was no parsable **targets** file on the glider and the **$HEADING** parameter was not active when attempting to launch the glider.
NO_SCIENCE	There was no parsable **science** file on the glider when attempting to launch the glider.
EXCEEDED_ABORT_DEPTH	The pilot-set depth not to exceed (**$D_ABORT)** was exceeded.
UNCOMMANDED_BLEED	There was movement of the VBD in the bleed direction that exceeded the pilot specified value (**$UNCOM_BLEED**) while another motor was running.
MAX_VBD_ERRORS	The pilot specified number of allowable errors in the VBD (**$VBD_MAXERRORS**) was exceeded.
NO_RAFOS_SCHEDULE	A RAFOS enabled glider did not have a valid RAFOS schedule on board at launch time.
UKNOWN_RECOVERY_REASON	The Seaglider went into recovery for no known reason. This is not normally seen and represents an internal software error.

Table D.12: $RECOV_CODE pseudo-parameters

Bibliography

Eriksen, C. C., Osse, T. J., Light, R. D., Wen, T., Lehman, T. W., Sabin, P. L., Ballard, J. W., and Chiodi, A. M.: Seaglider: A long-range autonomous underwater vehicle for oceanographic research, IEEE Journal of oceanic Engineering, 26, 424–436, 2001.