Slocum G2 Glider Operators Manual

P/N 4343, Rev. B
January 2012

Teledyne Webb Research
82 Technology Park Drive
East Falmouth, MA 02536
U.S.A.
Tel: +1 (508) 548-2077
Fax: +1 (508) 540-1686

www.webbresearch.com

TELEDYNE
WEBB RESEARCH
A Teledyne Technologies Company

Contents

1 Introduction 1-1
Forward Propulsion 1-2
Navigation and Flight 1-3
Safety and Handling Procedures 1-3
Architecture and Components 1-3
Architecture 1-3
Nose Dome 1-4
Forward Hull Section 1-4
Payload Bay Mid-Hull Section 1-5
Aft Hull Section 1-5
Aft Tail Cone 1-5
Wings 1-6
Components 1-6
Ballast Pump Assemblies 1-6
Pitch Vernier 1-9
Altimeter 1-10
Carbon Fiber Hulls 1-13
Science Bay Computers and Sensors 1-13
Conductivity, Temperature, and Depth (CTD) 1-14
Argos Satellite Platform Transmitter Terminal (PTT) 1-15
Catalyst 1-16
Air Pump System 1-16
Vehicle Controller 1-17
Hardware Interface Board 1-19
Attitude Sensor 1-19
Global Positioning System (GPS) 1-20
Iridium Satellite Telemetry 1-21
Radio Frequency (RF) Modem Telemetry 1-22
Science/Payload Computer Communications 1-23
Pressure Transducer 1-23
Leak Detect Sensor Boards 1-24
Air Bladder 1-24
Burn Wire and Jettison Weight 1-25
Power Umbilical 1-26
Digital Tail Fin (digifin) 1-26
Sacrificial Anode 1-27
Batteries/Coulomb Counter 1-28
Desiccant 1-30
Pick Point 1-30
Recovery System (optional) 1-30
Strobe (optional) 1-31

2 Pre-mission Testing 2-1
On the Beach, Boat, or Bench 2-1
In the Water 2-2

3 Deployment and Recovery 3-1
Deploying the Glider 3-1
Large Ship Deployment 3-4
Recovering the Glider. 3-5

4 Emergency Recovery. 4-1

5 Glider Communications. 5-1
Communicating with the RF Modem 5-1
Communicating with Argos 5-1
Communicating with Iridium 5-1
Communicating with RUDICS. 5-2

6 Software Architecture 6-1
PicoDOS 6-2
Config Folder 6-2
Bin Folder 6-3
Logs Folder 6-4
Missions Folder 6-4
Sentlogs Folder 6-4
Autoexec.bat 6-5
GliderDOS 6-5
Masterdata 6-6
Prefixes. 6-7
Sensor Commands 6-7
Device Commands 6-8
Science Computer. 6-11
Bin Folder 6-12
Config Folder 6-13
Autoexec.bat 6-14

7 Science Data Logging 7-1
System Requirements 7-1
Performance 7-1
Logfile Types 7-1
Configuration Files 7-2
Transparency 7-2
Exceptions to Transparency 7-3
Control 7-4

8 Flight Data Retrieval. 8-1
Vehicle Status: Glider on Surface Counting Down to Resume Mission 8-2

9 Dockserver Data Visualizer 9-1

10 Missions 10-1
.mi Files 10-1
.ma Files 10-1
Running Missions 10-2

A Abbreviations and Acronyms A-1

B Code Theory and Operation B-1
Operating Systems B-1
Code Design B-1
Control Levels B-2
Control: Stacks, States, and Abort Sequences B-2
General Control Structure B-3
Synchronous Abort B-3
Out of Band Abort B-4
Hardware Generated Abort B-4
Sample Mission and Comments B-4

C Abort Codes C-1

D Glider Software Website D-1

E Ancillary Equipment E-1

F FreeWave Configuration F-1
About FreeWave Transceivers F-1
Setting up the Glider Shoreside FreeWave F-1
Setting up the Glider FreeWave Slave (internal to the glider) F-2
Choosing a Location for the Transceivers F-3

G Iridium Service and the SIM Card G-1

H Argos Satellite Service and ID H-1
Instructions for Completing the Argos Technical Information Form H-1
Argos Data Format H-2

I How to Determine Mission Battery Longevity I-1

J How to Edit a Proglets.dat File J-1

K Quick Reference and Checklists K-1

L Glider Wiring Diagram L-1

M G2 Glider/Science Simulator User's Guide M-1
Introduction M-7
Format Notes M-7
Connections M-8
Front Panel M-10

Connecting the Simulator to the Dockserver Hardware M-11
Connecting the Simulator with a Direct Serial Cable M-11
Connecting the Simulator with a FreeWave Transceiver M-11
Connecting the Simulator with an Iridium Modem M-12
Differences between a Simulator and a Glider M-12
Simulation M-13
Levels of Simulation M-13
simul.sim file M-13
Simulating the Iridium Modem M-14
Simulating the FreeWave Modem M-14
Simulating Bad Input M-14
Simulation Details M-16
on_bench M-16
on_bench open M-16
just_electronics M-16
no_electronics M-17
Setting Initial Simulation Parameters M-17
Simulation in the Science Bay M-17
Software Control Hierarchy M-18
Glider Computer M-18
Config Folder M-19
Bin Folder M-19
Logs Folder M-20
Missions Folder M-20
Mafiles Folder M-20
Sentlogs Folder M-21
Autoexec.bat M-21
PicoDOS M-21
GliderDOS M-21
Masterdata M-23
Prefixes M-23
Sensor Commands M-23
Device Commands M-24
Science Computer M-27
Bin Folder M-29
Config Folder M-29
Autoexec.bat M-30
Science Data Logging M-30
System Requirements M-30
Performance M-30
Logfile Types M-31
Transparency M-32
Exceptions to Transparency M-32
Control M-32
Code Theory and Operation M-35
Operating Systems M-35
Code Design M-35
Control Levels M-36

Control: Stacks, States, and Abort Sequences. M-36
General Control Structure M-37
Synchronous Abort M-38
Out of Band Abort M-38
Hardware Generated Abort M-38
Sample Mission and Comments M-39
Glider Software Website M-44

Preface

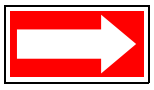
This manual provides the information required to operate the Slocum Glider G2 System. This manual is used in conjunction with the *Slocum Glider Maintenance Manual*. It is divided into the following sections:

- **Section 1—Introduction** provides a general description of the Slocum G2 glider and an overview of its architecture and components.
- **Section 2—Pre-mission Testing** describes the in- and out-of-water tests that should be conducted to qualify a glider before delivery or with new or modified software.
- **Section 3—Deployment and Recovery** provides tips and reference material to assist in deploying and recovering gliders.
- **Section 4—Emergency Recovery** provides tips for recovering a glider during an emergency.
- **Section 5—Glider Communications** includes steps for setting up communications with the glider from various sources, such as a radio frequency (RF) modem, Argos, Iridium, and RUDICS. Steps for setting up direct communication with the glider are also provided.
- **Section 6—Software Architecture** describes the structure of the glider's software: software control hierarchy, folder structure, commands, and masterdata prefixes.
- **Section 7—Science Data Logging** provides an introduction to science data logging and how this functionality works.
- **Section 8—Flight Data Retrieval** is a guide for retrieving data from the glider during a mission surfacing and when a mission has ended.
- **Section 9—Dockserver Data Visualizer** provides a brief description of the Data Visualizer and a link to documentation on how to use it.
- **Section 10—Missions** provides instructions for loading mission files onto a glider and running missions.
- **Appendix A—Abbreviations and Acronyms** provides a list of abbreviations and acronyms related to glider maintenance and operations.
- **Appendix B—Code Theory and Operation** provides details on the software used to operate the glider. Topics include a description of the operating systems used, the code design, abort sequences, general control structure for glider deployments, and sample code for a mission.
- **Appendix C—Abort Codes** contains a list of codes used to abort glider tasks and operations.
- **Appendix D—Glider Software Website** provides instructions for downloading the glider code from the repository.
- **Appendix E—FreeWave Configuration** includes excerpts from FreeWave Techonologies' *Spread Spectrum Users Manual* on how to set up FreeWave transceiver communications on a glider.
- **Appendix F—Iridium Service and the SIM Card** contains a list of companies who sell the Iridium SIM card, general billing and activation information, and how to determine which usage plan is the best option.

- **Appendix G—Argos Satellite Service and ID** provides instructions on how to set up Argos satellite service and an ID number.
- **Appendix H—Argos Data Format** provides links to reference material with the Argos data format for the gliders, along with tools to automate unpacking the Argos data.
- **Appendix I—How to Determine Mission Battery Longevity** provides information about how the length of a mission battery is determined and details on how to monitor the battery voltage.
- **Appendix J—How to Edit a Proglet.dat File** provides instructions for making changes to proglet.dat.
- **Appendix K—Storage Conditions** provides the temperature ranges for storing gliders.
- **Appendix L—Quick Reference and Checklists** contains a link to the latest copy of the *Slocum G2 Glider Operators Manual*, along with the quick reference guide and checklists used in operating the glider.
- **Appendix M—G2 Glider/Science Simulator User's Guide** contains a copy of this document.

Notes and Warnings

Where applicable, special notes and warnings are presented as follows:



NOTE A referral to another part of this manual or to another reference; a recommendation to check that certain criteria are met before proceeding further in a step or sequence; or general information applicable to the setup and operation of the Teledyne Webb Research Slocum G2 Glider.

CAUTION A reminder to follow certain precautions in order to prevent damage to equipment or injury to personnel.

WARNING A reminder that dangerous or damaging consequences could result if certain recommended procedures are not followed.

Format Notes

Glider sensors and commands will be denoted in the Courier font throughout this document, as shown in the example below:

Typing `Report ++ m_roll` will report measured roll (`m_roll`) every four seconds.

When displayed on a PC, some areas will be hyperlinked to information available on the Internet, such as:

http://www.webbresearch.com/

and protected documents by permission:

http://www.glider.webbresearch.com/

Many of the links and the code mentioned in this manual require access by prior arrangement. Please contact glidersupport@webbresearch.com to inquire about access to these protected documents.

Customer Service

We welcome your comments and suggestions for improving our products, documentation, and service of the glider system. Please contact Glider Support should you have any comments or suggestions about this manual, the glider system, or if you require service or support.

Please contact us at:

82 Technology Park Drive
East Falmouth, MA 02536

Telephone: +1 (508) 548-2077
Fax: +1 (508) 540-1686
E-mail: glidersupport@webbresearch.com

www.webbresearch.com

1 Introduction

Conceived by Douglas C. Webb and supported by Henry Stommel and others, the class of Slocum gliders is named after Joshua Slocum, the first man to single-handedly sail around the world. These gliders are innovative autonomous underwater vehicles with two primary designs: a 200-meter coastal glider and a 1000-meter glider. A third design in development is the long-range thermal glider. Each of the two existing gliders is specifically designed to maximize littoral or deep ocean capabilities with ranges from 4 to 200 meters for the 200-meter glider and 40 to 1000 meters for the 1000-meter glider. These platforms are a uniquely mobile network component capable of moving to specific locations and depths, and occupying controlled spatial and temporal grids. The gliders are driven in a sawtooth vertical profile by variable buoyancy and can move horizontally and vertically.

Long-range and satellite remote sensing systems are being realized in the ocean measurement field. These systems are being used to quantify currents, sea surface height, temperature, and optical properties of the water, enabling modeling and prediction of ocean state variables in the littoral zone. A similar nested grid of subsurface observations is required to maximize the impact and ground-truth of the more extensive surface remote sensing observations.

The long-range capabilities of the Slocum gliders make them ideally suited for subsurface sampling at a regional or larger scale. These gliders can be programmed to patrol for weeks or months at a time, surfacing to transmit their data to shore while downloading new instructions at regular intervals, at a substantial cost savings compared to traditional ship-based research methods.

The small relative cost and the ability to operate multiple vehicles with minimal personnel and infrastructure enables small fleets of gliders to study and map the dynamic (temporal and spatial) features of our subsurface coastal or deep ocean waters 24 hours daily, 365 days a year.

Forward Propulsion

Gliders are unique in the autonomous underwater vehicle (AUV) world, because their varying vehicle buoyancy creates forward propulsion. Wings and control surfaces convert the vertical velocity into forward velocity so that the vehicle glides downward when denser than water and glides upward when buoyant (see figure below, which is representative of a 200-meter glider). Gliders require no propeller and operate in a vertical sawtooth trajectory.

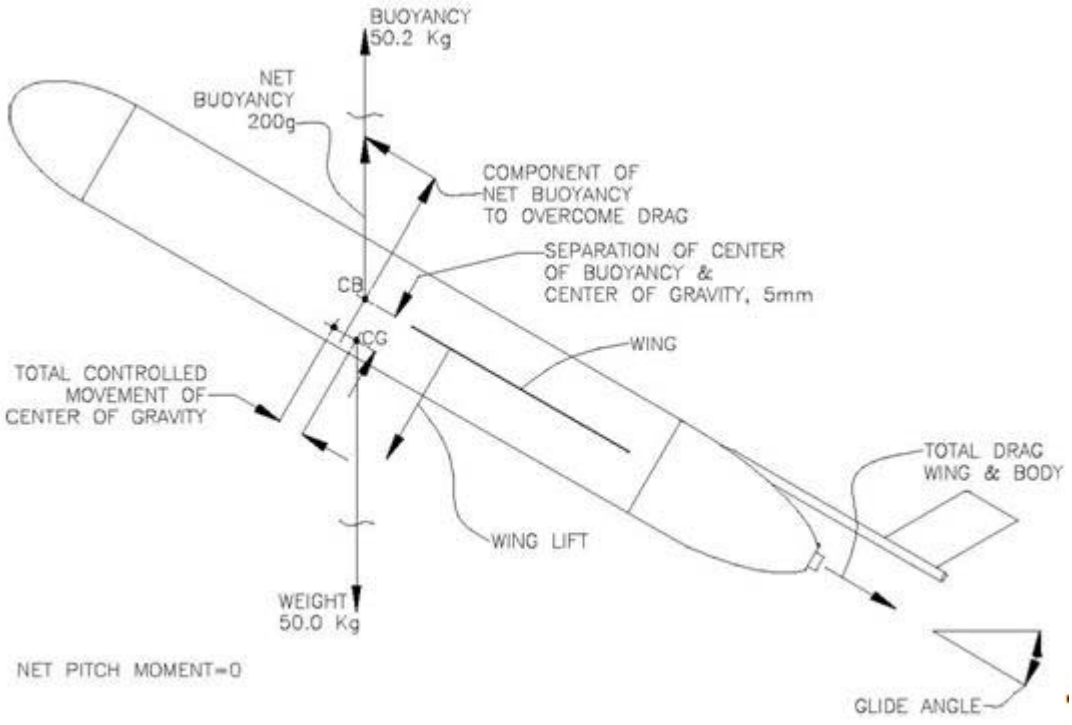


Figure 1-1 Force balance diagram of forces acting on the glider (angle of attack not included).

Navigation and Flight

The Slocum Glider navigates to waypoints via dead reckoning, inflecting at depths and altitudes as prescribed in a text mission file. As set by the mission, the glider periodically surfaces to obtain a GPS location fix and to communicate data and instructions with glider operators via satellite. Differences between its estimated dead reckoning position and its GPS position are attributed to currents, which are accounted for in subsequent mission segments via set and drift calculations.

For more information about the glider's flight dynamics, go to:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/how-it-works

This site includes various definitions, including how the glider dead reckons.

Safety and Handling Procedures

For information on safety and handling procedures for the Slocum Glider, see the "Warnings and Precautions" section of the *Slocum G2 Glider Maintenance Manual*.

Architecture and Components

Architecture

The Slocum Glider comprises five total sections (three main hull sections and two wet sections located fore and aft), a design which was chosen for its simplicity, economy, and expandability. The 200-meter glider's main hull sections are made of 6061 T6 aluminum alloy and have an outside diameter of 21.3 cm. The 1000-meter glider's main hull sections have an outside diameter of 22 cm and are manufactured from composite material or 6061 T6 aluminum. The nose end cap is a machined pressure resistant elliptical shape, and the tail cap is a truncated cone. Composite wings are swept backward at 45 degrees and are easily replaced using a quick release system.

WARNING Take care while removing and installing wings as they are not buoyant and will sink if dropped.

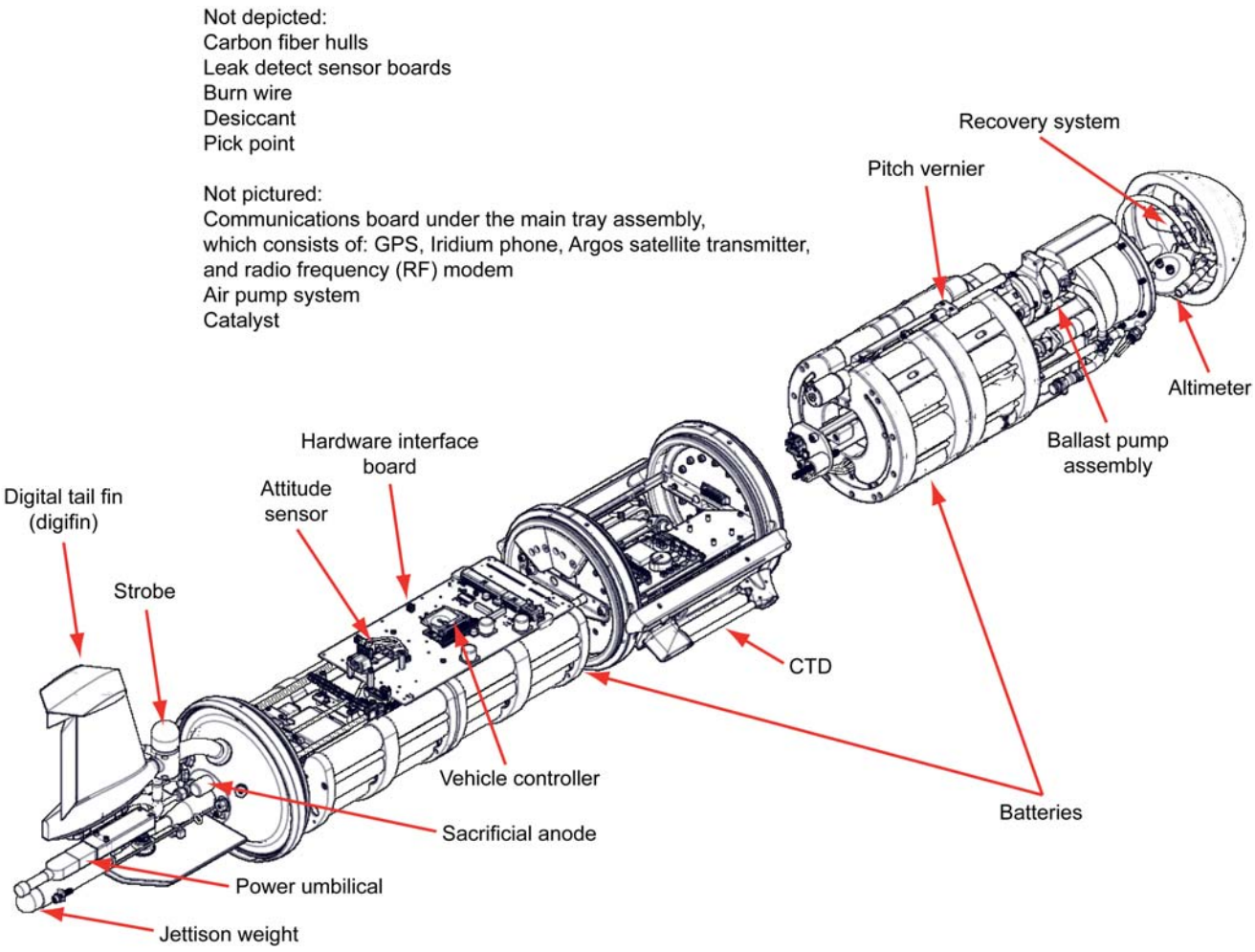


Figure 1-2 Components of the Slocum G2 Glider.

Nose Dome

The nose dome is an acoustically transparent plastic cap located at the front of the glider that protects the altimeter and ballast pump assemblies. The hole in the center of the nose dome allows the ballast pump to draw water in or push water out to change the vehicle's buoyancy. Some gliders are also outfitted with an optional nose dome recovery assembly.

Forward Hull Section

This section houses the ballast pump assembly, pitch vernier, forward batteries, and ballast weights. The pitch vernier is used to move the batteries forward or backward to adjust the glider's pitch. Internal wiring connectors that attach to the payload bay are located at the aft end of the ballast pump assembly.

Payload Bay Mid-Hull Section

The payload bay mid-hull section is capable of accommodating a flexible science payload where a variety of instruments can be easily removed and replaced. This section is 12 inches long, comprising two rings and a hull section, with a nominal capacity of 3 to 4 kg. The front ring is typically ported for the conductivity, temperature, and depth (CTD) sensor assembly, but it can be fitted with a penetrating connector to accommodate a variety of science sensors, such as the Rockland MicroRider.

The standard G2 glider has a payload bay computer ("persistor") that is connected to the glider's main computer. This persistor controls the sensor packages and collects and stores data via proglets.dat, an ASCII file that resides in the config directory. Each glider and glider type has a unique proglets.dat file that depends on the current configuration of scientific instruments. The original source document can be found here:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/target-science/config/proglets.dat

Depending on the configuration of science instruments in the payload bay, ballast weight can be added to this section. H-moment adjustments can also be made by moving weight, high or low, in this section of the vehicle.

Aft Hull Section

The aft hull section houses the main glider electronics and the chassis that ties the vessel together. The main controller board (upper electronics board) carries the flight computer (main persistor) and attitude sensor. The communications board, which is located beneath the main controller board, contains the Argos transmitter, GPS, Iridium modem, and radio frequency (RF) modem. The air pump system and a catalyst are attached to the underside of the chassis. This catalyst recombines hydrogen gas and oxygen into water to reduce the risk of explosion in the event that alkaline batteries are shorted or come into contact with water or sea water. The aft battery pack, which is not attached to the chassis, is positioned below the chassis in the bottom half of the aft hull section. Unlike the forward battery pack, the aft battery pack is fixed in place and is not used to control the glider's pitch. Behind the aft battery pack, the micron pressure transducer is ported through the aft end cap.

Aft Tail Cone

The aft tail cone is a wet area that houses the air bladder, burn wire, jettison weight, and power umbilical, all of which are connected to the aft channel, which attaches to the center of the aft end cap. This area can be modified to include additional devices, such as a strobe light and/or science sensors. The digital tail fin (digifin), which protrudes from the aft end cap through the aft tail cone, contains antennas for the GPS system, Iridium system, Argos transmitter, and radio frequency transmitter. The digifin is sturdy enough so that the glider can be lifted by it. However, it should be noted that the aft channel, which is located just below the digifin, is not sturdy and can be bent if grasped while liftin ghte glider.

Wings

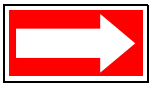
The standard carbon fiber wings for the G2 Slocum Glider are designed with a sweep angle of 45° which is suitable for both shallow and deep gliders. The G2's wings are located aft of the center of buoyancy and provide pitch stability during flight. Wings are attached to the glider via a quick release system that clicks into place. The quick release mechanism is located near the aft end of the wing rail. Adjustable-ballast wing rails, designed to expedite the field ballasting process, are now available for purchase. These wing rails are slotted to accommodate up to ten 14g stainless steel masses.

Components

CAUTION Some devices and equipment cannot or should not be tested while the glider is deployed and can only be tested in the lab. The glider should never be put into lab_mode while deployed. Properly trained personnel or glider pilots should be consulted before any of the testing below is attempted. Please contact glidersupport@webbresearch.com regarding questions or for clarification, or to troubleshoot unexpected test results. When glider sensors are not defined below and for default values, refer to the masterdata.

ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

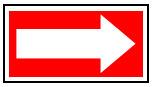
Ballast Pump Assemblies



NOTE Both deep pumps (stainless steel bellows) and shallow pumps (rolling diaphragm or bellophragm) have a 10000 cycle service life or 20000 m_tot_num_inflections (to full depth).

200-meter Ballast Pump Assembly

The 200-meter ballast pump assembly is a single-stroke piston design that uses a 90-watt motor and a rolling diaphragm seal to move 460 cc of sea water directly into and out of a short 12-mm diameter port on the nose centerline (the stagnation point). The pumps for different motor types (30-, 100- and 200-meter) are rated for different pressures based on the gearbox associated with the motor. The mechanical gear drive is not the limiting factor; it is the maximum amount of energy that must be pulled from the battery source. The selection of gearbox/motor assembly should be optimized for the working depth to allow for quick inflections (more important in shallow water) and to minimize energy used on the return stroke.



NOTE The displacement pump should not be run without either external pressure or internal vacuum on the rolling diaphragm. The vacuum inside the glider should be drawn to 6 inHg less from the external atmosphere to ensure that the diaphragm folds smoothly as it rolls, otherwise damage may result. To eliminate back drive of the pump at pressure, a latching brake is used to hold the motor when at rest.

Ratio	Pmax @ 6 Amp	Rated Pressure	Flow No Load
156:1	248 dbar	200 dbar	24 cc/sec
74:1	135 dbar	100 dbar	43 cc/sec
26:1	41 dbar	30 dbar	126 cc/sec

How to Configure

Factory configured. Please contact glidersupport@webbresearch.com for assistance.

How to Test

From `lab_mode`:

1. Type `wiggle on`.
2. Type `report ++ m_ballast_pumped`.
3. Make sure that the ballast pump successfully completes a full extension (`m_ballast_pumped` = +230 cc) and full retraction (`m_ballast_pumped` = -230 cc) without errors.
4. Type `wiggle off`.

How to Evaluate Data

With a properly ballasted glider, positive cc's should result in a positive buoyancy and a climb. Negative cc's will result in a negative buoyancy and a dive.

Relevant Sensors

Sensor Name	Description
m_ballast_pumped	Measured volume of buoyancy pumped in cc's
c_ballast_pumped	Commanded volume of buoyancy pumped in cc's

1000-meter Ballast Pump Assembly

The 1000-meter ballast pump is a rotary displacement design that moves oil from an internal reservoir to an external bladder to change the vehicle's buoyancy. It is necessary that the glider be under vacuum while operating the ballast pump, because the force of the internal vacuum is used to draw oil from the bladder back into the body of the glider. Although an operator will not damage the pump by running it without a vacuum, the bladder will be damaged. A rotary valve is used to control the flow of oil from the bladder to the reservoir.

Ratio	Pmax @ 7 Amp	Rated Pressure	Flow No Load
N/A	1240 dbar	1000 dbar	5 cc/sec

How to Configure

Factory configured. Please contact glidersupport@webbresearch.com for assistance.

How to Test

From `lab_mode`:

1. Type `wiggle on`.
2. Type `report ++ m_de_oil_vol.`
3. Make sure that the ballast pump successfully completes a full extension (`m_ballast_pumped` = +270 cc) and full retraction (`m_ballast_pumped` = -270 cc) without errors.
4. Type `wiggle off`.

How to Evaluate Data

With a properly ballasted glider, positive cc's should result in a positive buoyancy and a climb. Negative cc's will result in a negative buoyancy and a dive.

Relevant Sensors

Sensor Name	Description
m_de_oil_vol	Measured volume of buoyancy pumped in cc's
c_de_oil_vol	Commanded volume of buoyancy pumped in cc's

Pitch Vernier

Provided that the H moment is 6 mm ±1 (see "H Moment Calculation" and "Adjusting the H Moment" in section 3 of the *Slocum Glider Maintenance Manual*), the fluid movement from the ballast pump assembly provides the moment for changing pitch. (Water moves into the nose making the vehicle nose heavy when diving, similarly making the nose buoyant when rising.) To trim to the desired dive and climb angles, a lead screw drives the forward ~10 kg battery pack fore or aft as a vernier. The vehicle is designed to climb and descend at an angle of 26 degrees. During surfacing, the battery pack is moved all the way forward to better raise the tail out of the water for communications.

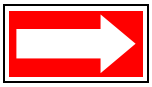
How to Configure

Factory configured. Please contact glidersupport@webbresearch.com for assistance.

How to Test

From `lab_mode`:

1. Type `wiggle on`.
2. Type `report ++ m_battpos`.
3. Make sure that the pitch battery successfully completes a full extension and retraction without errors.
4. Type `wiggle off` and `report clearall`.



NOTE The length of a full extension and retraction depends on the type of buoyancy pump (200-meter or 1000-meter) and battery (alkaline or lithium) as shown in the table below. The value of `f_battpos_safety_max` is specified in the glider's autoexec.mi file.

Pump and Battery Configuration	f_battpos_safety_max (inches)
1000 m, lithium	±0.9
1000 m, alkaline	±1.6
200 m, lithium	±1.1
200 m, alkaline	±1.6

How to Evaluate Data

With a properly ballasted glider, positive pitch battery movement will result in decreased angle of attack and negative pitch battery movement will result in increased angle of attack (i.e., moving the battery toward the aft of the vehicle will lift the nose and moving the battery forward will bring the nose down).

Relevant Sensors

Sensor Name	Description
m_battpos	Measured position of the battery in inches
c_battpos	Commanded position of the battery in inches
m_pitch	Measured vehicle pitch
c_pitch	Commanded vehicle pitch

Altimeter

The Airmar altimeter, with a 0-100 m range transducer, is mounted on the front of the ballast pump assembly, and its electronics are supported on the cylinder of the ballast pump assembly. The transducer leads feed through a bulkhead connector on the front end cap. The transducer is oriented so that it is parallel to a flat sea bottom at a nominal dive angle of 26 degrees.

How to Configure

Factory configured. Please contact glidersupport@webbresearch.com for assistance.

How to Test

In the lab:

1. Type `put c_alt_time 0` to ensure that the altimeter is updating as frequently as possible.
2. Listen closely to the altimeter to check for audible clicks similar ot the ticking of a watch.

3. Type `report ++ m_altimeter_voltage`, and confirm that the altimeter voltage is not a fixed value.
4. Type `report clearall`.

How to Evaluate Data

The altimeter is normally used only while diving to prevent the glider from hitting the sea floor. When the altimeter is in use, the glider dives until it reaches a set distance from the bottom; then it begins to climb. This distance from the bottom is specified in either the glider's mission or ma files as an argument for the `d_target_altitude` command. Each buoyancy pump design and mission requirement will dictate the necessary value to achieve successful bottom avoidance. The suggested minimum values for `d_target_altitude` are:

Pump Type	Suggested minimum value for `d_target_altitude`
1000 m	12 m
200 m	6 m
100 m	4 m
30 m	2 m

Relevant Sensors

Sensor Name	Description
`m_altitude(m)`	Height above the bottom
`m_raw_altitude(m)`	Height above bottom, unfiltered
`u_alt_min_depth(m)`	How deep the vehicle must be before the altimeter turns on A glider pilot may choose to increase `u_alt_min_depth` so that the altimeter will not turn on until it is below the thermocline or to reduce energy usage when the minimum water depth for the mission is known.
`u_alt_reqd_good_in_a_row(nodim)`	How many `m_raw_altitude` readings are used to calculate the processed `m_altitude` measurement
`u_alt_filter_enabled(bool)`	Enable median filter depth for altitude

Sensor Name	Description
u_alt_reduced_usage_mode(bool)	The glider calculates and uses the altimeter only when necessary. A glider pilot might decide to activate or deactivate reduce usage mode. Reduced usage will save energy; however, it may result in ineffective results in shallow water deployments.
m_water_depth(m)	Calculates the depth of the water water by adding m_depth and m_altitude
u_max_water_depth_lifetime(yos)	How long to use m_depth only in absence of new altimeter fixes

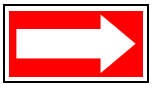
Carbon Fiber Hulls

How to Configure

Carbon fiber hulls are custom wound, compressible, engineered parts and should be handled with care. Any nicks in the paint finish should be examined to ensure that damage does not extend into the fiber winding. Careful inspection of interior and exterior finishes should be performed regularly.

How to Test

Pull and maintain the recommended vacuum on the vehicle (6 inHg for 200-meter gliders and 7 inHg for 100-meter gliders). For more details on adjusting the vacuum, see “Checking and Setting the Vacuum” in Section 4 of the *Slocum G2 Glider Maintenance Manual*.



NOTE The vacuum will fluctuate with the temperature.

How to Evaluate Data

The vacuum should be monitored. Note that the vacuum is affected by temperature (increases with cooling and decreases with warmth).

Relevant Sensors

`m_vacuum`—Measured internal vaccuum

Science Bay Computers and Sensors

How to Configure

A proglet controls each science sensor. The proglet sets the power bit and uart on the science bay controller and expects formatted data from each sensor.

How to Test

From GliderDOS, type `loadmission sci_on.mi`. All science bay sensors will be turned on and their data will be printed to the terminal. When it has been determined that all sensors are reporting data properly, type `loadmission sci_off.mi` to turn off the science sensors.

Each sensor can be addressed directly using u4stalk. Contact glidersupport@webbresearch.com for details.

How to Evaluate Data

The method of evaluating the data is unique to the sensors contained within each payload bay.

Relevant Sensors

The relevant sensors are unique to each payload bay.

Conductivity, Temperature, and Depth (CTD)

A Slocum glider does not require a CTD; however, a typical sensor package contained in the payload bay on the glider is a Sea Bird (SBE) pumped conductivity, temperature, and depth package. The CTD sensor is delicate and should be protected from abuse. For a 200-meter glider, a 500-PSI pressure transducer is used for the depth measurement. For a 1000-meter glider, a 2900-PSI pressure transducer is used for depth measurement. The SBE electronics and sensors are calibrated as a single unit.

The CTD is not used for flight, because the glider has an independent pressure sensor for dynamic flight control. It can be used in an emergency by changing the sensor `u_use_ctd_depth_for_flying`.

WARNING The Seabird Electronics (SBE) pumped CTD should not be run dry for more 30 seconds at a time.

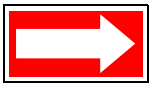
The Slocum Glider data page is located at http://www.seabird.com/products/profilers.htm. Scroll down to the bottom of this page to access the link to the data page.

How to Configure

Manufacturer and factory configured

How to Test

The steps to turn on the science sensors listed above can be used to evaluate the CTD data. When dry and in the lab, the measured conductivity will be 0 and can sometimes go negative. Pressure will fluctuate near 0 dbar. Temperature should closely match ambient temperature.



NOTE The CTD manufacturer provides plugs to keep contamination out and moisture in the cell. These plugs should be used during storage, but they must be removed before deployment.

How to Evaluate Data

See manufacturer’s documentation.

Relevant Sensors

Sensor Name	Description
sci_water_cond	units (S/m)
sci_water_temp	units (degrees C)
sci_water_pressure	units (bar)

Argos Satellite Platform Transmitter Terminal (PTT)

The Seimac X-Cat PTT is used in recovery situations to transmit last-known GPS positions, when available. The Argos service also provides periodic surface locations accurate to approximately 100 meters. See "Argos Data Format" on page H-2.

How to Configure

Factory configured

How to Test

A full system test can be performed by putting the glider outside with a clear view of the entire sky for three hours in `lab_mode`. The Argos account for the vehicle can then be checked for data. Locally, a detection device or receiver can be used to confirm transmissions.

How to Evaluate Data

A utility to decode Argos data can be found at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/windoze-bin/

Relevant Sensors

```
c_argos_on(enum) 0  #   <0 PTT is always turned off, even at surface
                    #   0 PTT powered off, but can be auto turned on at surface
                    #   >0 PTT is powered on and transmitting:
                    #   1 no diagnostic output
                    #   2 output xmitted chars to MLOG/TERM
                    #   3 output xmitted/recvd chars to MLOG/TERM
```

Catalyst

A catalyst is used to recombine hydrogen and oxygen into water to reduce the risk of explosion. The reaction is exothermic and the catalyst may become hot. This item does not need periodic replacement.

WARNING If the glider contains alkaline batteries, there is a small but finite possibility that batteries of alkaline cells will release a combustible gas mixture, especially if the batteries are exposed to water or sea water and/or shorted. This gas release generally is not evident when batteries are exposed to the atmosphere, as the gases are dispersed and diluted to a safe level. When the batteries are confined in a sealed instrument, the gases can accumulate and an explosion is possible. Teledyne Webb Research has added a catalyst inside of the glider to recombine hydrogen and oxygen into water, and the glider has been designed to relieve excessive internal pressure buildup by having the hull sections separate under internal pressure.

Teledyne Webb Research knows of no way to completely eliminate this hazard. The user is warned, and must accept and deal with this risk in order to use this instrument safely as so provided. Personnel with knowledge and training to deal with this risk should seal or operate the instrument.

Air Pump System

The air bladder in the flooded tail cone is used to provide additional buoyancy on the surface for assisting in lifting the tail from the water line for communications. It is inflated using air from the hull interior and can provide 1400 ml of reserve buoyancy. The air pump is mechanically switched off when the differential pressure (between the air bladder and the internal hull pressure) becomes 6.25 PSI. This has been factory set. When surfaced, the glider equilibrates with the tail elevated, and the boom holds the antenna clear of the water. This air is vented back into the vehicle via a latching solenoid valve for descent.

How to Configure

Factory configured. The air systems differential pressure switch is set to 3 in/Hg during manufacture.

How to Test

To activate the air pump and inflate the external air bladder in lab, type `put c_air_pump 1`. To open solenoid and deflate the external air bladder, type `put c_air_pump 0`. Full inflation should only be done with the cowling installed and will take between 3-10 minutes. The differential switch will stop the air pump when the pressure differential reaches 3 in/Hg. The glider's internal vacuum will increase by 3 in/Hg as atmosphere from the vehicle is moved into the external bladder.

How to Evaluate Data

The air bladder will be activated per the above sensors at the surface. `m_vacuum` will increase when functioning properly.

Relevant Sensors

- `c_air_pump`
- `m_air_pump`
- `m_vacuum`

Vehicle Controller

The glider's functions are controlled by a Persistor CF1 single-board miniature computer that is based on a Motorola 68338 processor. This board has low current consumption capability and supports the use of compact flash cards and miniature hard drives that are able to store large amounts of data. Controller code is written in C and architecturally is based on a layered single thread approach where each task is coded into a behavior. Behaviors can be combined in almost any order to achieve flexible, unique missions. Each device is labeled as a sensor and is logged every time that the value changes during a mission. Data is retrieved from the glider as a binary file and is post-parsed into a matrix that allows the user to easily construct graphical views of vehicle performance or scientific data. A subset of the sensors can be chosen as a science data package so as to reduce surface radio transmission time. The Persistor can have any number of pre-written missions (text files) in its memory that can be called, or new missions can be written, transmitted to the glider, and run. Missions can be altered to change the depth of inflections, send new GPS waypoints to the glider, or for many other reasons. For additional reading regarding the construction of binary data or the .dbd family of files and their contents, visit:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/specifications/dbd_file_format.txt

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/how-to-operate/controlling-contents-of-Xbd-file.txt

Lastly for a select few only…and more than you really wanted to know:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/software-howto/browsing-the-code/dbd/html/index.html

How to Configure

New G2 gliders are factory configured with production code. Production code is updated frequently and should be maintained. The pilot should review the updates and features added during each release:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/readme.txt

Instructions on updating the glider to the most recent production release can be found here:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/software-howto/updating-all-glider-software.txt

How to Test

Activate glider with green plug and from GliderDOS prompt, type `ver` to confirm release revision.

How to Evaluate Data

Carefully. The Slocum G2 glider monitors and records data from hundreds of glider sensors or variables, along with hundreds of potential science sensor variables. The user controls the vehicle by manipulating some of these sensorsand by writing missions that define hundreds of arguments within well-structured behaviors. Data from the glider often interact in complicated and sometimes non-intuitive fashions and need thoroughly investigated before the vehicle's behavior is understood.

Teledyne Webb Research provides a number of tools for viewing and processing data. Each dockserver also hosts a dataserver, which transfers raw glider binary data into a MySQL database. This database can be viewed using the Java GUI data visualizer, and any available glider sensor can then be plotted against time. Teledyne Webb Research also provides the tools to convert raw data to ASCII so that a user can format and publish data as desired. These tools can be found at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/windoze-bin/

or

ftp://ftp.glider.webbresearch.com/glider/windoze/production/linux-bin/

Relevant Sensors

See masterdata at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

Hardware Interface Board

The flight Persistor is mated to the main driver board that interfaces with all of the sensors, communications, and drive mechanisms. The board nominally runs on 15 volts DC. A section of the board is dedicated to a hardware abort mechanism. As a recovery precaution for errant events, a timer (set to 18 hours in the factory) is reset (COP_tickled) every time there is either a GPS fix or a keystroke while in GliderDOS, which would indicate that the glider is safely on the surface. If the timer elapses, however, the following circuits will be energized from the emergency circuit, forcing the glider to surface:

- Air pump
- Argos PTT
- Burn wire for the jettison weight

How to Configure

Factory configured. Pitch and buoyancy motors can be activated via controls located on the top of the board.

How to Test

See the glider's wiring diagram. To receive the latest version of the Slocum G2 Glider's wiring diagram, contact glidersupport@webbresearch.com.

The cop tickle circuit can be changed from 16 hrs to 20 minutes by moving a jumper. The cop tickle circuit can be changed from 16 hours to 2 hours by moving a jumper to JP55.

For further information regarding vehicle aborts and the function of the cop tickle, see the following support documentation:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/how-it-works/abort-sequuences.txt

Attitude Sensor

A precision navigation compass and attitude sensor monitors the bearing, pitch, and roll of the glider. These inputs are used for dead reckoning the vehicle while under water. Recalibrating the compass may be necessary at times (depending on the magnetic anomalies of the usage area). See the *Slocum Glider Maintenance Manual* for compass calibration instructions.

How to Configure

A calibration utility and its instructions can be found here:

ftp://ftp.glider.webbresearch.com/clients/compass-calibrator/files-for-windows/

How to Test

The compass is very susceptible to interference caused by ferrous materials and magnetic fields that can be present in electronics, vehicles, and buildings; for this reason, the calibration procedure should be performed as far as possible from any of these sources of interference. Type `report ++ m_heading m_pitch m_roll`, rotate the vehicle, and independently verify output.

How to Evaluate Data

The relevant sensors should reflect the orientation of the vehicle at all times.

Relevant Sensors

- `c_att_time(sec)`—0 turns on as fast as possible if off (-1)
- `m_heading`
- `m_pitch`
- `m_roll`

Global Positioning System (GPS)

The glider's GPS updates the unit's position every five seconds while the vehicle is on the surface. Output from the GPS is in the RMC NMEA 0183 format. The GPS can also be used to update the system's internal clock, if necessary.

How to Configure

Factory configured

How to Test

The glider must be outside with a clear view of the sky. If the vehicle has moved a great distance or has not been turned on for some time, a new GPS fix may require 5-10 minutes to be acquired. Type `put c_gps_on 3` to print GPS data to the screen for verification.

How to Evaluate Data

At each surfacing or when the command "where" is issued, the glider will print the surface display. Four types of locations are printed with the current time:

```
Current Time: Tue May 31 12:52:37 2011 MT: 307186
DR Location: 4116.681 N -7037.206 E measured 183.928 secs ago
Dead Reckoned or calculated position and age.
GPS TooFar: 69696969.000 N 69696969.000 E measured 1e+308 secs ago
GPS Invalid: 4116.696 N -7037.202 E measured 293.261 secs ago
```

```
Possible good location based on the sensor u_gps_reqd_valid_fixes
(default 6)

GPS Location: 4116.681 N -7037.206 E measured 185.342 secs ago
This is the last good recorded GPS location.
```

Note that each of these values has a time stamp to their age.

The vehicle calculates a maximum distance it can travel based on the sensor `u_max_water_speed`. This GPS location can be valuable if the glider is retrieved from the water and is moving with a vessel.

Relevant Sensors

There are many sensors for the GPS. The two most important are:

- `m_gps_lat`
- `m_gps_long`

To view the other sensors relevant to the GPS, refer to the masterdata at:

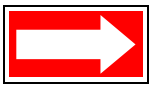
ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

Iridium Satellite Telemetry

The Iridium 9522b bidirectional satellite modem is on the lower electronics tray with a low-noise amplifier (LNA) switching board for the antenna, which is shared with the GPS. The LNA switch allows the IR modem to share its antenna with the GPS.

How to Configure

The glider's Iridium phone and SIM card are configured at the factory using the PicoDOS command `talk iridium`. Contact glidersupport@webbresearch.com for assistance depinning SIM cards or checking phone configuration.



NOTE Depinning SIM cards for the Iridium phone is normally a factory configuration and is only provided to users installing their own card or changing services. For more information on depinning, see the *Slocum G2 Maintenance Manual*.

The primary and alternate phone number to dial are entered into the glider's autoexec.mi file in the config directory. The `put` command can be used to change and test a phone number temporarily. For Rudics connections, the dockserver must also be configured for network connections as described in section 12 of the *GMC Users Guide* at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/gliderMissionControl/Documentation/gmcUserGuide.pdf

How to Test

Place the glider outside, with a clear view of the sky. Type `callback 0 0` to dial the primary number immediately. Type `callback 1 1` to call the alternate number in one minute. When satisfied, type `callback 30 0` to call back the primary number in 30 minutes. Thirty minutes is the maximum allowable callback time.

How to Evaluate Data

Connection time can vary depending on cloud cover and availability of satellites. A utility can be downloaded to confirm satellite coverage at:

http://gpredict.oz9aec.net/

Relevant Sensors

`m_iridium_signal_strength(nodim) # iridium received signal` where 0 equals no signal and 5 best

See each vehicle's autoexec.mi file for the primary and alternate numbers to be dialed by the Iridium phone.

To view the other sensors relevant to the Iridium phone, refer to the masterdata at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

Radio Frequency (RF) Modem Telemetry

When connected to the flight Persistor, the FreeWave 900 MHz radio modem is used for local high-speed communications with the glider. Due to its very high baud rate, no cable is used for communication and all in lab communications are facilitated through RF comms. See "FreeWave Configuration" on page F-1.

How to Configure

The FreeWave slave in the Slocum glider is factory configured to call all master FreeWave transmitters. The FreeWave Master shipped with each glider is configured to communicate with only one glider.

To configure a master to another glider connect the master unit to a computer with a terminal program, such as ProComm or HyperTerminal, configured for 19200 N-8-1. Power on the master, and press the button on the back of the master. From the FreeWave menu, edit the call book (option 2), and enter the seven-digit slave number from the autoexec.mi file from the desired glider into one of the 10 available slots. From the FreeWave menu, assign the call book entry to `call (C 0-9)`; do not use `call all(A)`.

How to Test

When the glider and master are powered, the red carrier detect (CD) light on the front of the master unit will turn green.

Science/Payload Computer Communications

When first powered on, the FreeWave radio modem is set to communicate with the flight Persistor. Communications can be established with the science Persistor from PicoDOS and GliderDOS.

While in PicoDOS, typing `consci` enables direct communication with the science Persistor via a hardware controlled connection. In order to resume communication with the flight Persistor, the carrier must be disconnected for three seconds, which can be accomplished by disconnecting power to the FreeWave master for about 10 seconds.

While in GliderDOS, typing `consci` enables direct communication with the science Persistor via a software-controlled connection known as the *clothesline*. In order to resume communication with the flight Persistor, enter the `quit` command.

How to Configure

Contact Glider Support.

How to Test

Type `consci` from PicoDOS and establish communication with science. The prompt should change from `(GPICO)C:\>` to `(SCI)C:\>`.

Pressure Transducer

G2 gliders are outfitted with Micron 2000 PSIA strain gage transducers, which are used for vehicle control and dead reckoning. The stainless steel transducer is ported through the aft cap and is isolated from the aluminum aft cap by a PEEK fitting.

How to Configure

Factory calibrated. Surface pressure valves can fluctuate, but the `zero_ocean_pressure` command can be used to reset the zero pressure voltage. It is recommended that the through port be examined and cleaned of contamination or blockage between deployments.

How to Test

In the field, CTD pressure can be compared against flight pressure transducer values. `m_depth` can be reported in the lab and will fluctuate.

How to Evaluate Data

Plot `m_depth` against `sci_water_pressure`.

Relevant Sensors

- `m_depth`

- m_depth_state
  - 99 ignore
  - 3 hover
  - 2 climbing
  - 1 diving
  - 0 surface
  - -1 none

Leak Detect Sensor Boards

Each glider is equipped with two leak detect sensor boards. The aft leak detect sensor is located on the bottom of the aft cap. The forward leak detect sensor is attached to the bottom of the front cap.These sensors normally report 2.5 volts. If exposed to moisture, the circuit is shorted, and any value below the masterdata default entry of 2 volts will cause an abort for leak detect.

How to Test

Type `report ++ m_leakdetect_voltage m_leakdetect_voltage_forward`. Both values should be greater than 2.3. The glider will abort missions for leak detect with any value less than 2.

How to Evaluate Data

If the leak detect sensors are less than 2.3, there is likely water in the vehicle as the result of a leak. Recovery is recommended.

Relevant Sensors

Sensor Name	Description
`f_leakdetect_threshold(volts)`	Any value of `m_leakdetect_voltage` below this threshold is considered a leak.
`m_leakdetect_voltage(volts)`	Voltage that was reported by the aft leak detect sensor. The lower the voltage, the worse the leak.
`m_leakdetect_voltage_forward(volts)`	Voltage that was reported by the forward leak detect sensor. The lower the voltage, the worse the leak.

Air Bladder

The glider's 1400 cc bladder, which is inflated by the air pump system, provides buoyancy and stability and lifts the antenna support out of the water while the glider is surfaced. Although the bladder is ruggedly constructed, care should be taken to have the aft tail cowling in place when the bladder is filling to prevent it from over-inflating. With the cowling in place, the bladder is

supported as it inflates until the pressure switch shuts off the air pump. Likewise, it is important to deflate the air bladder when removing the aft tail cowling, as it will be hard up against the cowling.

How to Test

See the air pump system test. The bladder should be inspected for damage before and after each deployment.

How to Evaluate Data

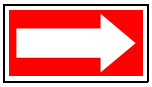
See the air pump system test.

Relevant Sensors

See the air pump system test.

Burn Wire and Jettison Weight

The glider is equipped with an emergency abort system. In the event that the vehicle is unable to surface during a mission, a battery-activated corrosive link will release the ~500-g stainless steel spring-loaded jettison weight, forcing the glider to surface. This burn process lasts for a few seconds in salt water and approximately four hours in fresh water. The jettison weight is positioned beneath the digifin tail assembly and is held in place by a 20 AWG Inconel burn wire that is mated and sealed to a single-pin Mecca connector on the aft end cap.



NOTE Activating the burn wire in air will have no effect, as it takes ions in the water to complete the return path to ground. The burn wire may be compromised if the glider is still wet but not submerged and the ejection weight is activated. See the *Slocum Glider Maintenance Manual* for more details.

How to Test

The ejection weight burn wire electronics can be tested in the lab by measuring voltage to the burn wire while the mecca connector is unplugged.

1. While in GliderDOS, type `lab_mode off`.
2. Disconnect the supply lead to the drop weight at the Mecca connector.
3. Connect the digital voltmeter between teh supply lead and the tail boom.
4. Type `put c_weight_drop 1`.
5. Verify that the voltage is at least 5 volts.
6. Type `exit pico`.
7. Verify that the voltage is 0.

8. Reconnect the drop weight supply lead.

Relevant Sensors

Sensor Name	Description
`c_weight_drop(bool)`	On-zero->drop the weight
`u_abort_min_burn_time(sec)`	During the abort sequence when a glider is having trouble getting to the surface, never drop the weight before this time.
`u_abort_max_burn_time(sec)`	During the abort sequence when a glider is having trouble getting to the surface, always drop the weight after this time.
`f_crush_depth(m)`	When the glider gets crushed, this is used to determine when to eject the weight.

Power Umbilical

An impulse cable is used to switch or supply power to the glider. When the stop plug (red) is inserted or the connector end is empty, there is no power applied to the vehicle. This is done so that power can be removed from the system without special tooling. By powering the glider via the umbilical, there is no need for an internal switch, which could generate a spark. To power the glider on either, use the provided external power cable or insert the go plug (green). Do not exceed 16 volts. The umbilical is accessible external to the glider aft tail cone.

How to Test

The glider is activated by inserting either wall power or the green plug. Removing the green plug and installing the red plug powers off the glider and should only be done after the `exit` command is issued and accepted by the glider. In an emergency, the green plug or wall power may be removed without a software command. This can result in corrupted files. A `chkdsk` should be performed on the compact flash (CF) cards and, if necessary, the cards should be reformatted.

Digital Tail Fin (digifin)

The self-calibrating tail fin assembly contains the vehicle's rudder and its three antennas:

- ARGOS 401 MHz
- FreeWave RF modem 900 MHz
- Combined GPS 1575 MHz and Iridium 1626 MHz

This tail fin is constructed of molded urethane and is rugged enough so that it can be used to handle and maniuplate the glider.

How to Configure

Factory configured

How to Test

From `lab_mode, wiggle on/wiggle off`. Range ±~25 degrees.

How to Evaluate Data

With a properly ballasted glider, positive fin movement will result in increased heading values.

Relevant Sensors

- `m_fin`
- `c_fin`
- `m_heading`
- `c_heading`
- `m_roll`

Sacrificial Anode

The outside of the aft and forward end caps are fitted with sacrificial zinc anodes to prevent corrosion of the glider's exposed aluminum and stainless steel components. The anodes should be checked for continuity to ground on a regular basis to ensure proper protection against corrosion, and they need to be replaced periodically. It is important to take note of any scratches to the glider's anodizing, as expolsed aluminum parts can be corroded. Scratches should be touched up with paint, but nail polish is effective in an emergency. The glider should be rinsed with fresh water every time it is exposed to salt water.

Several sizes of anodes are available. Contact glidersupport@webbresearch.com for assistance in determining the appropriate size for your deployment.

How to Test

1. Probe between the forward anode and top pump flange screw using a digital voltmeter on ohms setting.
2. Verify that the resistance is less than 10 ohms.
3. Probe between the aft anode and ejection weight tube using a digital voltmeter on ohms setting.
4. Verify that the resistance is less than 10 ohms.

Batteries/Coulomb Counter

Alkaline battery packs, which are nominally at 15 volts, consist of 10 diode-protected Duracell C-cells in series. As indicated below, the number of packs can be adjusted, depending on reserve buoyancy after payload considerations. Given 26 packs (260 C-cells), the total battery weight is ~18.2 kg with 7,800 kjoules of available energy.

Battery Type	Number of Battery Packs	Open Circuit Voltage (OCV)	Total Amp Hours (with a factor of safety of 10%)
Alkaline			
Pitch pack	12	N/A	73.44 Ahr
Aft pack	11	N/A	67.32 Ahr
Emergency pack	1 (located in aft pack)	N/A	6.12 Ahr
Lithium			
Pitch pack	12	11.79 V	324 Ahr with a constant load of 6 A
Aft pack	14	11.79 V	378 AHR with a constant load of 7 A
Emergency pack	1	11.79 V	6.3 AHR

For power management, typically all of the packs, except one of the aft battery packs, are tied into the main battery. In the event of a power loss, the emergency pack runs the main controller boards and performs the following functions:

- Abort timer
- Burn wire
- Argos
- Pinger (if available)

How to Configure

Factory configured. When installing new batteries, the following commands should be entered at a GliderDOS prompt. These will reset the coulomb counter to 0.

```
Put m_coulomb_amphr_total 0
```

```
Exit reset
```

From the GliderDOS prompt, type `get m_coulomb_amphr_total` and confirm that the value is close to zero and increasing when powered by battery. This value should not increase when powered by a power supply.

How to Test

When predicting mission longevity, see the electric mission spreadsheet:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/how-to-calibrate/

Battery voltage and coulomb measurements should be monitored during lab testing as well as during deployments.

The shallow glider with alkaline batteries will have 153 Ahr total.

The glider with lithium primary batteries will have 702 Ahr total (about 4 ¾ times the alkaline capacity).

Typical battery configurations:

3 in series 78 total DD in G2 lithium primary design. (pitch(36)/aft(42)) as well as 3 C cells in the emergency battery

Shipping info: UN3091 21 kg

How to Evaluate Data

See "How to Determine Mission Battery Longevity" on page I-1.

Relevant Sensors

- `m_battery`
- `m_battery_inst`
- `m_coulomb_amphr_total`
- `m_coulomb_current`
- `m_lithium_battery_relative_charge`

WARNING If the glider contains alkaline batteries, there is a small but finite possibility that batteries of alkaline cells will release a combustible gas mixture, especially if the batteries are exposed to water or sea water and/or shorted. This gas release generally is not evident when batteries are exposed to the atmosphere, as the gases are dispersed and diluted to a safe level. When the batteries are confined in a sealed instrument, the gases can accumulate and an explosion is possible. Teledyne Webb Research has added a catalyst inside of the glider to recombine hydrogen and oxygen into water, and the glider has been designed to relieve excessive internal pressure buildup by having the hull sections separate under internal pressure.

Teledyne Webb Research knows of no way to completely eliminate this hazard. The user is warned, and must accept and deal with this risk in order to use this instrument safely as so provided. Personnel with knowledge and training to deal with this risk should seal or operate the instrument.

Desiccant

The glider should be opened and sealed in a controlled, dry environment, if possible. Desiccant packs should be installed to absorb internal moisture and should be replaced for each deployment. When the glider is open for long periods, the desiccant should be stored in sealed plastic bags to prevent it from being saturated by atmospheric moisture. A fully saturated dessicant bag can increase in mass by ~40 grams and can therefore also affect ballasting.

How to Configure

Keep dry. When the glider is open desiccant will gradually gain mass from the atmosphere. Increase in mass can effect glider ballasting and cause desiccant to become ineffective.

How to Test

Desiccant weight can be monitored. The desiccant pouches provide by Teledyne Webb research weigh ~114 grams when fully dried out.

Pick Point

The payload or science bay can be outfitted with a lifting point.

Recovery System (optional)

How to Configure

See the *Slocum G2 Glider Maintenance Manual* for installation and configuration.

How to Test

1. Disconnect the supply lead to the recovery system at the Mecca connector.
2. Connect the digital voltmeter between the supply lead and the forward anode.
3. Type `put c_recovery_on 1`.
4. Verify that the voltage is at least 5 volts.
5. Type `put c_recovery_on 0`.
6. Reconnect the recovery system supply lead.

Relevant Sensors

`c_recovery_on`—A value of 1 will activate the recovery system. Unlike the jettison weight, it will require several minutes to release.

Strobe (optional)

How to Configure

Factory configured

How to Test

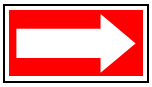
From GliderDOS or `lab_mode` or during a mission preceded by !, type `strobe on` to activate and `strobe off` to deactivate. The strobe can also be controlled autonomously during a glider surfacing behavior.

Relevant Sensors

- `c_strobe_ctrl`
- `m_strobe_ctrl`

2 Pre-mission Testing

These procedures should be followed to qualify a glider before delivery, or with new or modified software.



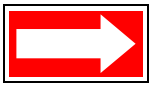
NOTE These tests need to be done outside, because the gliders need a clear view of the sky to get a GPS fix and to make Iridium communications.

On the Beach, Boat, or Bench

1. Power on the glider and enter GliderDOS by typing `ctrl-c` when instructed.
2. Type `put c_air_pump 0` to stop the air pump from running.
3. Type `callback 30` to hang up the Iridium phone.
4. Test the GPS by typing `put c_gps_on 3`. This will place GPS communications into a verbose mode. You should see the data stream change from V to A. Generally, several minutes of GPS acquisition is all that is necessary. However, if large geographical distances have been moved since the last position was acquired, it is recommended to let the GPS run for some time to build a new almanac. When satisfied with the GPS location, type `put c_gps_on 1` to return to the non-verbose mode.

WARNING NEVER deploy a glider in `lab_mode`!

5. Test the motors by typing `lab_mode on` and then `wiggle on`. Run for 3-5 minutes to check for any device errors or other abnormalities. Type `wiggle off` to stop wiggling. If there is a deep pump, you might choose to run the motor longer and report the pump's location to the screen.



NOTE Make sure the glider is set to `boot` app before deploying it in the water.

6. If no errors are found, type `lab_mode off` to return to the GliderDOS prompt. (Always ensure that the glider is not in PicoDOS or lab mode before deploying it in the water.)
7. Type `run status.mi` and confirm that all sensors are being read. The mission should end with this confirmation message, `mission completed normally`.

8. Load the glider into the boat and head out toward the first waypoint or deployment location.

In the Water

1. If possible, attach a line with flotation to the glider before putting the glider in the water. (However, if you have great confidence in the ballasting and are an experienced user, proceed without a floatation device.)
2. Once the glider is deployed, type `run status.mi` again. If ballasting has already been confirmed proceed to step 3 or 4. If unsure of ballasting, most operators will then run one or several of the following missions to quantify the quality of the ballasting. An operator may choose to remove the buoy during any number of the missions below if results are satisfactory:
   - `run ini0.mi`

     Does 1 yos, dive to 3 m
     Fixed pitch and fin
   - `run ini1.mi`

     Does 3 yos, dive to 5 m, climb to 3 m
     Heading north, pitch at ±20 degrees
   - `run ini2.mi`

     Goes to a waypoint 100 m south of dive point
     Does yos, dive to 5 m, climb to 3 m
   - `run ini3.mi`

     Goes to a waypoint 100 m north of dive point
     Does yos, dive to 30 m (alt 3.3), climb to 3 m

   If you are performing this mission on a line with flotation, ensure the line length is sufficient or modify yo depth.
3. When the glider returns to the surface, examine the data to evaluate whether it is OK to proceed.
4. If you have not removed the line from the glider, do it now.
5. From the GliderDOS prompt, type `exit reset`. This will force all of the sensor values to reinitialize. It is advisable to do an `exit reset` after removing the buoy but not necessary.
6. When the glider reboots, type `ctrl-c` to return to a GliderDOS prompt, and type `loadmission waterclr.mi` to zero any built-up water currents that are remembered long term.
7. Type `run stock.mi` to begin the stock mission or run the desired mission.

3 Deployment and Recovery

WARNING Deployment and recovery can be challenging and/or dangerous, especially in heavy seas. Plan accordingly to get the glider in and out of the water.

Deployment conditions and craft will vary. The gilder can be deployed using the pick point. With smaller vessels, the glider cart can be used on the gunwale to allow the glider to slip into the water. During handling, hold the glider by the digifin.

During recovery, the pick point or cart can be useful tools to manipulate and provide support while moving the glider aboard. A hook or lasso on a pole has also been used to manipulate the glider while in the water.

Photographs of deployment and recovery by cart can be found in the manual's link in the *Operators Handbook* at www.glider.webbresearch.com.

Gliders outfitted with a recovery system can be commanded to release the nose flotation. A vessel's overhead crane can then be used to lift the vehicle from the water line. To activate, type `put c_recovery_on 1.`

Deploying the Glider

Deployment at sea can be dangerous, and the welfare of crew and glider handlers should be considered while at the rail of a ship. From a small boat the glider cart can be used to let the glider slip easily into the water. Remove the nose ring in the original cart design or when ready release the nose ring with handle bar release on newer carts.

For larger boats, the pick point affixed to the payload bay should be used to lower and raise the glider with a crane or winch from the vessel to the water.

Glider with the Buoy and Rope Ready for the First Deployment

NOTE In the deployment sequence below, the digital tail fin (digifin) can be handled. The tail boom should be used for handling a glider not equipped with digifin.

Large Ship Deployment

A quick release system using the pick point can be fashioned from supplies found on most vessels, as illustrated in the following two images.

Recovering the Glider

NOTE A boat hook can be used to manipulate the glider in the water. Care should be taken with non-digifin gliders during deployment and recovery, because the fin can be knocked out of calibration or damaged if handled too aggressively. Handle non-digifin gliders by the tail boom or pick point only.

Lower the cart with nose ring into water, and manipulate the glider by the tail boom into position on the cart. Lift and tilt the glider onto the ship's deck.

4 Emergency Recovery

Contact glidersupport@webbreseach.com for specifics for the particular emergency, however, some or all of the following may need to be done in an emergency recovery scenario.

- Callback time can be increased significantly by changing these glider sensors:
  - `u_iridium_max_time_til_callback(sec) 1800.0 #`. This is the maximum legal value for `# c_iridium_time_til_callback`.
  - If the glider has blown the ejection weight or you feel safe, change `u_max_time` in GliderDOS from 900 to 3600. This will increase how often the glider cycles into a mission and tries to call in to save energy while it is stuck at the surface.
- Consider running special scripts to conserve energy.
- A pilot might change to a `callback 30` script on the Dockserver.
- Contact service Argos and turn on Argos ALP (all location processing).
- Determine the best-known position with the available data.

5 Glider Communications

The glider is intended to be used in conjunction with Dockserver. See the *GMC Users Guide* for information on communications to the glider while using the Dockserver.

Communicating with the RF Modem

The RF modem transmits at a frequency of 900 MHz at 1 watt nominally, but it can be configured to transmit at 0.05 watts.

A FreeWave modem should be paired with the vehicle using the Point-to-Point protocol (with repeaters, if necessary), the correct IDs in Call Book, and matching frequency keys. See "FreeWave Configuration" on page F-1.

1. Connect the FreeWave to the Dockserver or to the computer running terminal emulation.
2. Once power is applied to the FreeWave and the glider, communication should begin.

Communicating with Argos

The Argos device transmits at a frequency of ~401 MHz and power of ~1 watt.

Argos messages are nominally transmitted from the glider while it is powered up on the surface. See "Argos Data Format" on page H-2.

Communicating with Iridium

The Iridium phone transmits at a frequency of ~1600 MHz and power of ~1.1 watt.

Iridium is generally used in the absence of the FreeWave. The primary and secondary phone numbers are configured in the autoexec.mi file. While in GliderDOS, Iridium is the primary route of communication; however, while running a mission, Iridium will call only if FreeWave communication is not available. Similarly, during data transfer, the data is transferred via Iridium, only if FreeWave is not available. The option to force transfer through Iridium while FreeWave is connected is also available.

WARNING Never enter PicoDOS outside of a lab environment!

- To test Iridium while in PicoDOS, change the Dockserver to serial perspective and type `talk iridium` (refer to "PicoDOS" on page 6-2).
- To leave talk, type `control-c`. Pause for a moment; then type `control-c` again.

- To dial a number using a commercial card, type `AT 001 number to be dialed`. To dial a number using a military card type `AT 00697 number to be dialed`.
- `ata` answers an Iridium phone call.
- `ath` hangs up the phone after an Iridium call. Alternately, `control-c` can be used to hang up the phone.

Communicating with RUDICS

Refer to section 12 of the *GMC Manual*, which is located at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/gliderMissionControl/Documentation/gmcUserGuide.pdf

6 Software Architecture

The controller code is written in C and architecturally is based on a layered single-thread approach where each task is coded into a behavior, and behaviors can be combined in almost any order to achieve flexible and unique missions. These behaviors can also be constructed to deal with more complex issues, such as dead reckoning navigation, current correction, and adaptive sampling.

Each device is labeled as a sensor and is logged every time that the value changes during a mission. This data is retrieved as a binary file and is post-parsed into a matrix that allows you to replay flight dynamics or easily construct graphical views of vehicle performance or scientific data. A subset of the sensors can be chosen as a science data package so as to reduce surface radio transmission time, allowing near real-time data collection.

The glider can have in memory of the CF card any number of pre-written missions that can be called or a new mission can be created, downloaded to the glider via the RF modem or Iridium and run. Mission changes might include different inflect depths, new GPS waypoints, or turning a behavior on or off, such as current correction. Mission files are small text files. To further decrease the size of mission particulars, portions of missions can be broken out into .ma or mission acquisition files. This allows for transferring very small files to modify the most commonly adjusted mission sensors.

Table 6-1 Software Control Hierarchy

Component Name	Description
PicoDOS	Persistor's operating system.
GliderDOS	Glider's operating system.
masterdata	Defines the sensors; better known as the glider variables. There are approximately 1800 variables.
longterm.dat	Maintains the sensors/variables on a power cycle.
autoexec.mi	Defines glider specific variables.
.mi files	Mission files; define mission variables, mission behaviors and argument.
.ma files	Mission acquisition file; define mission behavior variables.

PicoDOS

PicoDOS is the operating system that ships with the Persistor CF1. Typing `help` will access the many DOS like functions and their command lines. The navigation devices may be tested in PicoDOS using the talk program. The syntax of the talk program is shown below:

Table 6-2 Syntax of the Talk Program

Command Name	Description
`talk gps`	Turns on the GPS and displays the NMEA output. This may also be used to acquire a full almanac. Leaving the GPS on for more than 15 minutes will refresh the almanac.
`talk att`	Turns on the attitude sensor and displays the pitch, roll, heading, and temperature output.
`talk iridium`	Turns on the Iridium modem and allows you to manually place or receive a call.
`talk arg`	Legacy: Turns on the ARGOS transmitter and displays the output. This command works only with the older Smartcat style PTT.

PicoDOS contains the following folder and file structure:

Volume in drive C is NONAME
Volume Serial Number is 75E1-51B6
Directory of C:
CONFIG
BIN
LOG
MISSION
SENTLOGS
STATE
AUTOEXEC.BAT

Config Folder

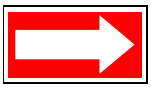
The files in the config folder are described in the table below.

Table 6-3 Files in the CONFIG Folder

File Name	Description
autoexec.mi	Configuration file for glider calibration constants and factory settings.
config.sci	Specify which sensors are sent to the glider and when.

Table 6-3 Files in the CONFIG Folder

File Name	Description
sbdlist.dat	Specify which sensors are recorded for a short binary data file.
mbdlist.dat	Specify which sensors are recorded for a medium binary data file.
simul.sim	Used to convert the glider into several versions of a simulator. This file must be deleted before an actual flight.
zmext.dat	Automatically places transferred files in the correct directory from any other directory.



NOTE The simul.sim file must be deleted before an actual flight.

Bin Folder

Pico executable programs are stored in the bin folder. All of these programs run in PicoDOS. These files are described in the table below.

Table 6-4 Files in the Bin Folder

File Name	Description
adtest.run	Displays and updates raw voltages for analog to digital devices. Used in calibration procedures.
alloff.run	Placed in autoexec.bat to start the world with all registers zeroed and turns on the RF modem for communication.
consci.run	Switches the RF modem to communicate with the payload/ science computer for direct access and code loads. A loss of carrier detection on the glider side will automatically switch back to the glider controller.
srtest.run	Allows switching selected bits on and off. Used for hardware interface board testing.
talk.run	Ports to specific components, such as Argos, attitude sensor, and GPS, for direct access and setup. `talk help` displays a list of parameters.
uarttest.run	Tests uart drivers for programming.
zr.run, zs.run	Required for the Zmodem to send and receive transfers.

NOTE A loss of carrier detection when in communication with science will automatically switch control and communications back to the glider controller.

Logs Folder

Mission derived data is stored in the logs folder. The data types are shown by file extension and described in the table below.

Table 6-5 Files Types in the Logs Folder

File Extension	Description
.dbd	Dinkum binary data—All sensors turned on for recording are stored in this type file.
.sbd	Short binary data—Records only those sensors specified in sbdlist.dat to reduce file size and thus communication time.
.mbd	Medium binary data—Records only those sensors specified in mbdlist.dat.
.mlg	Mission log—Tracks the calls for behaviors and device drives.
.log	Stores the process of opening and closing files and operations.

Missions Folder

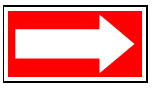
Missions are stored in this folder as .mi files. These are text files that, when run by the glider, determines the behavioral parameters.

Sentlogs Folder

This folder stores .dbd, .sbd, and .mlg files from log folders that were sent successfully.

Autoexec.bat

The autoexec.bat file is a DOS system file whose function is to automatically execute commands on system startup.



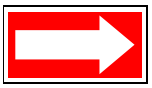
NOTE Do not confuse autoexec.bat with autoexec.mi. (See the config directory for a description of autoexec.mi.)

Typically the autoexec.bat file contains:

```
path \bin
prompt (GPico) $P$G
alloff
```

GliderDOS

The operating shell GliderDOS is a superset of PicoDOS. GliderDOS is an application that performs most of the PicoDOS functions and has knowledge of the glider. Typing `help` or `?` lists the functions available within GliderDOS. All of the variables used in GliderDOS are referred to as sensors and are defined in the Masterdata file (see "Masterdata" on page 6-6). The glider lists all of its sensors names with the `list` command. Behaviors then use these values to operate the vehicle.



NOTE The production copy of masterdata can be found here: ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

GliderDOS is an application that is loaded onto the Persistor (glider.app). When it is configured correctly, it boots up and calls autoexec.mi, a file containing all of the glider's calibration coefficients. Certain devices are set automatically to ensure the best possible surface expression:

- Ballast pump assembly full extension
- Pitch full forward
- Air pump on
- Argos on
- FreeWave on
- GPS on

It is necessary to be in PicoDOS in order to:

- Load new source code for GliderDOS.

- Work in the file structure without the device drivers being called.

WARNING The glider should never be deployed while in PicoDOS or displayed while set to `boot pico`.

For a detailed description of how to load a glider and science bay with a new release of glider production code, visit:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/software-howto/updating-all-glider-software.txt

When in GliderDOS (or as noted from PicoDOS), use the basic control commands in the table below.

Table 6-6 Basic Glider Control Commands

Command Name	Description
`control-c`	Takes control of the glider.
`exit pico`	Sends the glider to PicoDOS from GliderDOS.
`exit reset`	Return to GliderDOS (as long as Persistor is set to boot app).
`boot pico`	Commands the glider to start in PicoDOS when reset or the power is cycled. Use this when loading an application (glider.app). Never deploy a glider left to `boot pico`.
`boot app`	Commands the glider to start in GliderDOS when reset or the power is cycled. Use this after loading an application (glider.app).

Masterdata

The Masterdata file contains all of the sensor definitions and their default values as used by the application. Because Masterdata is essentially just a list of the sensors, it cannot be edited. A copy of the Masterdata file is available at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production

Prefixes

The prefixes in masterdata are described in the table below.

Table 6-7 Masterdata Prefixes

Prefix Name	Description
m_	Measured
c_	Commanded
u_	User defined before run time
f_	Set in factory. Do not change unless you know what you are doing.
x_	Never set this; typically computed at run-time.
s_	Simulated state variables
sci_	Science variable

Sensor Commands

Sensors can be changed on the GliderDOS command line. The sensor commands are described in the table below.

Table 6-8 Glider Sensor Commands

Command Name	Description
`list`	Prints all of the sensor names and values.
`get` *`sensor_name`*	Returns the present value of the sensor requested.
`put` *`sensor_name value`*	Changes the value of the sensor.
`report ?`	Prints help.
`report +` *`sensor_name`*	Reports the sensor value every time it changes.
`report ++` *`sensor_name`*	Reports the sensor value every cycle.
`report -` *`sensor_name`*	Removes sensor from reporting.
`report all`	Reports all changed sensors.
`report clearall`	Removes all sensors from reporting.
`report list`	Tells what is being reported.

Device Commands

The `use` command displays a list of all devices that are installed and in use. Installed devices are dictated by the glider's autoexec.mi file. During GliderDOS operation, an installed device is taken out of service if it receives two errors. During missions, this number is increased to 20 for most devices. Errors are primarily generated when the driven device is not moving. For this reason, the device is eventually taken out of service as a protective measure.

Table 6-9 Use Command Options

Command Option	Description
`use ?`	Prints help.
`use`	Lists all devices that are installed and in use.
`use + device_name`	Puts device(s) in service.
`use - device_name`	Takes device(s) out of service.
`use all`	Puts all installed devices in service.
`use none`	Takes all devices out of service.

From a GliderDOS prompt, the `help` command lists all commands available to the user. These commands are also listed in the table below.

Table 6-10 Device Commands

[In this table, all commands listed in all caps are executable during a mission when preceded by an exclamation point (!). The commands in lower case cannot be used during mission. Commands are not case-sensitive when you are entering them in the glider software.]

Command Name	Syntax and/or Description
ATTRIB	attrib [+ - rash] [d:][p][name]
ballast	ballast ?; for help
boot	boot [pico][pbm][app]
callback	callback <minutes til callback>
capture	capture [d:][p]fn [/Dx/B/N/E]
CD	Change directory
CHKDSK	chkdsk [d:][p][fn] [/F][/I] *
CLR_DRIFT_TABLE	Re-initializes neutral ballast vs depth table
CLRDEVERRS	Zero device errors
consci	consci [-f rf|irid]; console to science

Table 6-10 Device Commands

[In this table, all commands listed in all caps are executable during a mission when preceded by an exclamation point (!). The commands in lower case cannot be used during mission. Commands are not case-sensitive when you are entering them in the glider software.]

Command Name	Syntax and/or Description
COPY	copy source dest [/V]
CP	cp <src_path> <dest_path>; copies a file system branch
CRC	Computes CRC on memory
date	date [mdy[hms[a|p]]] /IEUMCP]
DELLOG	dellog all mlg dbd sbd
DEL	del [drv:][pth][name] [/p]
DEVICES?	Prints device driver information
DF	Prints disk space used and disk space free
digifin	digifin [dc|fc|lr|pr|rc|rr|rs|sa|sc|st|wr]
DIR	dir [d:][p][fn] [/pwblv4a:a]
DUMP	dump file[start[,end]] *
ERASE	erase [drv:][pth][name] [/P] *
exit	exit [-nofin] [poweroff|reset|pico|pbm]
GET	get <sensor name>
HARDWARE?	hardware? [-v]; hardware configuration
HEAP	Reports free memory
HELP	Prints help for commands
HIGHDENSITY	highdensity ?; for help
LAB_MODE	lab_mode [on|off]
li_ion	Need description here
LIST	Displays all sensor names
loadmission	Loads mission file
logging	logging on|off; during GliderDOS
LONGTERM_PUT	longterm_put <sensor name> <new value>
LONGTERM	longterm ?; for help

Table 6-10 Device Commands

[In this table, all commands listed in all caps are executable during a mission when preceded by an exclamation point (!). The commands in lower case cannot be used during mission. Commands are not case-sensitive when you are entering them in the glider software.]

Command Name	Syntax and/or Description
lpstop	lpstop [0 - 60 secs] ; puts the Persistor into low power mode for the specified amount of time. Will use 30 seconds as the low power time if none is specified.
LS	ls [path] ; list a file system branch
MBD	mbd ?; for help
MKDIR	mkdir [drive:][path]
MV	mv <src_path> <dest_path>; copy a file system branch
PATH	path—Shows search path * path [[d:]path[;...]] [/p] *
prompt	prompt [text] [/p] *
PRUNEDISK	Prunes expendable files to free space on disk
PURGELOGS	Deletes sent log files
PUT	put <sensor name> <value>
RENAME	rename [d:][p]oldname newname
REPORT	report ?; for help
RMDIR	rmdir [drive:][path]
RM	rm <path>; deletes a file system branch *
run	run [mission_file]; runs the mission file
SBD	sbd ?; ? for help
SEND	send [-f={rf}|{irid] [-num=<n>] [-t=<s>] [filespec ...]
sequence	sequence ?; do this for help
SETDEVLIMIT	setdevlimit devicename os w/s w/m
SETNUMWARN	setnumwarn [x]; sets max dev warnings to x
SET	set [var=[str]] [/slfe?] *
SIMUL?	Displays a print description of what is simulated
SRF_DISPLAY	srf_display ?; for help

Table 6-10 Device Commands

[In this table, all commands listed in all caps are executable during a mission when preceded by an exclamation point (!). The commands in lower case cannot be used during mission. Commands are not case-sensitive when you are entering them in the glider software.]

Command Name	Syntax and/or Description
strobe	strobe [on|off] ; flashes strobe light
sync_time	sync_time [offset]; syncs system time with gps time
tcm3	tcm3 ?; for help
time	time [hh:mm:ss [a|p]] [/m/c]
tvalve	tvalve [up|charge|down][backward] *
TYPE	type [drv:][pth][name]
USE	use ?; do this for help
VER	Displays firmware versions
WHERE	Prints latitude/longitude
whoru	whoru vehicle name:; displays vehicle name
WHY?	why? [abort#]; displays the reason for an abort
wiggle	wiggle [on|off] [fraction]; moves motor
ZERO_OCEAN_PRESSURE	Recalibrates zero ocean pressure
ZR	zmodem rec: zr ? for help
ZS	zmodem send: zs ? for help

** not often used by average user*

Science Computer

In order to communicate with the science Persistor:

- From PicoDOS, type `consci`.
- from GliderDOS, type `consci`.
- From a mission, type `ctrl-t`.

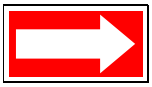
The science Persistor's folder/file structure is shown below.

bin
config
autoexec.bat

Proglets.dat is the file that contains the standard configuration values for all of the glider's science instruments. Each instrument is addressed in a section of code known as a *proglet*. Each proglet within the proglets.dat file can be modified to reflect changes to the science payload's configuration and/or instrumentation. Some of the available instruments are listed below.

Table 6-11 Selected Devices Connected to the Science Computer

Device Name	Description
ctd41cp	Sea-bird CTD (SBE-41CP) continuous profiling
bb2f	Wet labs bb2f fluorometer/backscatter sensor
bb2c	Wet labs, bb2c sensor
bb2lss	Wet labs, light scatter sensor
sam	Wet labs, Scattering Attenuation Meter
whpar	WHOI PAR Photosynthetically Active Radiation
whgpbm	WHOI Glider Bathy-Photometer
hs2	HobiLab HydroScat2 Spectral Backscattering Sensor
bam	Benthos Acoustic Modem



NOTE The science Persistor must always be set to `boot app` so that the science application runs the proglets.

Bin Folder

Pico executable programs are stored in the bin folder. All of these programs run in PicoDOS. These files are described in the table below.

Table 6-12 Files in the Bin Folder

File Name	Description
amconnct.run	Port to acoustic modem that allows communication for testing.
blast.run	Blasts characters to all ports for testing.
ctd_ctrl.run	Runs the CTD program for sampling. In autoexec.bat typed with a number ≤ 3 for fastest sampling, or > 3 for sampling at that rate in seconds. (No longer runs on gliders with software version 5.0 or greater.)
mcmd.run	Runs the acoustic modem.

Table 6-12 Files in the Bin Folder

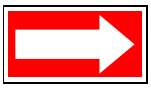
File Name	Description
mdatacol.run	Sets the acoustic modem to listen mode until the buffer is full, then tells the glider to surface.
u4talk.run	Testing of uart drivers for programming.
ctd_hs2.run	Runs either CTD, HydroScat 2, or both simultaneously. Type `help ctd_hs2` in the science computer for information on usage and options. (No longer runs on gliders with software version 5.0 or greater.)
zr.run, zs.run	Required for Zmodem send and receive transfers.

Config Folder

The files in the config folder are described in the table below.

Table 6-13 Files in the Config Folder

File Name	Description
appcmd.dat	Simulates functions in the science bay. This file must be deleted before an actual flight.
zmext.dat	Automatically places transferred files in the correct directory from any other directory.
proglets.dat	This contains configuration of the science computer sensors and defines which sensors are installed in the payload bay.



NOTE The appcmd.dat file must be deleted before an actual flight.

Autoexec.bat

This file contains the path to the executable and the PicoDOS prompt.

```
path \bin /p
prompt (sci)$p$g
```

7 Science Data Logging

In previous code releases, all science sensor data had to be passed over the serial data connection (also known as the *clothesline*) in real time to the main flight processor, where it was logged with glider sensor data. However, the clothesline imposes a limit on the number of science sensors per second that can be transferred between teh glider and science processors. Science data logging (SDL) is an architectural change that addresses the clothesline's limited throughput. In the new system using SDL, the science sensors do not have to be transmitted over the clothesline, which eliminates the bottleneck.

System Requirements

To fully implement this release, it is important to upgrade the glider processor's software to version 7.0 or greater and the science processor to version 3.0. This is absolutely necessary, because the baud rate for operating the clothesline has changed from 9600 to 4800. With science 3.0, it is still possible to fall back to the old method of operation without SDL, though this is not recommended. SDL can only be used with release 7.0 or greater, and science 3.0 or greater will be required going forward. The division between glider 6.38 and 7.0 is tied to the division between science 2.x and 3.0, and elements of old and new across this division must be consistent (either all old or all new) for the glider to operate successfully.

Performance

The science processor primarily collects and logs data, and although it can maintain a high cycle rate, it runs at reduced CPU speed by default. Of course, the actual attainable throughput depends on the installed sensor load and sensor data stream parsing overhead.

Logfile Types

Science now has a parallel logfile type for each logfile type on the glider, as shown in the table below. Each pair is formatted the same (i.e., .ebd files are formatted the same as .dbd files).

Table 7-1 Logfile Types on the Glider and Science Processors

Logfile Type on the Glider Processor	Equivalent Logfile Type on the Science Processor
.dbd	.ebd
.mbd	.nbd
.sbd	.tbd
.mlg	.nlg

Just as `m_present_time` should be present in mbdlist.dat and sbdlist.dat on the glider processor, `sci_m_present_time` should be present in nbdlist.dat and tbdlist.dat on the science processor.

The logfiles on the science processor are stored in the same directories as the glider processor:

\logs and \sentlogs, with the header cache files in \state\cache

To support SDL, some new Sc iD OS commands parallel GliderDOS commands on the glider:

`dellog`

`df`

`get` (any variable known to science, as seen by science)

`heap`

`list` (all variables known to science, with their values)

`prunedisk`

`put` (Think about whether it propagates to glider side or not.)

`send`

Configuration Files

There are also some new configuration files on science that are parallel to the glider's configuration files (see table below). Each of these pairs is also formatted the same.

Table 7-2 Configuration File Comparison on the Glider and Science Processors

Configuration File on the Glider Processor	Configuration File Equivalent on the Science Processor
\config\mbdlist.dat	\config\nbdlist.dat
\config\sbdlist.dat	\config\tbdlist.dat
\config\highdens.dat	\config\highdens.dat

Transparency

In the glider's previous code revisions, communicating with the glider could be a cumbersome and time-consuming activity that required separately connecting to the glider and science processors. In the most recent code revisions, however, these operations have become more streamlined (i.e., transparent) and user-friendly.

When the `logging on` command is issued to the glider processor, a message is passed over the clothesline that enables logging for the science processor. If science is not running at the time, a flag is set to turn on science logging when science is started. When glider logging is turned off, a clothesline message is sent to turn off science logging. If science is not running at the time, a flag is set that cancels the startlogging flag.

The log files will share the same file name root (i.e., 12345678.mlg and 12345678.nlg will represent the glider, respectively). Logfile names on science and glider are kept synchronized, but the science processor is unaware of these; the glider processor furnishes the names when it tells science to start logging each time.

Sending data files to the Dockserver is transparent. Issuing the `send` command from GliderDOS or the `s` command from the surface dialog causes log files to be sent first from science and then from the glider. The same command line is processed by each processor in turn. For example, if the command as typed is

```
send -num=3 *.sbd *.tbd
```

then science will send three .tbd files (while finding no match for *.sbd), and the glider will send three .sbd files (finding no match for *.tbd). The lack of matching files for some of the file specs is not considered an error. The `-num=3` command limits the number of files sent by each processor. In this case, the total number of files sent by two processors together is six. Sites using Data Server and Data Visualizer can view data as before. Other shoreside software tools will likely require changes to view science data. A merge tool has been developed to combine ASCII files to appear as they did before science data logging was introduced and can be found at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/windoze-bin/

Exceptions to Transparency

The `dellog` and `prunedisk` commands are local to either science or the glider. The `send` command issued directly from SciDOS is local to science (not recommended).

Control

In normal operation, only certain key science sensors are sent to the glider. For setup purposes, `c_science_on` is generally set to 2 or 3, and all sensor values are visible as they are being sent. In order to send all science sensors, as was the case before this release, a new sensor must be used (see `c_science_send_all` below).

Table 7-3 Sensors Added to the Glider

Sensor Name	Processor	Default Setting	Description
`c_science_send_all(bool)`	Glider	0	Tells the science processor whether to send all variables or just a few
`m_mission_start_time(timestamp)`	Glider	N/A	Propagates to science processor
`m_science_readiness_for_consci(enum)`	Glider	N/A	Tells if ready or, if not, why not
`sci_m_disk_free(Mbytes)`	Science	N/A	How much space is currently free on science
`sci_m_disk_usage(Mbytes)`	Science	N/A	How much space is currently used on science
`sci_m_present_secs_into_mission(sec)`	Science	N/A	Analog of m_present_secs_into_mission on science
`sci_m_free_heap(bytes)`	Science	N/A	Analog of `m_free_heap on science`
`sci_m_min_free_heap(bytes)`	Science	N/A	Analog of `m_min_free_heap on science`
`sci_m_min_spare_heap(bytes)`	Science	N/A	Analog of `m_min_spare_heap on science`
`sci_m_spare_heap(bytes)`	Science	N/A	Analog of `m_spare_heap on science`
`sci_x_disk_files_removed(nodim)`	Science	N/A	Count of files removed by last science processor's `prune` command
`sci_x_sent_data_files(nodim)`	Science	N/A	Count of files successfully transmitted by last science processor's `sent` command
`u_sci_cycle_time(secs)`	Glider	1.0	Tells the science processor how fast to run
`u_sci_dbd_sensor_list_xmit_control(enum)`	Glider	0	Always transmit the header to tell the science processor what to do
`x_science_logging_state(enum)`	Glider	N/A	Indicates the science processor's logging state

Table 7-3 Sensors Added to the Glider

Sensor Name	Processor	Default Setting	Description
u_science_send_time_limit_adjustment_ factor(nodim)	Glider	0.5	Puts an absolute limit on how long the glider can spend sending science files during a single consci session

Sending Science Data

The science `send` command is implemented as a pre-programmed consci batch command. It may take a long time to complete a single send command if a large number of files is to be transferred, so two sensors are used to limit the length of a consci session:

- `u_sci_cmd_max_consci_time` (seconds) establishes a limit for the length of a single consci session. The default value for this sensor is factory set to 3600 seconds (1 hour).
- `u_science_send_time_limit_adjustment_factor (nodim)` specifies the fraction of `u_sci_cmd_max_consci_time` that can be used for the sending of science data. This value is set conservatively to account for extra time spent enumerating files before the send, shuffling files after the send, and possible less-than-optimal communication conditions during the send. This makes it nearly (but not absolutely) certain that the send on science will not be aborted in the middle of a file by the consci timeout.

CAUTION If you need to run a huge send command for a large number of files, temporarily raise `u_sci_cmd_max_consci_time` to a potentially huge number for this purpose. (There are 3600 seconds in an hour, 86,400 seconds in a day.). Exercise caution before doing this in the water, and take care to restore the normal setting after the long send. Alternatively, send the files in small batches until all of the files have been transferred.

8 Flight Data Retrieval

The Dockserver can automate file retrieval and data storage, and display data and glider locations for easy viewing. Dockserver applications also allow data transmission via FTP. Flight data can be recorded and recovered in a number of ways. There are a number of nomenclatures described below for the way in which data is stored. Depending on the nature of the mission, the amount of data being transmitted will be customized by users to suit their particular mission needs.

This section is a guide to retrieving data from the glider during a mission surfacing, as well as when the mission has ended. Due to low transmission speed and expensive transmission rates, it is not recommended to send large files, such as .dbd, over the Iridium service. You may inexpensively and relatively quickly retrieve all *.dbd, *.mlg, and *.sbd files by using the FreeWave modem.

A separate document, *GMC User Guide*, provides instructions for using Glider Terminal, a Java application, and one of the Dockserver applications. This user guide can be found at the following location:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/gliderMissionControl/Documentation/gmcUserGuide.pdf

Table 8-1 Data File Types

Data File Extension	Description
.dbd	Dinkum binary data—All sensors (or variables) are recorded and stored in this type file. These are large files that, in most cases, would be undesirable to transmit during a mission—especially over Iridium.
.sbd	Short binary data—Records only those sensors specified in SBDLIST.DAT to reduce communication time. Customize this list of sensors to receive limited amounts of data during a mission.
.mbd	Medium binary data—A second user specified data set specified in mbdlist.dat.
.mlg	Mission log—Tracks the calls for behaviors and device drives. This is the dialog seen in communication sessions with the vehicle.
.log	Stores the process of opening and closing files and operations.

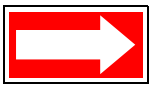
Data can be accessed and transmitted while using a terminal program while in communication with a vehicle by following the steps outlined below.

To retrieve glider data when the glider is at the surface during a mission:

1. Type `s *.dbd` or `s *.sbd`, or `s *.mlg` or `s *.mbd`.
2. The 30 most recent files of the type specified in step 1 will transmit. Remember, it is not desirable to send .dbd files over Iridium.

To retrieve glider data while the glider is not running a mission from a GliderDOS prompt, type `send *.dbd`, `send *.sbd`, `send *.mbd`, `send *.mlg`, or `send *.*`.

To retrieve science data while the glider is not running a mission from a GliderDOS prompt, type `send *.ebd`, `send *.tbd`, `send *.nbd`, `send *.nlg`, or `send *.*`



NOTE Send *.* sends all of the files of type .sbd, .mbd, .dbd .mlg, .tbd, .nbd, .ebd, and .nlg.

To send specific files in PicoDOS or GliderDOS, use `zmodem` to send the files desired by typing `zs <path><filename>`.

Vehicle Status: Glider on Surface Counting Down to Resume Mission

Table 8-2 Immediate Commands Available During Surface Mission Paused Dialog

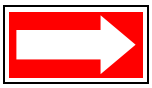
Command Name	Description
`ctrl-r`	Resumes glider mission, if so programmed.
`ctrl-c`	Aborts the mission, closes files, and remains on the surface in GliderDOS.
`ctrl-e`	Extends surface time by five minutes before resuming the mission.
`ctrl-p`	Starts the mission immediately.
`ctrl-f`	Reads the .ma files again.
`ctrl-t`	Switches to communications with the science bay Persistor.
`!`	Followed by an allowable GliderDOS command to run a subset of GliderDOS commands. This `!` command is also referred to a *bang*. (Commands listed as lower case in the help menu are not available during a mission.)

9 Dockserver Data Visualizer

Dockserver release versions 6.33 Ashumet and later include a Data Visualizer. The Data Visualizer plots all sensors from any data file transmitted to the Dockserver. For more information on how to use the Data Visualizer, see the *GMC Users Guide* at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/gliderMissionControl/Documentation/gmcUserGuide.pdf

10 Missions



NOTE The Teledyne Control Center, a software glider interface used for mission planning, has an online help system that describes how to use this product. For more information, see this help system in the Control Center.

Missions can be loaded to or drawn off the glider by using FTP and referring to Section 3 of the *GMC User Guide*. Alternatively, below is the method used to manipulate files directly while using a terminal program and the FreeWave modem.

To load a finalized mission file into the glider, you must first have it ready on your computer. Open a Glider Terminal and apply power to the glider. If the glider boots to GliderDOS and attempts to sequence a mission, exit to a GliderDOS: prompt by typing `control-c`. It is advisable to load all missions before starting field operations. However, when necessary, you can load missions from GliderDOS using the same procedure while the glider is deployed.

.mi Files

The main mission files that are run are *.mi files. These files contain all the main behaviors and sensor values for the glider. These files can be run independently or they can call mission acquisition (*.ma) files.

The autoexec.mi file is an important and unique .mi file that resides in the config directory. This file controls many of the individual settings particular to each glider. This is where calibrations for motors are stored, where the hardware expected to initialize is drawn from, where the phone number for Iridium dialing to a Dockserver is stored, and from where the glider draws its name.

.ma Files

The mission acquisition (*.ma) files called by *.mi files can contain modified behaviors, waypoint lists, surface instructions, or even sensor values. These files cannot be run independently and are always called from *.mi files. Typically, they are referenced by a number in the *.mi files, and the behavior calling.

For example, for a list of waypoints, the behavior used is goto_list, and it calls a file reference number of 07. Then the *.ma file should be called goto_l07.ma and should contain just a list of latitudes and longitudes.

Running Missions

Before running a mission, there are a few steps to follow to ensure that everything is functioning properly.

1. Connect the FreeWave radio to the antenna and a computer.
2. Run a glider terminal program on you computer.
3. Power the glider by inserting the go plug (green).
4. Check that the glider is not in Simulation Mode. (simul.sim in the glider Persistor flash card config directory and appcmd.dat in the Science Persistor flash card config directory must be deleted.)
5. Run `Status.mi` and check the following:
   - Everything is working properly.
   - There is a GPS fix.
   - The batteries are at an acceptable level.
   - No errors appear on the computer screen.

Missions can be run singly or sequenced by typing:

- `Run mission.mi`
- `Sequence mission.mi mission2.mi mission3.mi missionX.mi`
- `Sequence mission.mi(n)`, where n is number of time to run that mission.

A Abbreviations and Acronyms

ABBREVIATION OR ACRONYM	DESCRIPTION
AC or ac	Alternating Current
ASSY	Assembly
BAM	Beam Attenuation Meter
CTD	Conductivity/Temperature/Depth
COTS	Commercial Off-The-Shelf
DC or dc	Direct Current
DG	Dangerous Goods
GLMPC	Glider Mission Planning and Control
GMC	Glider Mission Control
GPS	Global Positioning System
IR	Infrared
ISO	International Organization for Standardization
ISU	Iridium Subscriber Unit
LNA	Low Noise Amplifier
MSDS	Material Safety Data Sheet
OC	Operations Center
OEM	Original Equipment Manufacturer
QCP	Quality Control Process
PPE	Personal Protective Equipment
PTT	Platform Transit Terminal (for Argos)
RHEL	Red Hat Enterprise Linux
RHN	Red Hat Network
RUDICS	Router-based Unrestricted Digital Internetworking Connectivity System
SE	Systems Engineering
SHCS	Socket Head Cap Screw
SN	Serial Number
SOP	Standard Operating Procedure
SSL	Secure Sockets Layer

ABBREVIATION OR ACRONYM	DESCRIPTION
STE	Secure Telephone Equipment
TWR	Teledyne Webb Research
U.S.	United States
USB	Universal Serial Bus
UUV	Unmanned Undersea Vehicle
VAC	Volts Alternating Current

B Code Theory and Operation

The brain of the Slocum G2 glider is a Persistor Instruments Inc. CF1 computer chip based on Motorola's MC68CK338 design. The disk space is provided via a removable compact flash card. The Slocum glider also contains a separate Persistor for logging and collecting scientific data (science computer).

Operating Systems

The operating system for the Persistor CF1 is PicoDOS (Persistor Instruments Card Or Disk Operating System). PicoDOS is smaller than, but very similar to DOS, and many DOS commands will work in PicoDOS. Uploads of new glider controller code and mission files, and retrievals of data files are all done in PicoDOS.

GliderDOS is an application that is loaded into the Persistor. This resident operating system is a superset of PicoDOS, which is used to run the glider controller code. Missions are executed from within GliderDOS. It is written in C/C++, compiled using Metrowerks CodeWarrior, and then post linked and uploaded to the glider via Motocross.

Code Design

The user commands the glider by writing *mission files* (text files with an .mi extension) and loading them onto the Persistor. The mission is then executed from within GliderDOS. Mission files are based on a layered single thread approach where tasks are coded into *behaviors* (behavior:_), which are composed of *behavior arguments* (b_arg:_).

During a mission, the glider is continually updating a large number (~1800) of variables. These variables are referred to as *sensors* (sensor:_). In the simplest terms, sensors are defined as any variable whose value is changing or set over the duration of a mission. Some examples of sensor values are GPS fixes, piston displacement, pump position, and CTD values.

All sensor definitions, behavior arguments and behaviors are defined in a file called *masterdata*, which is located at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/masterdata

Only experienced users should manipulate sensors found in masterdata. These sensors can be manipulated by the commands `put`, `longterm_put`, in the headers of missions, or by the interaction of the glider with its missions and environs. Masterdata itself is only edited by compiling a new release of code.

All lines beginning with # are comments. Any line not beginning with # will be acted upon during glider operation.

When a mission is run, GliderDOS checks the sensor values against those in masterdata. If a sensor value is set in the autoexec.mi file, the default value in the masterdata file is overwritten at glider startup. Sensor values that are set in the mission file supersede the sensor values specified in masterdata and autoexec.mi, and any sensor value can be specified in an .ma file, which supersedes all previous value entries. Users can also put sensors to any value from a GliderDOS prompt by typing the `put` command. Otherwise, the materdata value is used to determine the resulting physical outcome of the glider for that specific mission. Care should be taken when changing any of the masterdata values as they can/will adversely affect glider performance.

Control Levels

The glider controller software contains five layers, or levels, of control. These levels can be visualized in the following hierarchal structure:

- Layered control—Determines which behavior is controlling the glider's movement.
- Dynamic control—Once the layered control determines the behavior, dynamic control moves the motors, inflates and deflates the bladders, etc., to initiate the movement of behavior commanded under the layered control. In order to achieve this, dynamic control uses a specific set of device drivers to perform the movements.
- Device drivers/device scheduler
- Sensor processing—Involves the algorithm calculations to assign values (1 or 0) to the **sensors**.
- Data logging (.dbd, .sbd, .mlg, .log files)—Self explanatory, except that the values of all sensors (variables), instruments, GPS fixes, etc., are actually logged.

Control: Stacks, States, and Abort Sequences

In order to understand this controlled flow structure, it is useful to look at how various types of aborts are initiated and which layers are used to execute the aborts. Much of this information is taken from a text file called **abort-sequences** in

http://www.glider-doco.webbresearch.com/how-it-works/abort-sequuences.txt

First, the general structure of a mission file and the assignment of priority levels to a particular behavior will be explained. Then, the three types of aborts will be discussed. Following this discussion, the structure of an actual mission will be explained in further detail.

Once the glider has been instructed to execute a mission, GliderDOS reads the mission and assigns a priority to each behavior. The priority is denoted by a numerical assignment that is determined by the physical location of the behavior in the mission text file. The assigned priority list is called a *log_c_stack*, or *stack*, and takes on the following generic form. (The particular behaviors following each number will be explained in further detail later.)

```
log_c_stack():
1 – abend 2 – surface 3 – set_heading 4 – yo 5 – prepare_to_dive
6 – sensors_in
```

When one behavior is assigned priority over another and a behavior argument *(b_arg)* is satisfied in both of the behaviors (i.e., if two or more surface behaviors have been written into the mission), the behavior with the higher priority is acted upon.

After the *log_c_stack()* has been created, GliderDOS begins to scroll through the mission in order to activate various sensors and/or behaviors by changing the state of a particular behavior.

Whether a particular behavior changes from one state to another depends upon numerous sensor values.

While a mission is being executed, all behaviors are in one of the following states:

(1) Uninitiated

(2) Waiting for Activation

(3) Active

(4) Complete

Toward the end of MASTERDATA is a list of numbers and their assigned actions. This list is named *beh_args.h* and can be found in the C code. This list primarily deals with the two b_arg statements: *b_arg: start_when* and *b_arg: stop_when*, and determines when a particular behavior becomes active. Only one behavior can be active at a time.

General Control Structure

For a typical glider deployment, mission files are created by layering behaviors in a predetermined sequence to obtain the desired glider path and vertical movement. The glider mission is being executed through the layered control as displayed above. All motor settings are controlled through the behaviors.

There are three types of aborts, which the glider can trigger, to get to the surface:

- Synchronous abort
- Out of band abort
- Hardware generated abort

Each type of abort utilizes different control layers to perform the abort.

Synchronous Abort

A *synchronous abort* occurs when the glider mission does not specify completely what action to take for the current vehicle state. When all of the behaviors listed in the mission file have been called, and the glider actuators (ballast pump, pitch and heading) have not been given a commanded value, dynamic control takes over and brings the glider to the surface. At this point, the actuators are set to: ballast pump full positive buoyancy, pitch nose up, and fin straight forward. This condition continues until the glider detects that it is on the surface, or the user

interrupts with a keystroke (CTRL + C), after which control is returned to GliderDOS. While in the abort condition, all communication and location devices are also turned on (GPS, Argos, etc.), and all system log files are enabled to trace the cause of the abort. Once the abort terminates, control is returned to GliderDOS. You will see a prompt similar to:

GliderDos A # >.
GliderDos A # >. The *A* stands for abort.
GliderDos N # >. *N* indicates that the mission ended normally.
GliderDos I # >. *I* stands for initial, i.e., no mission has been run.
refers to a specific abort code. The abort codes are listed in Appendix C.

For more information about the control levels, see "Control Levels" on page B-2.

Out of Band Abort

During the second type of abort, referred to as an *out of band* abort, the glider assumes that the software is no longer reliable. For this reason, the upper two layers of software control are no longer utilized. Instead, only the device drivers/schedulers are utilized. As with the synchronous abort, the device drivers will be used to achieve positive buoyancy, the last known GPS fix is output and communication devices are turned on. The abort can only be terminated by user keystroke, even though the glider is on the surface. The following dialog is presented:

Want to reset the system? (Y or N)
Want to exit to the operating system?
Make sure you know what you are doing before answering Y
Want to exit to the operating system? (Y or N).

In general, you want to reset the system. When the system is reset, you will be returned to the GliderDOS prompt.

Hardware Generated Abort

The third type of abort is a *hardware generated* abort. The glider hardware is capable of autonomously generating an abort sequence and getting the glider to the surface by triggering the burn wire and dropping the weight. There is a watchdog circuit in the hardware with a time constant of either two hours or 16 hours, set by a jumper in the glider control board and referred to as COP_TICKLE (COP = Computer Operating Properly).

Sample Mission and Comments

The following is from an actual glider mission called *gy10v001.mi*. Black text denotes mission behaviors and behavior arguments (b_args) and is what appears in the mission text and/or masterdata. Blue text denotes comments regarding what each b_arg actually does. Red text denotes .ma files, which are discussed as they are presented.

The following behavior describes the conditions under which the glider must abort. Any text preceded by # is a comment and will not be recognized by the glider code.

behavior: abend

```
b_arg: overdepth(m)                     1000.0    # <0 disables,
                                                  # clipped to F_MAX_WORKING_DEPTH
b_arg: overdepth_sample_time(s)         10.0      # how often to check
b_arg: overtime(s)                      -1.0      # MS_ABORT_OVERTIME
                                                  # < 0 disables
b_arg: samedepth_for(s)                 120.0     # <0 disables
b_arg: samedepth_for_sample_time(s)     30.0      # how often to check
b_arg: no_cop_tickle_for(s)             7000.0    # secs, abort mission if watchdog not tickled
                                                  #this often, <0 disables
```

The following *behavior: surface* instructs the glider to come up if it hasn't had communication for a given period of time, in this case, 20 minutes.

behavior: surface

```
b_arg: args_from_file(enum)             10        # read from mafiles/surfac10.ma.
```

The corresponding .ma file is displayed below.

```
behavior_name=surface
# surfac10.ma
# climb to surface with ballast pump full out
# pitch servo'ed to 20 degrees
# Hand Written
# 08-Apr-02 tc@DinkumSoftware.com Initial
<start:b_arg>
# arguments for climb_to
    b_arg: c_use_bpump(enum)            2
    b_arg: c_bpump_value(X)             1000.0
    b_arg: c_use_pitch(enum)            3         # 1:battpos 2:setonce 3:servo
                                                  #      in          rad          rad, >0 climb
    b_arg: c_pitch_value(X)             0.3491    # 20 deg
<end:b_arg>

b_arg: start_when(enum)                 12        # BAW_NOCOMM_SECS 12, when have
                                                  # not had comms for WHEN_SECS secs
b_arg: when_secs(sec)                   1200      # 20 min, How long between surfacing, only
                                                  # if start_when==6,9, or 12
b_arg: when_wpt_dist(m)                 10        # how close to waypoint before surface, only
                                                  # if start_when==7. In this example, this
                                                  # b_arg is not active.
b_arg: end_action(enum)                 1         # 0-quit, 1 wait for ^C quit/resume, 2
                                                  # resume, 3 drift til "end_wpt_dist"
b_arg: report_all(bool)                 1         # T->report all sensors once, F->just gps
b_arg: gps_wait_time(sec)               120       # how long to wait for gps
```

```
behavior_name=surface
# surfac10.ma
# climb to surface with ballast pump full out
# pitch servo'ed to 20 degrees
# Hand Written
# 08-Apr-02 tc@DinkumSoftware.com Initial
<start:b_arg>
# arguments for climb_to
b_arg: keystroke_wait_time(sec)      300    # how long to wait for control-C
b_arg: end_wpt_dist(m)               0      # end_action == 3 ==> stop when
                                            # m_dist_to_wpt > this arg
```

The following *behavior: surface* instructs the glider to come up when the mission is done. This is determined by a lack of waypoints to direct the glider to in the x-y plane.

behavior: surface

```
b_arg: args_from_file(enum)          10     # read from mafiles/surfac10.ma
b_arg: start_when(enum)              3      # 0-immediately, 1-stack idle, 2-pitch idle,
                                            # 3-heading idle, 6-when_secs,
                                            # 7-when_wpt_dist
b_arg: end_action(enum)              0      # 0-quit, 1 wait for ^C quit/resume,
                                            # 2 resume
b_arg: gps_wait_time(s)              300    # how long to wait for gps
b_arg: keystroke_wait_time(s)        180    # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up briefly if yo finishes. This happens if a bad altimeter hit causes a dive and climb to complete in the same cycle. The glider surfaces and the yo restarts.

behavior: surface

```
b_arg: args_from_file(enum)          10     # read from mafiles/surfac10.ma
b_arg: start_when(enum)              2      # 0-immediately, 1-stack idle, 2-pitch idle,
                                            # 3-heading idle, 6-when_secs,
                                            # 7- when_wpt_dist
b_arg: end_action(enum)              1      # 0-quit, 1 wait for ^C quit/resume,
                                            # 2 resume
b_arg: gps_wait_time(s)              300    # how long to wait for gps
b_arg: keystroke_wait_time(s)        15     # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up every waypoint.

behavior: surface

```
b_arg: args_from_file(enum)          10     # read from mafiles/surfac10.ma
b_arg: start_when(enum)              8      # 0-immediately, 1-stack idle, 2-depth idle,
                                            # 6-when_secs 7-when_wpt_dist,
                                            # 8-when hit waypoint, 9-every when_secs
b_arg: when_secs(s)                  2700   # How long between surfacing, only if
                                            # start_when==6 or 9
b_arg: when_wpt_dist(m)              10     # how close to waypoint before surface,
```

behavior: surface

b_arg: end_action(enum)	1	# 0-quit, 1 wait for ^C quit/resume, # 2 resume
b_arg: report_all(bool)	0	# T->report all sensors once, F->just gps
b_arg: gps_wait_time(s)	300	# how long to wait for gps
b_arg: keystroke_wait_time(s)	300	# how long to wait for control-C

The following *behavior: surface* instructs the glider to come up when a surface request is made by the science computer.

behavior: surface

b_arg: args_from_file(enum)	10	# read from mafiles/surfac10.ma
b_arg: start_when(enum)	11	# BAW_SCI_SURFACE
b_arg: end_action(enum)	1	# 0-quit, 1 wait for ^C quit/resume, # 2 resume
b_arg: report_all(bool)	0	# T->report all sensors once, F->just gps
b_arg: gps_wait_time(s)	300	# how long to wait for gps
b_arg: keystroke_wait_time(s)	300	# how long to wait for control-C

The following *behavior: surface* instructs the glider to come up every 10 minutes. In this particular case/mission, it is commented out, and therefore not active.

behavior: surface

# b_arg: args_from_file(enum)	10	# read from mafiles/surfac10.ma
# b_arg: start_when(enum)	9	# 0-immediately, 1-stack idle, 2-depth idle, # 6-when_secs, 7-when_wpt_dist, # 8-when hit waypoint, 9-every when_secs
# b_arg: when_secs(s)	600	# How long between surfacing, only if # start_when=6 or 9
# b_arg: when_wpt_dist(m)	10	# how close to waypoint before surface
# b_arg: end_action(enum)	1	# 0-quit, 1 wait for ^C quit/resume, # 2-resume
# b_arg: report_all(bool)	0	#T->report all sensors once, F->just gps
# b_arg: gps_wait_time(s)	300	# how long to wait for gps
# b_arg: keystroke_wait_time(s)	300	# how long to wait for control-C

The following *behavior: goto_list* tells the glider where its waypoints are. For this case, we have again used the .ma convention, which allows us to write a more general mission and insert the particular waypoints/coordinates.

behavior: goto_list

b_arg: args_from_file(enum)	10	# read from mafiles/goto_l10.ma
b_arg: start_when(enum)	0	# 0-immediately, 1-stack idle 2-heading idle

The corresponding .ma file is displayed below.

```
behavior_name=goto_list
# Written by gen-goto-list-ma ver 1.0 on GMT:Tue Feb 19 18:56:54 2002
# 07-Aug-02 tc@DinkumSoftware.com Manually edited for spawars 7aug02 op in buzzards bay
# 07-Aug-02 tc@DinkumSoftware.com Changed from decimal degrees to degrees, minutes,
# decimal minutes
# goto_l10.ma
# Flies a hexagon around R4
<start:b_arg>
    b_arg: num_legs_to_run(nodim)      -1      # loop
    b_arg: start_when(enum)            0       # BAW_IMMEDIATELY
    b_arg: list_stop_when(enum)        7       # BAW_WHEN_WPT_DIST
    b_arg: initial_wpt(enum)           -2      # closest
b_arg: num_waypoints(nodim)            6
<end:b_arg>
<start:waypoints>
-7040.271 4138.861
-7040.271 4138.807
-7040.333 4138.780
-7040.395 4138.807
-7040.395 4138.861
-7040.333 4138.888
<end:waypoints>
```

The following *behavior: yo* instructs the glider to perform a *yo* (i.e., a single up and down pattern through the water column). Again, the .ma convention is used to designate the depth and altitude (together, the range) over which the yo is to be performed.

behavior: surface

```
b_arg: args_from_file(enum)        10      # read from mafiles/yo10.ma
b_arg: start_when(enum)            2       # 0-immediately, 1-stack idle 2-depth idle
b_arg: end_action(enum)            2       # 0-quit, 2 resume
```

The corresponding .ma file is displayed below.

```
behavior_name=yo
# yo-c3x5-d20-a3-p20.ma
# climb 3.5m dive 10m alt 20m pitch 20 deg
# Hand Written
# 18-Feb-02 tc@DinkumSoftware.com Initial
# 13-Mar-02 tc@DinkumSoftware.com Bug fix, end_action from quit(0) to resume(2)
# 03-aug-02 tc@DinkumSoftware.com DREA01 at ashument, went to depth only
<start:b_arg>
b_arg: start_when(enum)            2       # pitch idle (see doco below)
b_arg: num_half_cycles_to_do(nodim) -1     # Number of dive/climbs to perform
                                           # <0 is infinite, i.e. never finishes

# arguments for dive_to
    b_arg: d_target_depth(m)       5
    b_arg: d_target_altitude(m)    -1
```

```
    b_arg: d_use_pitch(enum)          3          #      1:battpos 2:setonce 3:servo
                                                 #             in        rad       rad
                                                 # <0 dive
    b_arg: d_pitch_value(X)           -0.3491    # -20 deg

# arguments for climb_to
    b_arg: c_target_depth(m)          3.5
    b_arg: c_target_altitude(m)       -1
    b_arg: c_use_pitch(enum)          3          #      1:battpos 2:setonce 3:servo
                                                 #             in        rad       rad
                                                 # >0 climb
    b_arg: c_pitch_value(X)           -0.3491    # 20 deg
    b_arg: end_action(enum)           2          # 0-quit, 2 resume
<end:b_arg>
```

The following *behavior: prepare_to_dive* instructs the glider to get ready to dive after waiting for as long as 12 minutes for a gps fix.

behavior: prepare to dive

```
b_arg: start_when(enum)           1          # 0-immediately, 1-stack idle,
                                             # 2-depth idle
b_arg: wait_time(s)               720        # 12 minutes, how long to wait for
                                             # gps
```

The following ‘behavior: sensors_in’ turns on most of the input sensors.

behavior: sensors_in

C Abort Codes

```
MS_NONE=-3
MS_COMPLETED_ABNORMALLY=-2
MS_COMPLETED_NORMALLY=-1
MS_IN_PROGRESS=0
MS_ABORT_STACK_IS_IDLE=1
MS_ABORT_HEADING_IS_IDLE=2
MS_ABORT_PITCH_IS_IDLE=3
MS_ABORT_BPUMP_IS_IDLE=4
MS_ABORT_THRENG_IS_IDLE=5
MS_ABORT_BEH_ERROR    =6
MS_ABORT_OVERDEPTH    =7
MS_ABORT_OVERTIME     =8
MS_ABORT_UNDERVOLTS   =9
MS_ABORT_SAMEDEPTH_FOR=10
MS_ABORT_USER_INTERRUPT = 11
MS_ABORT_NOINPUT      =12
MS_ABORT_INFLECTION   =13
MS_ABORT_NO_TICKLE    =14
MS_ABORT_ENG_PRESSURE =15
MS_ABORT_DEVICE_ERROR = 16
MS_ABORT_DEV_NOT_INSTALLED = 17
MS_ABORT_WPT_TOOFAR   = 18
MS_ABORT_UNREASONABLE_SETTINGS = 19
MS_ABORT_LMC_NOT_FIXED = 20
MS_ABORT_NO_HEAP       = 21
MS_ABORT_LOG_DATA_ERROR= 22
MS_ABORT_THERMAL_NOT_ENABLED=23
MS_ABORT_LEAK=24
MS_ABORT_VACUUM=25
MS_ABORT_NO_HEADING_MEASUREMENT=26
```

```
MS_ABORT_STALLED=27
MS_ABORT_DE_PUMP_IS_IDLE=28
MS_ABORT_DE_PUMP_NOT_ENABLED=29
MS_ABORT_CPU_LOADED=30
MS_ABORT_NO_ABEND_BEHAVIOR=31
MS_ABORT_LOW_REL_CHARGE=32
```

D Glider Software Website

In order to download the glider code from the repository, you must arrange for access through Teledyne Webb Research by contacting glideraccess@webbresearch.com.

A user name and password will be provided or can be requested. Teledyne Webb Research requires the organization, name, phone number and email addresses of each person requesting access. To access the glider code, log in at:

https://dmz.webbresearch.com/

The glider software website is:

http://www.glider-doco.webbresearch.com/

The glider code update procedure is located at:

http://www.glider-doco.webbresearch.com/software-howto/updating-all-glider-software.txt

E Ancillary Equipment

For more information about ancillary equipment for the gliders, see “Ancillary Glider Equipment” in Section 4 of the *Slocum G2 Glider Maintenance Manual*.

F FreeWave Configuration

Following are excerpts from the FreeWave Technologies, Inc. *Spread Spectrum Users Manual*. Refer to www.FreeWave.com for complete details.

About FreeWave Transceivers

FreeWave transceivers operate in virtually any environment where RS232 data communications occur. The transceivers functions on a nine-pin null modem cable. If the FreeWave transceivers are to be used in an application where a null modem cable is used, such as communication between two computers, then the FreeWave transceivers can be connected directly. If FreeWave transceivers are to be used to replace a straight-through RS232 cable, then a null modem cable must be placed between the transceiver and the DCE instrument to which it is connected.

Setting up the Glider Shoreside FreeWave

This mode allows a shoreside FreeWave to communicate with several slaves.

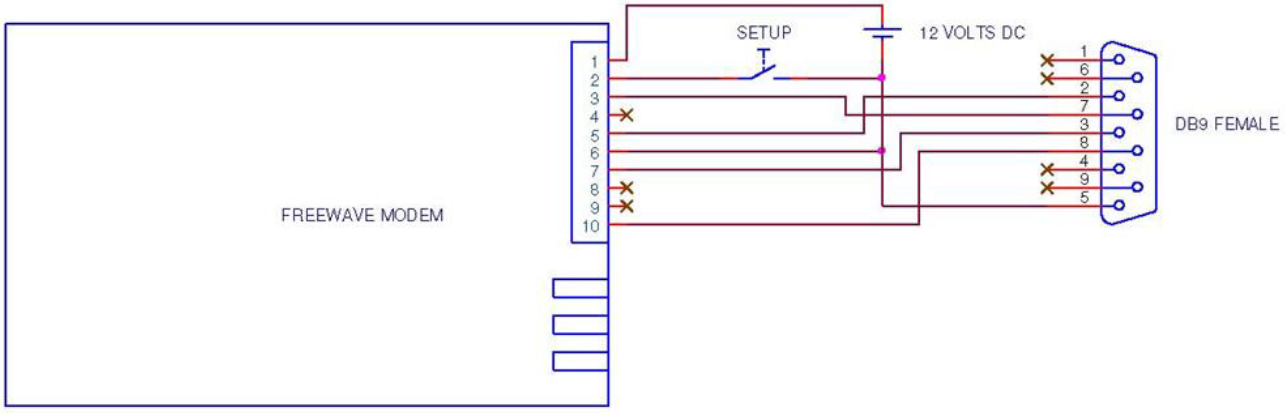
1. Connect the transceiver to the serial port of your computer through a serial cable.
2. Open a HyperTerminal session. Use the following settings to connect with HyperTerminal:
   - Connect to COMx (depending on which COM port your cable is connected to)
   - Set data rate to 19,200, data bits—8, parity-none, stop bits—1, flow control—none.
3. Press the setup button next to the serial port on the back of the radio.
   - The three lights on the board should all turn green, indicating setup mode.
   - The Main menu will appear on the screen.
4. Press 0 to access the Operation Mode menu.
   - Press 0 to set the radio as a point-to-point master.
   - Press Esc to return to the Main menu.
5. Press 1 in the Main menu to change the baud rate.
   - The baud rate in setup mode is always 19200.
   - The baud rate must be changed to match the baud rate of the device to which the radio is attached.
   - Press 1 to set the radio communication baud rate to 115,200.
   - Press Esc to return to the Main menu.
6. At the Main menu, press 3.
   - Set the frequency key.
   - Press 0 to set or change the frequency key.
   - Press 5 to set or change the frequency key to 5.
   - Press Esc to return to the Main menu.

7. At the Main menu, press 2.
   - Press 0 (0 through 9 may be used) to add the serial number of the FreeWave in the glider.
   - Press C to select the glider to which this master will communicate.
   - Press 0 to select entry 0, 1 to select entry 1, etc., or A for all radios on the list for this master to communicate with.
   - Press Esc to return to the Main menu.
   - Press Esc to exit set-up.

Setting up the Glider FreeWave Slave (internal to the glider)

This mode allows a slave to communicate with several shoreside FreeWaves.

1. Connect the transceiver to the serial port of your computer through a serial cable (see the drawing below).



2. Open a HyperTerminal session. Use the following settings to connect with HyperTerminal:
   - Connect to COMx (depending on which COM port your cable is connected to)
   - Set data rate to 19,200, data bits—8, parity-none, stop bits—1, flow control—none.
3. Press and release the Setup button. A 0 volt level on this pin will switch the radio into setup mode.
   - The three lights on the board should all turn green, indicating Setup mode.
   - The main menu will appear on the screen.
4. Press 0 to access the Operation Mode menu.
   - Press 1 to set the radio as a point-to-point slave.
   - Press Esc to return to the Main menu.
5. Press 1 in the Main menu to change the baud rate.

- The baud rate in setup mode is always 19,200.
- The baud rate must be changed to match the baud rate of the device to which the radio is attached.
- Press 1 to set the radio communication baud rate to 115,200.
- Press Esc to return to the Main menu.

6. At the Main menu, press 3.
   - Set the frequency key.
   - Press 0 to set or change the frequency key.
   - Press 5 to set or change the frequency key to 5.
   - Press Esc to return to the Main menu.
7. At the Main menu, press 2.
   - Press 0 (0 through 9 may be used) to add the serial number of the shoreside FreeWave.
   - Press C to select the glider to which this slave will communicate.
   - Press 0 to select entry 0, 1 to select entry 1, etc., or A for all radios on the list.
   - Press Esc to return to the Main menu.
   - Press Esc to exit set-up.

Choosing a Location for the Transceivers

Placement of the FreeWave transceiver is likely to have a significant impact on its performance. Height is key. In general, FreeWave units with a higher antenna placement will have a better communication link. In practice, the transceiver should be placed away from computers, telephones, answering machines, and other similar equipment. The six-foot RS232 cable included with the transceiver usually provides ample distance for placement away from other equipment. To improve the data link, FreeWave Technologies offers directional and omnidirectional antennas with cable lengths ranging from three to 200 feet. When using an external antenna, placement of that antenna is critical to a solid data link. Other antennas in close proximity are a potential source of interference; use the radio statistics to help identify potential problems. It is also possible that an adjustment as little as two feet in antenna placement can solve noise problems. In extreme cases, such as when the transceiver is located close to pager or cellphone transmission towers, the standard and cavity band pass filters that FreeWave offers can reduce the out-of-band noise.

G Iridium Service and the SIM Card

To obtain an Iridium SIM card, locate and choose a provider. Iridium charges can be a significant expense, so it is worth shopping for a good rate. There are many different plans and providers, and you can use any provider you wish.

Teledyne Webb Research uses Joubeh:
Paul Hill
Sales Manager
JouBeh Technologies Inc.
21 Thornhill Dr.
Dartmouth, Nova Scotia B3B 1R9
Canada
(902) 405-4428, x203
http://www.joubeh.com/

Stratos:
(800) 563-2255 (toll-free in North America)
(709) 748-4226 (worldwide)
support@stratosglobal.com
http://www.stratosglobal.com/

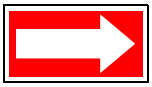
CLS America or Argos:
(301) 925-4411
userservices@clsamerica.com
http://www.clsamerica.com/

NAL Research (offers competitive rates):
(703) 392-1136
contact@nalresearch.com
http://www.nalresearch.com/Airtime.html

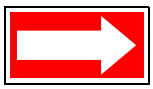
Infosat Communications:
(888) 524-3038
info@infosat.com
http://www.infosat.com/

Specify *data only* service. No equipment is needed, except for a commercial Iridium SIM card.

Billing for the service is monthly. The SIM card is required during the manufacturing process and must be activated 30 days before shipment. Teledyne Webb Research will need the actual card and the unlocking PIN so that the card can be unlocked and the PIN code deactivated permanently.



NOTE Iridium usage based on one of our users averaged 90 minutes per day per glider using some of the data reduction techniques that we provide. This cost is roughly $108/day. Expect significantly larger usage for your first deployment, because you will be monitoring, testing, and learning. After that, plan on 75-90 minutes per day per glider.



NOTE Depinning SIM cards for the Iridium phone is normally a factory configuration and is only provided to users installing their own card or changing services. For more information on depinning, see the *Slocum G2 Maintenance Manual*.

H Argos Satellite Service and ID

Glider users must provide IDs in both decimal and hexadecimal. The standard repetition rate is 90 seconds. To establish Argos service, submit a program application form to the Argos User Office. They will respond with an acknowledgement and user manual.

To contact Service Argos:
http://www.argosinc.com

North American users:
useroffice@argosinc.com

Australia and New Zealand users:
clsargos@bom.gov.au:

All other nations:
useroffice@cls.fr

Instructions for Completing the Argos Technical Information Form

Processing:	A2 (hex output)
Results format:	DS
Number of platforms:	Number of IDs
Lifetime of platform:	Number of years glider will be in use
Service required:	Location Platform
Type:	Other -Glider Message
Length:	256 bits, 20-bit IDS
Output power:	1 watt
Platform manufacturer:	Teledyne Webb Research
Platforms model:	Slocum Autonomous Glider
Transmitter manufacturer:	Seimac
Transmitters model:	X-Cat Transmission
Duty cycle:	Other -Period Surfacing/Day

It is helpful if you ask that we receive a copy of the IDs. Type of Argos application: Oceanography User Requirements: Global coverage, low transmitter power, platform compatibility, location, transmitter small size and weight, system access.

It is vital to ensure that the IDs are not in use. Please verify this.

Argos ID format change: Since 1999, CLS Argos has been preparing to change the format of the platform IDs. ~2003 Service Argos began offering two types of ID. The ID format will change from 20 bits to 28 bits. As a result, each 28-bit message will contain 31 data bytes instead of 32. This will change the format of the Argos data. For gliders with 20-bit IDs and X-cat transmitters, refer to the legacy Argos data format for decoding. For gliders with 20- or 28-bit IDs and smart cat transmitters, use rev1 format for decoding. You may need to revise their data processing software accordingly.

Teledyne Webb Research can provide data format information upon request. If your glider is missing and Argos transmissions are needed to find it, contact Service Argos and ask that ALP processing (All Location Processing) be activated.

Argos Data Format

The Argos transmitter in the glider will dictate the Argos data format. If your glider is equipped with the newer X-cat transmitter, the following data format will be transmitted at a 90-second repetition rate while the glider is on the surface. To determine the transmitter type, refer to autoexec.mi and the following sensor and sensor notes: sensor: f_argos_format(enum).

Below is the data format for gliders with X-cat ptts and 28-bit Argos IDs.

http://www.glider-doco.webbresearch.com/specifications/index.html

There are tools for automating the unpacking of the Argos data available at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/windoze-bin

prntargo.exe and prntargo_rev1.exe should be put in the c drive and run from a command prompt to work properly. Please contact glidersupport@webbresearch.com if you are having trouble using this tool while searching for your glider.

I How to Determine Mission Battery Longevity

Mission battery longevity is estimated by considering a number of factors, including the type of battery, style of pump, science sensor types and sampling strategy, surface time required for real time data transmission, and the starting battery voltage.

During an alkaline deployment, monitor the battery voltage from the glider surface dialog or the .sbd data stream by using the masterdata sensor:

```
m_battery(volts)
```

When plotting `m_battery`, note that voltage drops during heavy current usage are expected and normal (i.e., where buoyancy pump adjustments occur at depth).

The glider will begin aborting missions when the voltage drops below 10 volts per the abort behavior below:

```
b_arg: undervolts(volts) 10.0 # < 0 disables
```

When using lithium batteries, it is recommended that the following sensors be monitored in the surface dialog or included in the files:

```
sensor: m_coulomb_amphr_total(amp-hrs) 0.0   # persistent amp-hours total

sensor: f_coulomb_battery_capacity(amp-hrs) 720.0 # nominal battery capacity

sensor: m_lithium_battery_relative_charge(%) 0 # relative cumulative charge
```

The above sensors interact to cause a low battery abort behavior from the following `abend` argument:

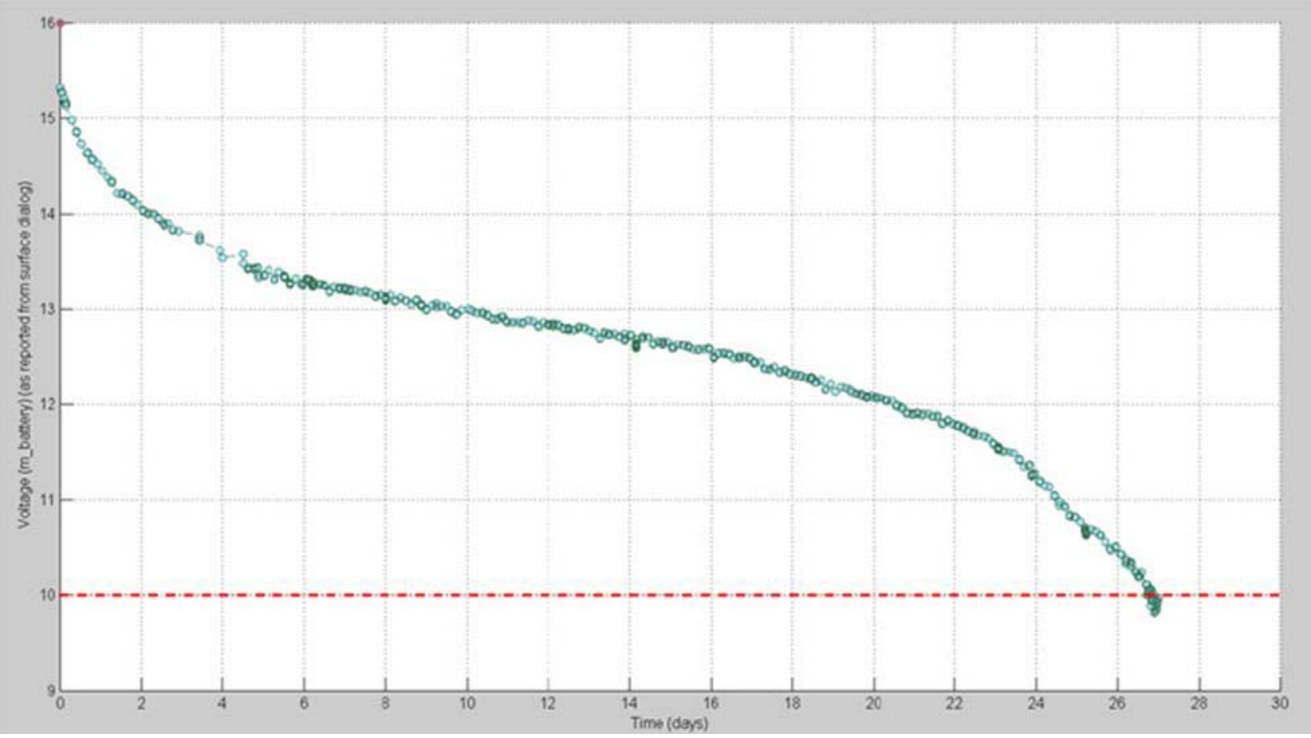
```
behavior: abend

b_arg: remaining_charge_min(%)10.0  # MS_ABORT_CHARGE_MIN out of limits
```

NOTE Each time new batteries are installed, the m_coulomb_amphr_total sensor should be reset to zero by using the `put` command, followed by an `exit reset`.

To estimate battery longevity, use the 4347 Rev. 02 Glider Endurance Tool, which is found at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/how-to-calibrate/



Above is an actual alkaline battery voltage data plot from a 27-day glider deployment. You would typically begin monitoring available recovery scenarios as the voltage approaches 12 volts. In this deployment, recovery became critical as voltage dropped below 11 volts on day 25. In this deployment, while using the factory settings, the glider began to abort all missions for under voltage beginning at day 27.

If you find yourself in a position where recovery is not possible as the voltage drops below 11.5 volts, contact glidersupport@webbresearch.com for recommendations for battery preservation, including termination of science sampling, limiting dive depth, limiting communication, and/or drifting at a shallow depth until recovery is possible.

J How to Edit a Proglets.dat File

In order to communicate with the science Persistor:

- From PicoDOS, type `consci`.
- from GliderDOS, type `consci`.
- From a mission, type `ctrl-t`.

The science Persistor's folder/file structure is shown below.

bin
config
autoexec.bat

A single science program with a standard configuration for each instrument allows the run-time selection of which instruments are actually in a given glider. A file in config directory called proglets.dat controls the wiring and configuration of the science computer. There is *proglet* for each device connected to the science computer.

1. Type `cd config`.
2. Type `dir`.
3. Confirm that proglet.dat is in the **config** directory.
4. Transfer proglets.dat from the config folder to the Dockserver by typing:

   `zs proglets.dat`
5. Preserve the original proglets.dat by changing its name. Type `rename proglets.dat proglets.org`.
6. Edit proglets.dat in the **from glider** directory.
7. Move the edited proglets.dat file to the **To glider** directory on the Dockserver.
8. Confirm that the glider is still in SciDOS. (There is a 1200-second default timeout value.)
9. Type `cd config`, if necessary.
10. Type `dockzr proglets.dat`.
11. When the transfer is complete, type `type proglet.dat`. The new file will now be displayed to screen.
12. Confirm the edits.
13. Type `quit` to exit the science computer.
14. Do one of the following:
    - If in a mission, type `!use - science_super`, and wait for science to power down. Then type `!use + science_super`.
    - If not in a mission, type `exit reset`.

Below is an excerpt from proglets.dat, in which the Aanderaa sensor is commented out to remove it from service.

Original document partial text:

```
#--------------------------------------------------------------------------

Aanderaa Oxygen Optode 3835
proglet = oxy3835
    uart    = 3       # U4Soem Pins T-2,R-3 (we only use receive)
    bit     = 34      # power control for sensor
    start_snsr = c_oxy3835_on(sec)

#--------------------------------------------------------------------------
```

The same partial text is edited to remove the Aanderaa sensor from service:

```
#--------------------------------------------------------------------------

Aanderaa Oxygen Optode 3835
proglet = oxy3835
    #   uart      = 3       # U4Soem Pins T-2,R-3 (we only use receive)
    #   bit       = 34      # power control for sensor
    #   start_snsr = c_oxy3835_on(sec)

#--------------------------------------------------------------------------
```

K Quick Reference and Checklists

To download or review the most recent *Slocum G2 Glider Operators Guide*, the quick reference, and checklists used to operate the glider, visit:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/MANUAL/

L Glider Wiring Diagram

To receive the latest version of the Slocum G2 Glider's wiring diagram, contact glidersupport@webbresearch.com.

M G2 Glider/Science Simulator User's Guide

This document is attached to this appendix.

G2 Glider/Science Simulator User's Guide

Teledyne Webb Research
82 Technology Park Drive
East Falmouth, MA 02536
U.S.A.
Tel: +1 (508) 548-2077
Fax: +1 (508) 540-1686

www.webbresearch.com

TELEDYNE
WEBB RESEARCH
A Teledyne Technologies Company

Contents

Introduction M-7
Format Notes M-7
Connections M-8
Front Panel M-10
Connecting the Simulator to the Dockserver Hardware M-11
Connecting the Simulator with a Direct Serial Cable M-11
Connecting the Simulator with a FreeWave Transceiver M-11
Connecting the Simulator with an Iridium Modem. M-12
Differences between a Simulator and a Glider M-12
Simulation M-13
Levels of Simulation M-13
simul.sim file M-13
Simulating the Iridium Modem M-14
Simulating the FreeWave Modem. M-14
Simulating Bad Input. M-14
Simulation Details M-16
Setting Initial Simulation Parameters M-17
Simulation in the Science Bay. M-17
Software Control Hierarchy M-18
Glider Computer M-18
PicoDOS M-21
GliderDOS. M-21
Masterdata M-23
Science Computer M-27
Science Data Logging M-30
System Requirements M-30
Performance. M-30
Logfile Types M-31
Transparency M-32
Control. M-32
Code Theory and Operation. M-35
Operating Systems. M-35
Code Design M-35
Control: Stacks, States, and Abort Sequences. M-36
General Control Structure M-37
Sample Mission and Comments M-39
Glider Software Website M-44

Introduction

Conceived by Douglas C. Webb and supported by Henry Stommel and others, the class of Slocum gliders is named after Joshua Slocum, the first man to single-handedly sail around the world. These gliders are innovative autonomous underwater vehicles with two primary designs: a 200-meter coastal glider and a 1000-meter glider. A third design in development is the long-range thermal glider. Each of the two existing gliders is specifically designed to maximize littoral or deep ocean capabilities with ranges from 4 to 200 meters for the 200-meter glider and 40 to 1000 meters for the 1000-meter glider. These platforms are a uniquely mobile network component capable of moving to specific locations and depths, occupying controlled spatial and temporal grids. The gliders are driven in a sawtooth vertical profile by variable buoyancy and can move horizontally and vertically.

Long-range and satellite remote sensing systems are being realized in the ocean measurement field. These systems are being used to quantify currents, sea surface height, temperature, and optical properties of the water, enabling modeling and prediction of ocean state variables in the littoral zone. A similar nested grid of subsurface observations is required to maximize the impact and ground-truth of the more extensive surface remote sensing observations.

The long-range and duration capabilities of the Slocum gliders make them ideally suited for subsurface sampling at a regional or larger scale. These gliders can be programmed to patrol for weeks or months at a time, surfacing to transmit their data to shore while downloading new instructions at regular intervals, at a substantial cost savings compared to traditional surface ships.

The small relative cost and the ability to operate multiple vehicles with minimal personnel and infrastructure enables small fleets of gliders to study and map the dynamic (temporal and spatial) features of our subsurface coastal or deep ocean waters around the clock and calendar.

Format Notes

Glider sensors and commands will be denoted in the Courier font throughout this document, as shown in the example below:

Typing Report `++ m_roll` will report measured roll (`m_roll`) every 4 seconds.

When displayed on a PC, some areas will be hyperlinked to information available on the Internet, such as:

http://www.webbresearch.com/

and protected documents by permission:

http://www.glider.webbresearch.com/

Many of the links and the code mentioned in this manual require access by prior arrangement. Please contact glidersupport@webbresearch.com to inquire about access to these protected documents.

Connections

In this section, the connections for the simulator are shown and described.

From left to right: External power 12 VDC, glider freewave connection, direct science connection, Iridium modem connection

NOTE On some models, the power and communications connectors are on the back, rather than the side, of the simulator.

Table 13-1 Simulator Connections

Connector Name	Description
External power 12 VDC	2.1 mm jack for power input from supplied 12-volt plug-in power supply
Glider FreeWave connection	DB9 F connection for communications with the simulator via the Dockserver. Connect this port to a serial port on the Dockserver configured as **freewave** or **direct** with the supplied DB9M-F cable supplied. The CD switch on the front panel should be set to **High** when this connection is used. This port can also be used for an externally connected freewave modem (configured as **slave**) for wireless communication to another freewave modem (configured as **master**) connected to the Dockserver. Set the CD switch on the front panel to **External**. Communication settings: 115200 baud, no parity, 8 data bits, 1 stop bit (115200,N,8,1).
Direct science connection	DB9F for direct connection to the science bay Persistor. This connection is parallel to the console connection between the glider main board and the science bay motherboard. Connect this port (when needed) to a serial port on the Dockserver configured as **direct**. Communication settings: 115200 baud, no parity, 8 data bits, 1 stop bit (115200,N,8,1).
Iridium modem connection	DB9F for Iridium communications. Connect this port to a serial port on the Dockserver configured as direct (This port does not support CD to the Dockserver). This port may optionally be connected to an external modem (either desktop or Iridium modem) for communications with another modem connected to a serial port on the Dockserver configured as modem. Communication settings: 4800 baud, no parity, 8 data bits, 1 stop bit (4800,N,8,1).

Front Panel

In this section, the front panel settings are shown and described.

From left to right: Carrier detect selection, glider persistor reset, science persistor reset, power on/off/science selection

Table 13-2 Front Panel Settings

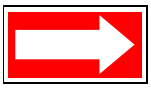
Setting Name	Description
Carrier detect selection	• **ON**—Simulates CD to the glider and to the Dockserver. • **OFF**—Forces CD low to the glider and to the Dockserver. • **EXTERNAL**—Uses the actual CD line from an external freewave modem.
Glider and science persistor resets	• **RESET**—Forces the glider or science Persistor to reset to its boot state ("boot Pico" or "boot app") • **PBM RESET**—Hold down and press "RESET**"** to force the glider or science Persistor to reset to PBM (persistor boot monitor)
Power on/off/science selection	• **OFF**—Simulator is off. • **ON**—Glider and science are on. • **SCI**—Science is on.

Connecting the Simulator to the Dockserver Hardware

The glider simulator can be connected to the Dockserver hardware by direct serial cable, FreeWave wireless data transceiver, or Iridium modem. However, to connect using a FreeWave transceiver or an Iridium modem, the simulator must have the FreeWave or modem physical device installed.

Connecting the Simulator with a Direct Serial Cable

To connect a shoebox simulator to a Dockserver machine with a direct serial cable, connect one end of a serial cable to the RS232 port labeled **Glider Comms** on the simulator and the other end to a serial port on the Dockserver machine.



NOTE For shoebox simulators directly connected by serial cable to the Dockserver machine (i.e., no Freewave or Iridium involved), a serial port configured as "direct" (ignores CD Switch setting) or "FreeWave" (follows CD switch setting) may be used. Serial port 4 on the four-port USB serial adaptor is factory configured as a "direct" device, and serial ports 2 and 3 are factory configured as "FreeWave." Refer to section 2.6 of the *GMC Users Guide* to change the factory delivered serial port configuration.

Connecting the Simulator with a FreeWave Transceiver

To connect a shoebox simulator to a Dockserver with a FreeWave transceiver:

1. Connect one end of a serial cable to the RS232 port of the **Master** FreeWave corresponding to the simulator's connected **Slave** FreeWave. Connect the other end to a serial port on the Dockserver machine.
2. Position the simulator's **Carrier Detect Source** switch to **CD External**. This position forces the simulator to read the CD line of the connected **Slave** FreeWave transceiver.

NOTE For shoebox simulators connected by Freewave transceiver to the Dockserver machine, a serial port configured as a "Freewave" device must be used. Serial ports 2 and 3 on the four-port USB serial adaptor is factory configured as a "Freewave" device. Refer to section 2.6 of the *GMC Users Guide* to change the factory delivered serial port configuration.

Connecting the Simulator with an Iridium Modem

To connect a shoebox simulator to a Dockserver with an Iridium modem, connect a U.S. Robotics model 3453B modem (configured as described in section 1.3 of the *GMC Users Guide*) to a Dockserver machine serial port using the 25-pin to 9-pin cable supplied with the modem.

NOTE For shoebox simulators connected by modem (Iridium) to the Dockserver machine, a serial port configured as a "modem" device must be used. The Dockserver's internal serial port is factory configured as a "modem" device. If the Dockserver machine has no internal serial port, then serial port 1 on the four-port USB serial adaptor is factory configured as a "modem" device. Refer to section 2.6 of the *GMC Users Guide* to change the factory delivered serial port configuration.

Differences between a Simulator and a Glider

The differences between a shoebox glider simulator and an actual glider with respect to how the Dockserver behaves are:

1. When connected as a direct device, a shoebox simulator behaves just like a pocket simulator.
2. When connected as a modem or FreeWave device, a shoebox simulator behaves like a real glider out of the water whose simul.sim file contains "on_bench."

Simulation

A powerful feature of the software is the capability to convert the glider into a simulator, operate the electronics of the glider as a simulator, or create a simulator out of a standalone Persistor. This allows testing of new code, pre-running of missions, and providing a hands-on training tool.

There are also various ways to operate the glider in a simulated environment. This also allows testing of software, simulation of missions, etc. The simulation is reasonably accurate.

The simulated physics of the glider are far from perfect but generally adequate for verifying software and missions.

Levels of Simulation

The simulation levels are described in the table below.

Table 13-3 Simulation Levels

Simulation Level	Description
no_electronics	Persistor alone—no glider board—pocket simulator
just_electronics	No hardware, motor, etc., just the electronics board and Persistor—shoebox simulator
on_bench	This is a complete glider on the bench (i.e., not in the water).

simul.sim file

The level of simulation is set when the Persistor boots by the contents of the simul.sim file in the config directory of the glider flash card. A single line of text controls the level of simulation:

- no_electronics
- just_electronics
- on_bench

If the file is missing or does not have the required line, no simulation is done.

WARNING A glider should never be deployed with a simul.sim file remaining in the config directory.

The glider simulator is shipped with a simul.sim file in the config directory that contains the text, *just_electronics null_modem*. See the next section for a description of *null_modem*.

Simulating the Iridium Modem

The Iridium satellite modem has two levels of simulation. Both are controlled by a parameter on the just_electronics line:

- just_electronics modem
- just_electronics null_modem

modem—An actual modem is attached to the Iridium connector on the electronics board. This can be an Iridium satellite modem (such as the Iridium 9522, or the 9505A handset with data adapter), although any Hayes-compatible modem should work.

null_modem—A null modem cable connects the Iridium connector on the electronics board to a Dockserver or terminal emulator. All modem responses, such as OK and CONNECTED 4800, are simulated. Set the terminal the emulator to 4800-baud/8 bit/no parity.

If neither is added to the just_electronics line, the Iridium device is taken out of service. This is the same as entering `use - Iridium` from GliderDOS.

Simulating the FreeWave Modem

Normally, in just_electronics, a direct cable is used without a FreeWave modem and the simulator always simulates the FreeWave carrier detect (CD) bit. If the following is present:

```
just_electronics [Iridium modem options see above]
Freewave_cd_switch
```

the actual CD bit from the hardware is read. This is useful in testing code that acts on the presence or absence of the FreeWave carrier.

Simulating Bad Input

The capability also exists to simulate bad input from some devices. This is useful for testing the operation of the software in the presence of bad input data and/or failed devices. The following lines in the simul.sim file determine this.

In all cases:

- **<min mt>**—Mission time to start returning bad values.
- **<max mt>**—Mission time to stop returning bad values.
- **<probability>**—0-1; the probability of returning a bad value.
- **<bad value>**—The bad value to return.

bad_device: altimeter <min mt> <max mt> <probability> <bad value>
Makes the altimeter return the specified bad value sometimes.

bad_device: attitude <min mt> <max mt> <probability> <bad value>
Makes the attitude return an error sometimes.
<bad value> The error code to return.

bad_device: gps <min mt> <max mt> <probability>

Makes the gps return an invalid fix sometimes (i.e., controls the A or V line).

bad_device: gps_error <min mt> <max mt> <probability> <ini_err_meters> <alpha>
Makes the gps return a fix an added error:

- <ini_err_meters> The err.or on first fix after power on. A random direction for the error is chosen
- <alpha> How much to reduce the error on each gps fix:
  New error = alpha * prior error.

The direction of the error remains constant.

bad_device: watchdog_oddity <min mt> <max mt> <probability>
Make the watchdog issue a SIMULATED oddity sometimes.

bad_device: pitch_stalled <min mt> <max mt> <probability>
Makes the pitch_motor appear to stall (jam) sometimes. Should generate a motor not moving warning.

bad_device: bpump_stalled <min mt> <max mt> <probability>
Makes the buoyancy appear to stall (jam) sometimes. Should generate a motor not moving warning.

bad_device: bpump_overheated <min mt> <max mt> <probability>
Makes the buoyancy overheat bit come up sometimes.

bad_device: memory_leak <min mt> <max mt> <probability>
<bytes to leak>
Consumes some heap memory and never give it back. Used for testing system behavior with inadequate heap space.
<bytes to leak> How many bytes to allow () and never free () on each simdrvr call (typically every 2 seconds)

bad_device: Iridium_no_carrier <min mt> <max mt> <probability>
Simulates a NO CARRIER condition from Iridium modem.

bad_device: leakdetect <min mt> <max mt> <probability> <bad value>
Simulates a water leak by using <bad value> as M_LEAKDETECT_VOLTAGE.

bad_device: vacuum <min mt> <max mt> <probability> <bad value>
Simulates a bad vacuum by using <bad value> as M_VACUUM.

bad_device: pressure_drift <min mt> <max mt> <probability>
<bad value>
Simulates a pressure drift by sometimes returning <bad value> as the drift. One must be simulating the device before it has any effect.

Simulation Details

The simulation is a full end-to-end simulation; when appropriate the inputs to system A/Ds and digital inputs are computed and used so that all the normal software from the lowest level is still employed.

The cycle-to-cycle timing is fairly true. Some of the subcycle timings (less than a second or so) have to be "faked" because the cpu isn't fast enough to simulate them.

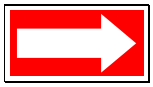
There are a number of S_xxx sensor variables. These are all computed by simulation code in the simulation driver: simdrvr.

In general, what we do at each of the levels:

on_bench

Requires fully assembled and functional glider that is not in the water. Moves all the fins and pumps, but simulates environmental inputs. Specifically:

- Compute all the S_xxx variables.
- Let the motors and other devices run normally.
- Only supply A/D input for devices, which monitor outside physical items.



NOTE For experienced users only, the on_bench open argument may be added. For more information on this argument, see the section below.

on_bench open

Used when operating a full glider on the bench ***with a dummy buoyancy pump motor*** to cause the simulation code to simulate the vacuum reading to prevent vacuum aborts.

WARNING The glider will be damaged if it is operated on_bench with the glider open and the vehicle buoyancy pump attached.

just_electronics

This requires only the glider control board. This is typically used to test science computer programs and the Iridium modem. The glider control board is only required for the uarts. This is the interface to the science computer and Iridium.

- Compute all the S_xxx variables.
- Supply simulated inputs to all A/Ds and shift registers.

- Simulate all of the uarts, *except* science and Iridium.
- Simulate modem responses if **null_modem** is specified.
- Remove Iridium from service if neither **null_modem** or **modem** are specified.

no_electronics

- Requires only a Persistor.
- Computes all the S_xxx variables.
- Supplies simulated inputs to all A/Ds, shift registers, and uarts.

Setting Initial Simulation Parameters

To begin a mission, set the origination point of the glider by:

```
Put s_ini_lat xxxx.xxxx <>
Put s_ini_lon xxxx.xxxx <>
```

A file named **loadsim.mi** is available in the **\missions** directory. This mission file allows the user to define all of the simulated values and set them by using the `loadmission` command.

Simulation in the Science Bay

The following text file must be present on the Science Flash Card:

config\appcmd.dat

containing a single line:

supersci [-nog] [-sim] [-echo_uart] [-echo_cl]

- **-nog**—No glider is attached.
- **-sim**—No sensors attached; simulate them.
- **-echo_uart**—Show send/recv lines from sensor uarts.
- **-echo_cl**—Show send/rev lines from clothesline (glider).

Software Control Hierarchy

The components of the simulator's software are described in the table below.

Table 13-4 Components of the Simulator's Software

Component Name	Description
PicoDOS	Persistor's operating system.
GliderDOS	Glider's operating system.
masterdata	Defines the sensors; better known as the glider variables. Three are approximately 1400 variables.
longterm.dat	Maintains the sensors/variables on a power cycle.
autoexec.mi	Defines glider specific variables.
.mi files	Mission files; define mission variables.
.ma files	Mission acquisition file; define mission behavior variables.

Glider Computer

PicoDOS and GliderDOS contain the following folder and file structure:

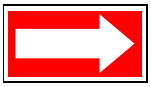
Volume in drive C is NONAME
Volume Serial Number is 75E1-51B6
Directory of C:
CONFIG
BIN
LOG
MISSION
SENTLOGS
STATE
AUTOEXEC.BAT

Config Folder

The files in the config folder are described in the table below.

Table 13-5 Files in the Config Folder

File Name	Description
autoexec.mi	Configuration file for calibration constants and factory settings.
config.sci	Specifies which sensors are sent to the glider and when.
sbdlist.dat	Specifies which sensors are recorded for a short binary data file.
mbdlist.dat	Specifies which sensors are recorded for a medium binary data file.
simul.sim	Can convert the glider into several versions of a simulator. See "Simulation" on page M-13. This file must be deleted before an actual flight.
zmext.dat	Automatically places transferred files in the correct directory from any other directory.



NOTE The simul.sim file must be deleted before an actual flight.

Bin Folder

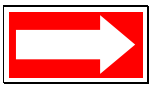
Pico executable programs are stored in the BIN folder. All of these programs run in PicoDOS. These files are described in the table below.

Table 13-6 Files in the Bin Folder

File Name	Description
adtest.run	Displays and updates raw voltages for A to D devices. Used in calibration procedures.
alloff.run	Placed in autoexec.bat to start the world with all registers zeroed and turns on the RF modem for communication.
consci.run	Switches the RF modem to the payload/science computer for direct access and code loads. A loss of carrier detection on the glider side will automatically switch back to the glider controller.
srtest.run	Allows switching selected bits on and off. Used for hardware interface board testing.

Table 13-6 Files in the Bin Folder

File Name	Description
talk.run	Ports to specific components, such as Argos, attitude sensor, and GPS, for direct access and setup. `Talk help` displays a list of parameters.
uarttest.run	Tests uart drivers for programming.
zr.run, zs.run	Required for the Zmodem to send and receive transfers.



NOTE A loss of carrier detection on the glider side will automatically switch back to the glider controller.

Logs Folder

Mission derived data is stored in the LOGS folder. The data types are shown by file extension and described in the table below.

Table 13-7 Files Types in the Logs Folder

File Extension	Description
.dbd	Dinkum Binary Data—All sensors turned on for recording are stored in this type file.
.sbd	Short Binary Data—Records only those sensors specified in sbdlist.dat to reduce communication time.
.mbd	Medium binary data—Records only those sensors specified in mbdlist.dat.
.mlg	Mission Log—Tracks the calls for behaviors and device drives.
.log	Stores the process of opening and closing files and operations.

Missions Folder

Missions are stored in this folder as .mi files. These are text files that, when run by the glider, determines the behavioral parameters.

Mafiles Folder

Mission acquisition files are stored in this folder as .ma files. These are text files that are called by missions to set behavior arguments.

Sentlogs Folder

This folder stores .dbd, .sbd, and .mlg files from log folders that were sent successfully.

Autoexec.bat

Typically the autoexec.bat contains:

```
path \bin
prompt (GPico) $P$G
alloff
```

PicoDOS

PicoDOS is the operating system that ships with the Persistor CF1. Typing `help` accesses the many DOS like functions and their command lines.

GliderDOS

The operating shell GliderDOS is a superset of PicoDOS. GliderDOS is an application that performs most of the PicoDOS functions and has knowledge of the glider. Typing `help` or `?` lists the functions available. Sensors make up all of the variables in the glider and are defined in "Masterdata" on page M-23. Behaviors then use these values to operate the vehicle. The glider lists all of its sensors names with the `list` command.

GliderDOS is an application that is loaded onto the Persistor (glider.app). When it is configured correctly, it boots up and calls an autoexec.mi file containing all of the glider's calibration coefficients. Certain devices are set automatically to ensure the best possible surface expression:

- Ballast pump assembly full extension
- Pitch full forward
- Air Pump on
- ARGOS on
- FreeWave on
- GPS on

The reasons to go to PicoDOS are to:

- Load new source code for GliderDOS.
- Work in the file structure without the device drivers being called.

WARNING The glider should never be deployed while in PicoDOS or set to boot pico.

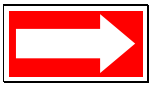
For detailed description of how to load a glider and science bay with a new release of glider production code, visit:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/software-howto/updating-all-glider-software.txt

When in GliderDOS (or as noted from PicoDOS), use the basic control commands in the table below.

Table 13-8 Basic Glider Control Commands

Command Name	Description
`control-c`	Takes control of the glider.
`exit pico`	Sends the glider to PicoDOS from GliderDOS.
`exit reset`	Return to GliderDOS (as long as Persistor is set to boot app).
`boot pico`	Commands the glider to start in PicoDOS when reset or the power is recycled. Use this when loading an application (glider.app). Never deploy a glider left to `boot pico`.
`boot app`	Commands the glider to start in GliderDOS when reset or the power is recycled. Use this after loading an application (glider.app).



NOTE The boot commands must be set in PicoDOS.

Masterdata

Masterdata contains all of the sensors or variable definitions and default values. Unlike mission files, masterdata cannot be changed as a text file. The application is inclusive of the sensor values represented by the text version of masterdata found at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production

Prefixes

The prefixes in masterdata are described in the table below.

Table 13-9 Masterdata Prefixes

Prefix Name	Description
# m_	Measured
# c_	Commanded
# u_	User defined before run time
# f_	Set in factory. Do not change unless you know what you are doing.
# x_	Never set this. Typically computed at run-time.
# s_	Simulated state variables
#sci_	Science variable

Sensor Commands

Sensors can be changed on the GliderDOS command line. The sensor commands are described in the table below.

Table 13-10 Glider Sensor Commands

Command Name	Description
`list`	Prints all of the sensor names and values.
`get` *`sensor_name`*	Returns the present value of the sensor requested.
`put` *`sensor_name value`*	Changes the value of the sensor.
`report ?`	Prints help.
`report +` *`sensor_name`*	Reports the sensor value every time it changes.
`report ++` *`sensor_name`*	Reports the sensor value every cycle.
`report -` *`sensor_name`*	Removes sensor from reporting.

Table 13-10 Glider Sensor Commands

Command Name	Description
`report all`	Reports all changed sensors.
`report clearall`	Removes all sensors from reporting.
`report list`	Tells what is being reported.

Device Commands

The `use` command displays a list of all devices that are installed and in use. During GliderDOS operation, if a device is installed and receives two errors, it is taken out of service. Currently during missions, this number is increased to 20 for most devices. Errors are primarily generated when the driven device is not moving; therefore, the device is taken out of service as a protective measure.

Table 13-11 Use Command Options

Command Option	Description
`use ?`	Prints help.
`use`	Lists all devices that are installed and in use.
`use + dev_name`	Puts device(s) in service.
`use - device_name`	Takes device(s) out of service.
`use all`	Puts all installed devices in service.
`use none`	Takes all devices out of service.

From a GliderDOS prompt, the `help` command lists all commands available to the user. These commands are also listed in the table below.

Table 13-12 Device Commands

Command Name	Syntax and/or Description
`CHKDSK`	CHKDSK [d:][p][fn] [/F][/I] *
`CLRDEVERRS`	Zero device errors
`attrib`	ATTRIB [+ - RASH] [d:][p][name]
`ballast`	BALLAST ?; for help
`boot`	boot [PICO][PBM][APP]
`callback`	callback <minutes til callback>

Table 13-12 Device Commands

Command Name	Syntax and/or Description
capture	capture [d:][p]fn [/Dx/B/N/E]
cd	Change directory
chkdsk	CHKDSK [d:][p][fn] [/F][/I] *
clrdeverrs	Zero device errors
consci	consci [-f rf|irid]; console to science
copy	copy source dest [/V]
cp	CP <src_path> <dest_path>; copies a file system branch
crc	Computes CRC on memory
date	DATE [mdy[hms[a|p]]] /IEUMCP]
dellog	DELLOG ALL MLG DBD SBD
del	DEL [drv:][pth][name] [/P]
devices?	Prints device driver information
df	Prints disk space used and disk space free
dir	DIR [d:][p][fn] [/PWBLV4A:a]
dump	DUMP file[start[,end]] *
erase	ERASE [drv:][pth][name] [/P] *
exit	exit [-nofin] [poweroff|reset|pico|pbm]
get	GET <sensor name>
hardware?	HARDWARE? [-v]; hardware configuration
heap	Reports free memory
help	Prints help for commands
highdensity	HIGHDENSITY ?; for help
lab_mode	LAB_MODE [on|off]
list	Displays all sensor names
loadmission	Loads mission file
logging	logging on|off; during GliderDOS
longterm_put	LONGTERM_PUT <sensor name> <new value>

Table 13-12 Device Commands

Command Name	Syntax and/or Description
longterm	LONGTERM ?; for help
ls	LS [path] ; list a file system branch
mbd	MBD ?; for help
mkdir	MKDIR [drive:][path]
mv	MV <src_path> <dest_path>; copy a file system branch
path	PATH Show search path * PATH [[d:]path[;...]] [/P] *
prompt	prompt [text] [/P] *
prunedisk	Prunes expendable files to free space on disk
purgelogs	Deletes sent log files
put	PUT <sensor name> <value>
rename	RENAME [d:][p]oldname newname
report	REPORT ?; for help
rmdir	RMDIR [drive:][path]
rm	RM <path>; deletes a file system branch *
run	run [mission_file]; runs the mission file
sbd	SBD ?; ? for help
send	SEND [-f={rf}|{irid] [-num=<n>] [-t=<s>] [filespec ...]
sequence	SEQUENCE ?; do this for help
setdevlimit	SETDEVLIMIT devicename os w/s w/m
path	PATH Show search path * PATH [[d:]path[;...]] [/P] *
setnumwarn	SETNUMWARN [X]; sets max dev warnings to X
set	SET [var=[str]] [/SLFE?] *
simul?	Displays a print description of what is simulated
srf_display	SRF_DISPLAY ?; for help
sync_time	sync_time [offset]; syncs system time with gps time
tcm3	TCM3 ?; for help

Table 13-12 Device Commands

Command Name	Syntax and/or Description
time	TIME [hh:mm:ss [a|p]] [/M/C]
tvalve	tvalve [up|charge|down][backward] *
type	TYPE [drv:][pth][name]
use	USE ?; do this for help
ver	Displays firmware versions
where	Prints latitude/longitude
whoru	whoru Vehicle Name:; displays vehicle name
why?	WHY? [abort#]; displays the reason for an abort
wiggle	wiggle [on|off] [fraction]; moves motor
zero_ocean_pressure	Recalibrates zero ocean pressure
zr	Zmodem Rec: zr ? for help
zs	Zmodem Send: zs ? for help

** not often used by average user*

Science Computer

To transfer to communication with the science Persistor:

Type `consci` from PicoDOS

or

Type `c` from GliderDOS

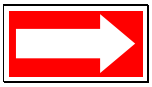
This folder and file structure is found in the science Persistor's PicoDOS:

bin
config
autoexec.bat

A single science program with a standard configuration for each instrument allows the run-time selection of which instruments are actually in a given glider. The wiring and configuration of the science Persistor is controlled by a file in the config directory called proglets.dat. There is a proglet for each device connected to the science computer. Some of these devices are listed below.

Table 13-13 Selected Devices Connected to the Science Computer

Device Name	Description
ctd41cp	Sea-bird CTD (SBE-41CP) continuous profiling
ctd41	Sea-bird CTD (SBE-41) "old" pulsed style
bb2f	Wet labs bb2f fluorometer/backscatter sensor
bb2c	Wet labs, bb2c sensor
bb2lss	Wet labs, light scatter sensor
sam	Wet labs, Scattering Attenuation Meter
whpar	WHOI PAR Photosyntheticly Active Radiation
whgpbm	WHOI Glider Bathy-Photometer
hs2	HobiLab HydroScat2 Spectral Backscattering Sensor
bam	Benthos Acoustic Modem



NOTE The science computer must always be set up to boot app so that the science application runs the proglets.

WARNING The Seabird Electronics (SBE) pumped CTD should not be run dry for more 30 seconds at a time.

The Slocum Glider data page is located at http://www.seabird.com/products/profilers.htm. Scroll down to the bottom of this page to access the link to the data page.

Bin Folder

Pico executable programs are stored in the Bin folder. All of these programs run in PicoDOS. These files are described in the table below.

Table 13-14 Files in the Bin Folder

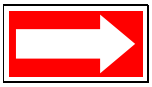
File Name	Description
amconnct.run	Port to acoustic modem that allows communication for testing.
blast.run	Blasts characters to all ports for testing.
ctd_ctrl.run	Runs the CTD program for sampling. In autoexec.bat typed with a number ≤ 3 for fastest sampling, or > 3 for sampling at that rate in seconds. (No longer runs on gliders with software version 5.0 or greater.)
mcmd.run	Runs the acoustic modem.
mdatacol.run	Acoustic modem command that sets the acoustic modem to listen mode, until the buffer is full, then tells the glider to surface.
u4talk.run	Testing of uart drivers for programming.
ctd_hs2.run	Runs either CTD, HydroScat 2 or both simultaneously. Type `help ctd_hs2` in the science computer for information on usage and options. (No longer runs on gliders with software version 5.0 or greater.)
zr.run, zs.run	Required for Zmodem send and receive transfers.

Config Folder

The files in the config folder are described in the table below.

Table 13-15 Files in the Config Folder

File Name	Description
appcmd.dat	Simulates functions in the science bay. See "Simulation" on page M-13. This file must be deleted before an actual flight.
zmext.dat	Automatically places transferred files in the correct directory from any other directory.
proglets.dat	This contains configuration of the science computer sensors and defines which sensors are installed in the payload bay.



NOTE The appcmd.dat file must be deleted before an actual flight.

Autoexec.bat

This file contains the path to the executable and the PicoDOS prompt.

```
path \bin /p
prompt (sci)$p$g
```

Science Data Logging

Science data logging (SDL) is an architectural change to address the limited throughput of the serial data connection (also known as the *clothesline*) between the glider and science processors. In previous releases, all science sensor data had to be passed over the clothesline in real time for logging commingled with glider sensor data on the glider processor, with a severe limit on the number of science sensors per second that could be transferred. In the new system using SDL, the science sensors do not have to be transmitted over the clothesline, which eliminates the bottleneck.

System Requirements

To fully implement this release, it is important to upgrade the glider processor to version 7.0 or greater and the science processor to version 3.0. This is absolutely necessary, because the baud rate for operating the clothesline has changed from 9600 to 4800. However, with science 3.0, it is still possible to fall back to the old method of operation without SDL, though this is not recommended. Only release 7.0 and higher can be used with SDL, and science 3.0 or greater will be required going forward. The division between glider 6.38 and 7.0 is tied to the division between science 2.x and 3.0, and elements of old and new across this division must be consistent (either all old or all new) for the glider to operate successfully.

Performance

Since the science processor primarily collects and logs data, although by default it is running at reduced CPU speed, it can maintain a high cycle rate. The cycle time is independent of the glider cycle time and is set to one second by default. Of course, actual throughput attainable depends on installed sensor load and sensor data stream parsing overhead.

Logfile Types

Science now has a parallel logfile type for each logfile type on the glider, as shown in the table below. Each pair is formatted the same (i.e., .ebd is formatted the same as .dbd).

Table 13-16 Logfile Types on the Glider and Science Processors

Logfile Type on the Glider Processor	Logfile Type Equivalent on the Science Processor
.dbd	.ebd
.mbd	.nbd
.sbd	.tbd
.mlg	.nlg

There are also some new configuration files on science that are parallel to the glider's configuration files (see table below). Each of these pairs is also formatted the same.

Table 13-17 Configuration File Comparison on the Glider and Science Processors

Configuration File on the Glider Processor	Configuration File Equivalent on the Science Processor
\config\mbdlist.dat	\config\nbdlist.dat
\config\sbdlist.dat	\config\tbdlist.dat
\config\highdens.dat	\config\highdens.dat

Just as m_present_time should be present in mbdlist.dat and sbdlist.dat on the glider, sci_m_present_time should be present in nbdlist.dat and tbdlist.dat on science.

The logfiles on science are stored in the same directories as the glider:

\logs and \sentlogs, with the header cache files in \state\cache

To support SDL, some new SciDOS commands parallel GliderDOS commands on the glider:

`dellog`

`df`

`get` (any variable known to science, as seen by science)

`heap`

`heap`

`list` (all variables known to science, with their values)

`prunedisk`

`put` (Think about whether it propagates to glider side or not.)

`send`

Transparency

Whenever logging is turned on in the glider, a clothesline message is sent to turn it on in science using the same file name root (i.e, the xxxxxxxx in xxxxxxxx.?bd/?lg). If science is not running at the time, a flag is set to turn on science logging when science is started.

Whenever logging is turned off on the glider, a clothesline message is sent to turn it off on science. If science is not running at the time, a flag is set that cancels the start-logging flag.

Logfile names on science and glider are kept synchronized. Science has no idea of these names itself; the glider furnishes the names when it tells science to start logging each time.

Sending data files to the Dockserver is transparent. Issuing the `send` command from GliderDOS or the `s` command from the surface dialog causes logfiles to be sent first from science and then from the glider.

The same command line is processed by each processor in turn. For example, if the command as typed is

```
send -num=3 *.sbd *.tbd
```

then science will send three .tbd files, finding no match for *.sbd, and the glider will send three .sbd files, finding no match for *.tbd. The lack of matching files for some of the filespecs is not considered an error. The
"-num=3" limits the number of files sent by each processor alone.

In this case, the total number of files sent by two processors together is six.

Sites using Data Server and Data Visualizer can view data as before. Other shoreside software tools will likely require changes to view science data. A merge tool has been developed to combine ASCII files to appear as they did before science data logging was introduced and can be found in the following location:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/windoze-bin/

Exceptions to Transparency

The `dellog` and `prunedisk` commands are local to either science or the glider. The `send` command issued directly from SciDOS is local to science (not recommended).

The `dellog` and `prunedisk` commands are local to either science or the glider. The `send` command issued directly from SciDOS is local to science (not recommended).

Control

In normal operation, only certain key science sensors are sent to the glider. For setup purposes, customarily c_science_on is set to 2 or 3, and all sensor values are visible as they are being sent.

Provision is made via a new sensor (see below, c_science_send_all) to alter behavior such that all science sensors are sent (as was always the case before this release), in order to still use this technique.

Table 13-18 Sensors Added to the Glider Simulator

Sensor Name	Processor	Default Setting*	Description
c_science_send_all(bool)	Glider	0	Tells the science processor whether to send all variables or just a few
m_mission_start_time(timestamp)	Glider	N/A	Propagates to science processor
m_science_readiness_for_consci(enum)	Glider	N/A	Tells if ready or, if not, why not
sci_m_disk_free(Mbytes)	Science	N/A	How much space is currently free on science
sci_m_disk_usage(Mbytes)	Science	N/A	How much space is currently used on science
sci_m_present_secs_into_mission(sec)	Science	N/A	Analog of m_present_secs_into_mission on science
sci_m_free_heap(bytes)	Science	N/A	Analog of m_free_heap on science
sci_m_min_free_heap(bytes)	Science	N/A	Analog of m_min_free_heap on science
sci_m_min_spare_heap(bytes)	Science	N/A	Analog of m_min_spare_heap on science
sci_m_spare_heap(bytes)	Science	N/A	Analog of m_spare_heap on science
sci_x_disk_files_removed(nodim)	Science	N/A	Count of files removed by last science processor's `prune` command
sci_x_sent_data_files(nodim)	Science	N/A	Count of files successfully transmitted by last science processor's `sent` command
u_sci_cycle_time(secs)	Glider	1.0	Tells the science processor how fast to run
u_sci_dbd_sensor_list_xmit_control(enum)	Glider	0	Always transmit the header to tell the science processor what to do
x_science_logging_state(enum)	Glider	N/A	Indicates the science processor's logging state

Table 13-18 Sensors Added to the Glider Simulator

Sensor Name	Processor	Default Setting*	Description
u_science_send_time_limit_adjustment_ factor(nodim)	Glider	0.5	Science send is implemented as a pre-programmed consci batch command. If a large number of files is to be transferred in a single send command, consideration needs to be given to u_sci_cmd_max_consci_time, which puts an absolute limit on how long a single consci session can be active (default has been changed from 1200 to 3600 seconds to allow sending a single 180K tbd file via Iridium). What science does is limit a send command to u_sci_cmd_max_consci_time times u_science_send_time_limit_adjustment_ factor seconds. This clips any -t option which may have been specified in the send command, and forms a time limit which may cut short any -num option which may have been specified in the send command. u_science_send_time_limit_adjustment_factor is set conservatively, to allow for extra time spent enumerating files before the send, shuffling files after the send, and possible less than optimal comms conditions during the send. This makes it nearly (but not absolutely) certain that the send on science will not be aborted in the middle of a file by the consci timeout.

** N/A—Not applicable*

CAUTION If you need to run a huge send command for a large number of files, temporarily raise u_sci_cmd_max_consci_time to a potentially huge number for this purpose. (There are 3600 seconds in an hour, 86,400 seconds in a day.). Exercise caution before doing this in the water, and take care to restore the normal setting after the long send. Alternatively, send the files in small batches until all of the files have been transferred.

Code Theory and Operation

The brain of the Slocum G2 glider is a Persistor Instruments Inc. CF1 computer chip based on Motorola's MC68CK338 design. The disk space is provided via a removable compact flash card. The Slocum glider also contains a separate Persistor for logging and collecting scientific data (science computer) in addition to CTD measurements, which are logged by the glider computer.

Operating Systems

The operating system for the Persistor CF1 is PicoDOS (Persistor Instruments Card Or Disk Operating System). PicoDOS is smaller than, but very similar to DOS, and many DOS commands will work in PicoDOS. Uploads of new glider controller code and mission files, and retrievals of data files are all done in PicoDOS.

GliderDOS is an application that is loaded into the Persistor. This resident operating system is a superset of PicoDOS, which is used to run the glider controller code. Missions are executed from within GliderDOS. It is written in C/C++ and compiled using Metrowerks CodeWarrior, and then post linked and uploaded to the glider via Motocross.

Code Design

The user commands the glider by writing *mission files* (text files with an .mi extension) and loading them onto the Persistor. The mission is then executed from within GliderDOS. Mission files are based on a layered single thread approach where tasks are coded into *behaviors* (behavior:_), which are composed of *behavior arguments* (b_arg:_).

During a mission, the glider is continually updating a large number (~1400) of variables.

These variables are referred to as *sensors* (sensor:_). In the simplest terms, sensors are defined as any variable whose value is changing or set over the duration of a mission. Some examples of sensor values are gps fixes, piston displacement pump position and CTD values.

WARNING The Seabird Electronics (SBE) pumped CTD should not be run dry for more 30 seconds at a time.

The Slocum Glider data page is located at http://www.seabird.com/products/profilers.htm. Scroll down to the bottom of this page to access the link to the data page.

All sensor definitions, behavior arguments and behaviors are defined in a file called *masterdata*, which is located at:

ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/code/masterdata

Only experienced users should manipulate sensors found in masterdata. These sensors can be manipulated by the commands `put`, `longterm_put`, or in the headers of missions or by the interaction of the glider with its missions and environs. Masterdata itself is only edited by compiling a new release of code.

All lines beginning with # are comments. Any line not beginning with # will be acted on during glider operation. If a sensor value is set in the autoexec.mi file (.mi), the default value in the masterdata file is overwritten; this is done at glider startup. When a mission is run, GliderDOS checks the sensor values against those in the masterdata file or presently set by the Autoexec.mi file. When a mission is run if a sensor value is set in the mission file (.mi), the default value is over written; otherwise, the masterdata value is used to determine the resulting physical outcome, of the glider for that specific mission. Any sensor value can also be overwritten in an .ma file, which supersedes all previous value entries. Users can also put sensors to any value from a GliderDOS prompt. Care should be taken when changing any of the masterdata values as they can/will adversely affect glider performance.

Control Levels

The glider controller software contains five layers, or levels, of control. These layers of control can be visualized in the following hierarchal structure:

- Layered control—Determines which behavior is controlling the glider's movement.
- Dynamic control—Once the layered control determines the behavior, dynamic control moves the motors, inflates and deflates the bladders, etc., to initiate the movement of behavior commanded under the layered control. In order to achieve this, dynamic control uses a specific set of device drivers to perform the movements.
- Device drivers/device scheduler
- Sensor processing—Involves the algorithm calculations to assign values (1 or 0) to the **sensors**.
- Data logging (.dbd, .sbd, .mlg, .log files)—Self explanatory, except that the values of all sensors (variables), instruments, GPS fixes, etc., are actually logged.

Control: Stacks, States, and Abort Sequences

In order to understand this controlled flow structure, it is useful to look at how various types of aborts are initiated and which layers are used to execute the aborts. Much of this information is taken from the abort-sequences text file in

http://www.glider-doco.webbresearch.com/how-it-works/abort-sequuences.txt

First, the general structure of a mission file and the assignment of priority levels to a particular behavior will be explained. Then, the three types of aborts will be discussed. Following this discussion, the structure of an actual mission will be explained in further detail.

Once the glider has been instructed to execute a mission, GliderDOS reads the mission and assigns a priority to each behavior. The priority is denoted by a numerical assignment and is determined by the physical location of the behavior in the mission text file. The assigned priority list is called a *log_c_stack*, or *stack*, and takes on the following generic form. (The particular behaviors will be explained in further detail later.)

log_c_stack():
1 – abend 2 – surface 3 – set_heading 4 – yo 5 – prepare_to_dive
6 – sensors_in

Where the words following each number are specific behaviors. When one behavior is assigned priority over another and a behavior argument ***b_arg*** is satisfied in both of the behaviors (i.e.: if two or more surface behaviors have been written into the mission), the behavior with the higher priority wins out.

After the *log_c_stack()* has been created, GliderDOS begins to scroll through the mission in order to activate various sensors and/or behaviors by changing the state of a particular behavior.

Whether a particular behavior changes from one state to another depends upon numerous sensor values.

While a mission is being executed, all behaviors are in one of the following states:

- (1) Uninitiated
- (2) Waiting for Activation
- (3) Active
- (4) Complete

Toward the end of MASTERDATA is a list of numbers and their *assigned* actions. This list is named *beh_args.h* and can be found in the C code. This list primarily deals with the two b_arg statements: *b_arg: start_when* and *b_arg: stop_when*, and determines when a particular behavior becomes *active*. Only one behavior can be active at a time.

General Control Structure

For a typical glider deployment, layering behaviors in a predetermined sequence to obtain the desired glider path and vertical movement creates a mission file. The glider mission is being executed through the layered control as displayed above. All motor settings are controlled through the behaviors.

There are three types of aborts, which the glider can trigger, to get to the surface:

- Synchronous abort
- Out of band abort
- Hardware generated abort

Each type of abort utilizes different control layers to perform the abort.

Synchronous Abort

A *synchronous abort* is an abort in which the behaviors selected in the mission files are no longer called. Instead, dynamic control is used to bring the glider to the surface. For more information about the control levels, see "Control Levels" on page M-36.

The dynamic control uses the device drivers to move motors so that positive buoyancy is achieved. All communication and location devices are also turned on (GPS, ARGOS, etc.). Also, all system log files are available to trace the cause of the abort. The abort terminates when the glider detects that it is on the surface or if the user interrupts with a keystroke (CTRL + C). Once the abort terminates, control is returned to GliderDOS. You will see a prompt similar to:

GliderDos A # >.
GliderDos A # >. The *A* stands for abort.
GliderDos N # >. *N* indicates that the mission ended normally.
GliderDos I # >. *I* stands for initial, i.e., no mission has been run.
refers to a specific abort code. The abort codes are listed in Appendix B of the *Slocum G2 Glider Operators Manual*.

Out of Band Abort

During the second type of abort, referred to as an *out of band* abort, the glider assumes that the software is no longer reliable. For this reason, the upper two layers of software control are no longer utilized. Instead, only the device drivers/schedulers are utilized. As with the synchronous abort, the device drivers will be used to achieve positive buoyancy, the last known GPS fix is output and communication devices are turned on. The abort can only be terminated by user keystroke, even though the glider is on the surface. The following dialog is presented:

Want to reset the system? (Y or N)
Want to exit to the operating system?
Make sure you know what you are doing before answering Y
Want to exit to the operating system? (Y or N).

In general, you want to reset the system. When the system is reset, you will be returned to the GliderDOS prompt.

Hardware Generated Abort

The third type of abort is a *hardware generated* abort. The glider hardware is capable of autonomously generating an abort sequence and getting the glider to the surface by triggering the burn wire and dropping the weight. There is a watchdog circuit in the hardware with a time constant of either two hours or 16 hours, set by a jumper in the glider control board and referred to as COP_TICKLE (COP = Computer Operating Properly).

Sample Mission and Comments

The following is from an actual glider mission called *gy10v001.mi*. Black text denotes mission behaviors and behavior arguments (b_arg's) and is what appears in the mission text and/or masterdata. Blue text denotes comments regarding what each b_arg actually does. Red text denotes .ma files, which are discussed as they are presented.

The following behavior describes the conditions under which the glider must abort. Any text preceded by # is a comment and will not be recognized by the glider code.

behavior: abend

```
b_arg: overdepth(m)                     1000.0    # <0 disables,
                                                  # clipped to F_MAX_WORKING_DEPTH
b_arg: overdepth_sample_time(s)         10.0      # how often to check
b_arg: overtime(s)                      -1.0      # MS_ABORT_OVERTIME
                                                  # < 0 disables
b_arg: samedepth_for(s)                 120.0     # <0 disables
b_arg: samedepth_for_sample_time(s)     30.0      # how often to check
b_arg: no_cop_tickle_for(s)             7000.0    # secs, abort mission if watchdog not tickled
                                                  #this often, <0 disables
```

The following 'behavior: surface' instructs the glider to come up if it hasn't had communication for a given period of time, in this case, 20 minutes.

behavior: surface

```
b_arg: args_from_file(enum)             10        # read from mafiles/surfac10.ma.
```

The corresponding .ma file is displayed below.

```
behavior_name=surface
# surfac10.ma
# climb to surface with ballast pump full out
# pitch servo'ed to 20 degrees
# Hand Written
# 08-Apr-02 tc@DinkumSoftware.com Initial
<start:b_arg>
# arguments for climb_to
    b_arg: c_use_bpump(enum)            2
    b_arg: c_bpump_value(X)             1000.0
    b_arg: c_use_pitch(enum)            3         # 1:battpos 2:setonce 3:servo
                                                  #     in          rad         rad, >0 climb
    b_arg: c_pitch_value(X)             0.3491    # 20 deg
<end:b_arg>


b_arg: start_when(enum)                 12        # BAW_NOCOMM_SECS 12, when have
                                                  # not had comms for WHEN_SECS secs
b_arg: when_secs(sec)                   1200      # 20 min, How long between surfacing, only
                                                  # if start_when==6,9, or 12
```

```
behavior_name=surface
# surfac10.ma
# climb to surface with ballast pump full out
# pitch servo'ed to 20 degrees
# Hand Written
# 08-Apr-02 tc@DinkumSoftware.com Initial
<start:b_arg>
# arguments for climb_to
b_arg: when_wpt_dist(m)              10      # how close to waypoint before surface, only
                                             # if start_when==7. In this example, this
                                             # b_arg is not active.
b_arg: end_action(enum)              1       # 0-quit, 1 wait for ^C quit/resume, 2
                                             # resume, 3 drift til "end_wpt_dist"
b_arg: report_all(bool)              1       # T->report all sensors once, F->just gps
b_arg: gps_wait_time(sec)            120     # how long to wait for gps
b_arg: keystroke_wait_time(sec)      300     # how long to wait for control-C
b_arg: end_wpt_dist(m)               0       # end_action == 3 ==> stop when
                                             # m_dist_to_wpt > this arg
```

The following *behavior: surface* instructs the glider to come up when the mission is done. This is determined by a lack of waypoints to direct the glider to in the x-y plane.

behavior: surface

```
b_arg: args_from_file(enum)          10      # read from mafiles/surfac10.ma
b_arg: start_when(enum)              3       # 0-immediately, 1-stack idle, 2-pitch idle,
                                             # 3-heading idle, 6-when_secs,
                                             # 7-when_wpt_dist
b_arg: end_action(enum)              0       # 0-quit, 1 wait for ^C quit/resume,
                                             # 2 resume
b_arg: gps_wait_time(s)              300     # how long to wait for gps
b_arg: keystroke_wait_time(s)        180     # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up briefly if yo finishes. This happens if a bad altimeter hit causes a dive and climb to complete in the same cycle. The glider surfaces and the yo restarts.

behavior: surface

```
b_arg: args_from_file(enum)          10      # read from mafiles/surfac10.ma
b_arg: start_when(enum)              2       # 0-immediately, 1-stack idle, 2-pitch idle,
                                             # 3-heading idle, 6-when_secs,
                                             # 7- when_wpt_dist
b_arg: end_action(enum)              1       # 0-quit, 1 wait for ^C quit/resume,
                                             # 2 resume
b_arg: gps_wait_time(s)              300     # how long to wait for gps
b_arg: keystroke_wait_time(s)        15      # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up every waypoint.

behavior: surface

```
b_arg: args_from_file(enum)        10    # read from mafiles/surfac10.ma
b_arg: start_when(enum)            8     # 0-immediately, 1-stack idle, 2-depth idle,
                                         # 6-when_secs 7-when_wpt_dist,
                                         # 8-when hit waypoint, 9-every when_secs
b_arg: when_secs(s)                2700  # How long between surfacing, only if
                                         # start_when==6 or 9
b_arg: when_wpt_dist(m)            10    # how close to waypoint before surface,
b_arg: end_action(enum)            1     # 0-quit, 1 wait for ^C quit/resume,
                                         # 2 resume
b_arg: report_all(bool)            0     # T->report all sensors once, F->just gps
b_arg: gps_wait_time(s)            300   # how long to wait for gps
b_arg: keystroke_wait_time(s)      300   # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up when a surface request is made by the science computer.

behavior: surface

```
b_arg: args_from_file(enum)        10    # read from mafiles/surfac10.ma
b_arg: start_when(enum)            11    # BAW_SCI_SURFACE
b_arg: end_action(enum)            1     # 0-quit, 1 wait for ^C quit/resume,
                                         # 2 resume
b_arg: report_all(bool)            0     # T->report all sensors once, F->just gps
b_arg: gps_wait_time(s)            300   # how long to wait for gps
b_arg: keystroke_wait_time(s)      300   # how long to wait for control-C
```

The following *behavior: surface* instructs the glider to come up every 10 minutes. In this particular case/mission, it is commented out, and therefore not active.

behavior: surface

```
# b_arg: args_from_file(enum)      10    # read from mafiles/surfac10.ma
# b_arg: start_when(enum)          9     # 0-immediately, 1-stack idle, 2-depth idle,
                                         # 6-when_secs, 7-when_wpt_dist,
                                         # 8-when hit waypoint, 9-every when_secs
# b_arg: when_secs(s)              600   # How long between surfacing, only if
                                         # start_when=6 or 9
# b_arg: when_wpt_dist(m)          10    # how close to waypoint before surface
# b_arg: end_action(enum)          1     # 0-quit, 1 wait for ^C quit/resume,
                                         # 2-resume
# b_arg: report_all(bool)          0     #T->report all sensors once, F->just gps
# b_arg: gps_wait_time(s)          300   # how long to wait for gps
# b_arg: keystroke_wait_time(s)    300   # how long to wait for control-C
```

The following *behavior: goto_list* tells the glider where its waypoints are. For this case, we have again used the .ma convention, which allows us to write a more general mission and insert the particular waypoints/coordinates to where the glider will glide.

behavior: goto_list

```
b_arg: args_from_file(enum)        10    # read from mafiles/goto_l10.ma
b_arg: start_when(enum)            0     # 0-immediately, 1-stack idle 2-heading idle
```

The corresponding .ma file is displayed below.

```
behavior_name=goto_list
# Written by gen-goto-list-ma ver 1.0 on GMT:Tue Feb 19 18:56:54 2002
# 07-Aug-02 tc@DinkumSoftware.com Manually edited for spawars 7aug02 op in buzzards bay
# 07-Aug-02 tc@DinkumSoftware.com Changed from decimal degrees to degrees, minutes,
# decimal minutes
# goto_l10.ma
# Flies a hexagon around R4
<start:b_arg>
    b_arg: num_legs_to_run(nodim)  -1    # loop
    b_arg: start_when(enum)        0     # BAW_IMMEDIATELY
    b_arg: list_stop_when(enum)    7     # BAW_WHEN_WPT_DIST
    b_arg: initial_wpt(enum)       -2    # closest
b_arg: num_waypoints(nodim)        6
<end:b_arg>
<start:waypoints>
-7040.271 4138.861
-7040.271 4138.807
-7040.333 4138.780
-7040.395 4138.807
-7040.395 4138.861
-7040.333 4138.888
<end:waypoints>
```

The following *behavior: yo* instructs the glider to perform a *yo* (i.e., a single up and down pattern through the water column). Again, the .ma convention is used to designate the depth and altitude (together: the RANGE) over which the yo is to be performed.

behavior: surface

```
b_arg: args_from_file(enum)        10    # read from mafiles/yo10.ma
b_arg: start_when(enum)            2     # 0-immediately, 1-stack idle 2-depth idle
b_arg: end_action(enum)            2     # 0-quit, 2 resume
```

The corresponding .ma file is displayed below.

```
behavior_name=yo
# yo-c3x5-d20-a3-p20.ma
# climb 3.5m dive 10m alt 20m pitch 20 deg
# Hand Written
# 18-Feb-02 tc@DinkumSoftware.com Initial
# 13-Mar-02 tc@DinkumSoftware.com Bug fix, end_action from quit(0) to resume(2)
# 03-aug-02 tc@DinkumSoftware.com DREA01 at ashument, went to depth only
<start:b_arg>
b_arg: start_when(enum)                 2           # pitch idle (see doco below)
b_arg: num_half_cycles_to_do(nodim)     -1          # Number of dive/climbs to perform
                                                    # <0 is infinite, i.e. never finishes


# arguments for dive_to
    b_arg: d_target_depth(m)            5
    b_arg: d_target_altitude(m)         -1
    b_arg: d_use_pitch(enum)            3           #      1:battpos 2:setonce 3:servo
                                                    #            in        rad       rad
                                                    # <0 dive
    b_arg: d_pitch_value(X)             -0.3491     # -20 deg

# arguments for climb_to
    b_arg: c_target_depth(m)            3.5
    b_arg: c_target_altitude(m)         -1
    b_arg: c_use_pitch(enum)            3           #      1:battpos 2:setonce 3:servo
                                                    #            in        rad       rad
                                                    # >0 climb
    b_arg: c_pitch_value(X)             -0.3491     # 20 deg
    b_arg: end_action(enum)             2           # 0-quit, 2 resume
<end:b_arg>
```

The following *behavior: prepare_to_dive* instructs the glider to get ready to dive after waiting for as long as 12 minutes for a gps fix.

behavior: prepare to dive

```
b_arg: start_when(enum)     1       # 0-immediately, 1-stack idle,
                                    # 2-depth idle
b_arg: wait_time(s)         720     # 12 minutes, how long to wait for
                                    # gps
```

The following ‘behavior: sensors_in’ turns on most of the input sensors.

behavior: sensors_in

Glider Software Website

In order to download the glider code from the repository, you must arrange for access through Teledyne Webb Research. This can be arranged by emailing glideraccess@webbresearch.com.

A user name and password will be provided or can be requested. Teledyne Webb Research will require the organization, name, phone number and email addresses of each person using access. To access documents log in at:

https://dmz.webbresearch.com/

The glider software website is:

http://www.glider-doco.webbresearch.com/

Glider code update procedure:

http://www.glider-doco.webbresearch.com/software-howto/updating-all-glider-software.txt

Index

Symbols

! command, 8-2
? command, 6-5, M-21

A

abbreviations list, A-1 to A-2
abend argument, I-1
abort codes, C-1 to C-2
abort sequences, B-2 to B-4, M-36 to M-38
aborting missions, 6-11, C-1 to C-2, I-1 to I-2, M-27
accessing
    flight data, 8-1 to 8-2
    GliderDOS, 2-1
    operating mode, FreeWave transceivers, F-1, F-2
acoustic modems, 6-12 to 6-13, M-29
acronym list, A-1 to A-2
adtest.run, 6-3, M-19
aft batteries, location of, 1-5
aft hulls, architecture of, 1-5
aft tail cones, architecture of, 1-5
air bladders, 1-5, 1-16, 1-24 to 1-25
air pumps, 1-5, 1-16 to 1-17, 1-19, 2-1
air, activating burn wire in, 1-25
alkaline batteries,dangers of, 1-29
alkaline battery voltage, monitoring, I-1 to I-2
alloff.run, 6-3, M-19
almanacs, GPS locations, 2-1
altimeters, 1-10 to 1-12
amconnct.run, 6-12, M-29
angles, composite wings, 1-3
anodes, sacrificial, 1-27
antennas, FreeWave transceivers, F-3
appcmd.dat, 6-13, M-29, M-30
architecture
    gliders, 1-3 to 1-6
    science data logging (SDL), 7-1 to 7-5, M-30 to M-34
    software
        GliderDOS, 6-5 to 6-6, M-21 to M-22
        masterdata, 6-6 to 6-11, M-23 to M-27
        overview, 6-1
        Persistor, 6-11 to 6-14, M-27 to M-30
        PicoDOS, 6-2 to 6-5, M-18 to M-21
        simulators, M-18 to M-30
Argos
    contact information, G-1
    data format of transmitters, H-2
    establishing satellite service and IDs with, H-1 to H-2

glider communication with, 5-1
Argos ALP, 4-1
Argos satellite platform transmitter terminal (PTT), 1-15 to 1-16, 1-19
Argos transmitters, location of, 1-5
arguments
behavior, B-1, B-5 to B-9, M-39 to M-43
See also specific argument names
ASCII files, combining, 7-3, M-32
assemblies
ballast pump
configuring, 1-7, 1-8
described, 1-6, 1-8
evaluating data, 1-7, 1-8
sensors, 1-8, 1-9
testing, 1-7, 1-8
ballast, location of, 1-4
communications board, modems on, 1-5
CTD sensor, 1-14 to 1-15
at command, 5-2
ata command, 5-2
ath command, 5-2
attitude sensors, 1-19 to 1-20
attitude sensors, location of, 1-5
attrib command, 6-8, M-24
autoexec.bat, 6-3, 6-5, 6-12, 6-14, M-19, M-21, M-29, M-30
autoexec.mi, 6-1, 6-2, 6-5, M-18, M-19, M-21

B

bad input, simulating, M-14 to M-15
<bad value> text line, M-14
bad_device, M-14 to M-15
ballast command, 6-8, M-24
ballast pump assemblies
configuring, 1-7, 1-8
described, 1-6, 1-8
evaluating data, 1-7, 1-8
location of, 1-4
sensors, 1-8, 1-9
testing, 1-7, 1-8
ballast weights, location of, 1-4
bam device, 6-12, M-28
batteries
dangers of, 1-29
location of, 1-4, 1-5
longevity during missions, I-1 to I-2
battery packs, 1-28 to 1-29
baud rates
changing, F-1, F-2
science data logging (SDL), 7-1, M-30
bays (payload), architecture of, 1-5
bb2c device, 6-12, M-28

bb2f device, 6-12, M-28
bb2lss device, 6-12, M-28
beaches, testing gliders on, 2-1 to 2-2
behavior arguments, B-1, B-5 to B-9, M-39 to M-43
benches, testing gliders on, 2-1 to 2-2
billing, Iridium SIM card services, G-2
bin folder, 6-3, 6-12 to 6-13, M-19 to M-20, M-29
bits, turning on and off, 6-3, M-19
bladders, air, 1-5, 1-16, 1-24 to 1-25
blast.run, 6-12, M-29
boards
    hardware interface, 1-19
    sensor, leak detect, 1-24
    switchboards, science and payload computers, 1-23
boats, testing gliders aboard, 2-1 to 2-2
boot app, 6-12
boot app command, 6-6, M-22
boot command, 6-8, M-24
boot pico command, 6-6, M-21, M-22
brakes, latching, 1-7
buoyancy
    air bladders, 1-24
    wings, 1-3
burn wires, 1-5, 1-19, 1-25 to 1-26
buying Iridium SIM cards, G-1 to G-2

C

c command, M-27
c_ prefix, 6-7, M-23
C programming language, 6-1, B-1, M-35
cables
    for communications, F-1
    serial, connecting simulators with, M-11
c_air_pump sensor, 1-17
callback 30 command, 2-1
callback 30 script, 4-1
callback command, 6-8, M-24
caps, nose end and tail, 1-3
capture command, 6-8, M-25
carbon fiber hulls, 1-13
cards (Iridium SIM), purchasing, G-1 to G-2
c_argos_on sensor, 1-16
carrier detect selection settings, M-10
carrier detection, loss of, 6-3, M-19
carts, 3-1
catalysts, 1-5, 1-16
c_att_time sensor, 1-20
c_ballast_pumped sensor, 1-8
c_battpos sensor, 1-10
cd command, 6-8, M-25
c_de_oil_vol sensor, 1-9

cellphone transmission towers, F-3
chassis, location of, 1-5
checking for device and sensor errors, 2-1
checklists, downloading, K-1
chkdsk command, 6-8, M-25
circuits, cop tickle, 1-19
C_IRIDIUM_TIME_TIL_CALLBACK, maximum legal value for, 4-1
clotheslines. *See* science data logging (SDL)
clrdeverrs command, 6-8, M-25
clr_drift_table command, 6-8
CLS America, contact information, G-1
code
    abort, C-1 to C-2
    design of, B-1 to B-4, M-35 to M-38
    mission sample and comments, B-4 to B-9
command line processing, 7-3, M-32
commanded prefix, 6-7, M-23
commands
    device, 6-8 to 6-11, M-24 to M-27
    GliderDOS, 7-2, M-31 to M-32
    SciDOS, 7-2 to 7-3, M-31 to M-32
    sensor, 6-7, M-23 to M-24
    using during deployment, 1-6, 2-1
    See also specific command names
comments, sample mission, B-4 to B-9, M-39 to M-43
communications
    cables used in, F-1
    testing, 6-12, M-29
    verbose mode, 2-1
    with gliders, 5-1 to 5-2
    with the science Persistor, 6-11 to 6-12, J-1 to J-2, M-27 to M-28
communications board assemblies, modems on, 1-5
composite wings, angle and replacement of, 1-3
computers
    science bay, 1-13 to 1-14
    science. *See* Persistor
    switchboards, 1-23
conductivity, temperature, and depth (CTD) sensors, 1-14 to 1-15
cones, aft tail, 1-5
config folder, 6-2 to 6-3, 6-13, M-19, M-29
config.sci, 6-2, M-19
configuration
    air pumps, 1-17
    altimeters, 1-10
    Argos satellite transmitter (PTT), 1-15
    attitude sensors, 1-19
    ballast pump assemblies, 1-7, 1-8
    battery packs, 1-28
    carbon fiber hulls, 1-13
    conductivity, temperature, and depth (CTD) sensors, 1-14
    coulomb counters, 1-28

desiccants, 1-30
global positioning system (GPS), 1-20
hardware interface board, 1-19
Iridium satellite telemetry, 1-21
pitch vernier, 1-9
pressure transducers, 1-23
science bay computers and sensors, 1-13
strobes, 1-31
switchboards, science and payload computers, 1-23
vehicle controllers, 1-18
configuration files, science and glider processors, 7-2
confirmation messages, mission end, 2-1
connecting with HyperTerminal sessions, F-1, F-2
connections
serial data. *See* science data logging (SDL)
simulators, M-8 to M-12
consci command, 1-23, 6-8, 6-11, J-1, M-25, M-27
consci.run, 6-3, M-19
conserving energy during emergency recovery operations, 4-1
contact information, Iridium SIM card distributors and service providers, G-1 to G-2
control levels, B-2 to B-4, M-36 to M-38
control-c command, 5-1, 6-6, 10-1, M-22
controllers, vehicle, 1-17 to 1-18
cop tickle circuit, 1-19
COP_TICKLE, B-4, M-38
copy command, 6-9, M-25
coulomb counters, 1-28 to 1-29
countdowns, resuming missions, 8-2
counters, coulomb, 1-28 to 1-29
cp command, 6-9, M-25
c_pitch sensor, 1-10
cranes, overhead, 3-1
crc command, 6-9, M-25
c_recovery_on sensor, 1-30
c_science_send_all sensor, 7-4, M-33
c_strobe_ctrl sensor, 1-31
CTD (conductivity, temperature, and depth) sensors, 1-14 to 1-15
CTD program, 6-12, M-29
ctd41 device, M-28
ctd41cp device, 6-12, M-28
ctd_ctrl.run, 6-12, M-29
ctd_hs2.run, 6-13, M-29
CTDs, running dry, 1-14
ctrl-c command, 2-1, 2-2, 8-2
ctrl-e command, 8-2
ctrl-f command, 8-2
ctrl-p command, 8-2
ctrl-r command, 8-2
ctrl-t command, 6-11, 8-2, J-1
currents, effects on voltage, I-1
c_weight_drop sensor, 1-26

cycle rates, science processor, 7-1, M-30
cycles of missions, adjusting frequency of, 4-1

D

data
    evaluating
        air bladders, 1-25
        air pumps, 1-17
        altimeters, 1-11
        Argos satellite transmitter (PTT), 1-15
        attitude sensors, 1-20
        ballast pump assemblies, 1-7, 1-8
        battery packs, 1-29
        carbon fiber hulls, 1-13
        conductivity, temperature, and depth (CTD) sensors, 1-15
        coulomb counter, 1-29
        global positioning system (GPS), 1-20 to 1-21
        Iridium satellite telemetry, 1-22
        leak detect sensor boards, 1-24
        pitch vernier, 1-10
        pressure transducers, 1-23
        science bay computers and sensors, 1-14
        vehicle controllers, 1-18
data files, sending to Dockserver, 7-3, M-32
data format, Argos transmitters, H-2
data from flights, retrieving from Dockserver, 8-1 to 8-2
data plots, alkaline battery voltage, I-2
Data Server, 7-3, M-32
Data Visualizer, 7-3, 9-1, M-32
date command, 6-9, M-25
.dbd files, 6-4, 7-1, 8-1, M-20, M-21, M-31
deactivating PINs, Iridium SIM cards, G-2
deep pumps, service life of, 1-6
default values, finding, 1-6
del command, 6-9, M-25
deleting files before flights, 6-3, 6-13, M-19, M-30
dellog command, 6-9, 7-2, 7-3, M-25, M-31, M-32
deployment
    commands to avoid during, 1-6, 2-1
    gliders, 3-1, 6-6, M-21
desiccants, 1-30
designs, gliders, 1-1, M-7
device commands, 6-8 to 6-11, M-24 to M-27
devices
    checking for errors, 2-1
    connected to science computer, 6-12, M-28
    flotation, deploying gliders with, 2-2
devices, testing during glider deployment, 1-6
devices? command, 6-9, M-25
df command, 6-9, 7-2, M-25, M-31
digifin (digital tail fin), 1-5, 1-26 to 1-27

digifin command, 6-9
digital tail fin (digifin), 1-5, 1-26 to 1-27
Dinkum binary data files (.dbd), 8-1
Dinkum binary data files. *See* .dbd files, 6-4, M-20
dir command, 6-9, M-25
direct science connection, M-9
disk operating system (DOS), compatibility with PicoDOS, B-1, M-35
displacement pumps, running, 1-7
Dockserver
    behavior comparison, simulators vs. gliders, M-12
    callback 30 script, 4-1
    connecting simulators to, M-11 to M-12
    Data Visualizer, 9-1
    glider communications with, 5-1 to 5-2
    retrieving flight data from, 8-1 to 8-2
    sending data files to, 7-3, M-32
domes, nose, 1-4
done, M-13
DOS (disk operating system), compatibility with PicoDOS, B-1, M-35
downloading
    glider software, D-1, M-44
    quick references and checklists, K-1
drivers, uart, 6-3, M-20
dump command, 6-9, M-25

E

.ebd files, 7-1, M-31
editing proglet.dat files, J-1 to J-2
ejection weights, 4-1
emergency recovery, gliders, 4-1
end caps, nose, 1-3
end of missions, confirmation messages, 2-1
energy savings, during emergency recovery operations, 4-1
equipment, testing during glider deployment, 1-6
erase command, 6-9, M-25
errors, checking in devices and sensors, 2-1
estimates, battery longevity, I-2
exit command, 6-9, M-25
exit pico command, 6-6, M-22
exit reset command, 2-2, 6-6, I-2, M-22
expressions (surface), GliderDOS, 6-5, M-21
external power connector, M-9
external setting, M-10

F

f_ prefix, 6-7, M-23
factory prefix, 6-7, M-23
f_coulomb_battery_capacity sensor, I-1
f_crush_depth sensor, 1-26
files
    configuration, science and glider processors, 7-2

deleting before flights, 6-3, 6-13, M-19, M-30
log, science and glider processors, 7-1 to 7-3, M-31 to M-32
mission, 10-1
PicoDOS, 6-2 to 6-5, M-19
running send commands for large numbers of, 7-5, M-34
See also specific file names
f_leakdetect_threshold sensor, 1-24
flights
deleting files before, 6-3, 6-13, M-19, M-30
gliders, 1-3
retrieving data from Dockserver, 8-1 to 8-2
using conductivity, temperature, and depth (CTD) sensors for, 1-14
flotation devices, deploying gliders with, 2-2
folders
PicoDOS, 6-2 to 6-5, M-19
science computer, 6-12 to 6-13, M-29
format of data, Argos transmitters, H-2
forward hulls, architecture of, 1-4
FreeWave modems
configuration of, F-1 to F-3
connecting simulators with, M-11 to M-12
glider communications with, 5-1
simulating, M-14
telemetry, 1-22
frequency keys, setting, F-1, F-3
frequency of missions, adjusting, 4-1
front panel settings, simulators, M-10

G

get command, 6-9, 7-2, M-25, M-31
get sensor_name command, 6-7, M-23
glider and science persistor reset settings, M-10
Glider Endurance Tool, I-2
glider freewave connection, M-9
glider sensors, undefined, 1-6
glider simulators. *See* simulators
glider software, downloading, D-1, M-44
glider variables. *See* masterdata
glider.app, 6-5, M-21
GliderDOS
accessing, 2-1
commands, 7-2, M-31 to M-32
described, 6-1, 6-5 to 6-6, B-1, M-18, M-21 to M-22, M-35
editing proglet.dat files, J-1 to J-2
sending flight data files in, 8-2
See also Persistor
GliderDOS prompts, returning to, 2-2
gliders
adding FreeWave transceiver serial numbers to, F-2, F-3
architecture of, overview, 1-3 to 1-6
capabilities of, 1-1, M-7

communications with, 5-1 to 5-2
components of, 1-6 to 1-31
data format, Argos transmitters, H-2
deployment of, 3-1, 6-6, M-21
designs of, 1-1, M-7
differences between simulators and, M-12
downloading software for, D-1, M-44
navigation of, 1-3
operating systems for, B-1, M-35
propulsion of, 1-2
recovery of, 3-1, 4-1, I-2
simulation of, M-13 to M-17
software architecture
GliderDOS, 6-5 to 6-6, M-21 to M-22
masterdata, 6-6 to 6-11, M-23 to M-27
overview, 6-1
Persistor, 6-11 to 6-14, M-27 to M-30
PicoDOS, 6-2 to 6-5, M-18 to M-21
testing equipment during deployment, 1-6
testing, pre-mission, 2-1 to 2-2
vehicle status, 8-2
wiring diagrams, L-1
global positioning system (GPS), 1-20 to 1-21, 2-1
global positioning system (GPS) modems, locations of, 1-5
GPS (global positioning system), 1-20 to 1-21, 2-1
GPS (global positioning system) modems, locations of, 1-5

H

hardware generated abort, B-4, M-38
hardware interface boards, 1-19, 6-3, M-19
hardware interface boards, testing, 6-3, M-19
hardware? command, 6-9, M-25
hazards, batteries, 1-29
heap command, 6-9, 7-2, M-25, M-31
height, effects on FreeWave transceiver communications, F-3
help command, 6-2, 6-5, 6-8, 6-9, M-21, M-24, M-25
help ctd_hs2 command, 6-13, M-29
help, printing, 6-7, M-23
highdensity command, 6-9, M-25
H-moment, adjustments to, 1-5
hooks on poles, 3-1
horizontal moment (H-moment), adjustments to, 1-5
hs2 device, 6-12, M-28
hulls
architecture of, 1-4 to 1-5
carbon fiber, 1-13
HyperTerminal sessions, connecting with, F-1, F-2

I

ID (identification) numbers, establishing with Argos, H-1 to H-2
identification (ID) numbers, establishing with Argos, H-1 to H-2

Infosat Communications, contact information, G-1
input (bad), simulating, M-14 to M-15
installing wings, 1-3
interference with FreeWave transceivers, causes of, F-3
invoicing, Iridium SIM card services, G-2
Iridium modem connection, M-9
Iridium modems
    connecting simulators with, M-12
    locations of, 1-5
    simulating, M-14
Iridium phones, 2-1, 5-1 to 5-2
Iridium satellite telemetry, 1-21 to 1-22
Iridium SIM cards, purchasing, G-1 to G-2

J

jettison weights, 1-5, 1-19, 1-25 to 1-26
JouBeh Technologies, contact information, G-1
just_electronics simulation level, M-13, M-16 to M-17

K

keys (frequency), setting, F-1, F-3

L

lab environments, entering PicoDOS in, 5-1
labels, sensors, 6-1
lab_mode command, 6-9, M-25
    avoiding during deployment, 1-6, 2-1
    testing motors, 2-1
lassos, 3-1
latching brakes, 1-7
leak detect sensor boards, 1-24
life of batteries
    during missions, I-1 to I-2
life of service, pumps, 1-6
list command, 6-5, 6-7, 6-9, 7-2, M-21, M-23, M-25, M-31
lithium battery voltage, monitoring, I-1
loadmission command, 1-13, 6-9, M-17, M-25
loadmission waterclr.mi command, 2-2
loadsim.mi, M-17
.log files, 6-4, 8-1, M-20
logfiles, science and glider processors, 7-1 to 7-3, M-31 to M-32
logflie names, 7-3, M-32
logging command, 6-9, M-25
logging on command, 7-3
logging, science data (SDL), 7-1 to 7-5, M-30 to M-34
logs folder, 6-4, M-20
longevity of batteries
    during missions, I-1 to I-2
longterm command, 6-9, M-26
longterm.dat, 6-1, M-18

longterm_put command, 6-9, M-25
lp stop command, 6-10
ls command, 6-10, M-26

M

m_ prefix, 6-7, M-23
.ma files, 6-1, 10-1, B-5, B-8, M-18, M-20, M-39 to M-43
mafiles folder, M-20
m_air_pump sensor, 1-17
m_altitude sensor, 1-11
masterdata
    described, 6-1, 6-6 to 6-11, M-18, M-23 to M-27
    location of, 1-6
    production copy, location of, 6-5
<max mt> text line, M-14
m_ballast_pumped sensor, 1-8
m_battery sensor, 1-29, I-1
m_battery_inst sensor, 1-29
m_battpos sensor, 1-10
mbd command, 6-10, M-26
.mbd files, 6-4, 7-1, 8-1, M-20, M-31
mbdlist.dat, 6-3, 6-4, 7-2, 8-1, M-19, M-20, M-31
mcmd.run, 6-12, M-29
m_coulomb_amphr_total sensor, 1-29, I-1, I-2
m_coulomb_current sensor, 1-29
mdatacol.run, 6-13, M-29
m_de_oil_vol sensor, 1-9
m_depth command, 1-23
m_depth sensor, 1-23
m_depth_state sensor, 1-24
measured prefix, 6-7, M-23
medium binary data files (.mbd), 6-4, 8-1, M-20
menus, Operation Mode, F-1, F-2
merge tool, combining ASCII files, 7-3, M-32
Metrowerks CodeWarrior, B-1, M-35
m_heading sensor, 1-20
.mi files, 6-1, 6-4, 10-1, B-1, M-18, M-20, M-35
micron pressure transducers, location of, 1-5
mid-hulls, architecture of, 1-5
<min mt text line>, M-14
m_iridium_signal_strength sensor, 1-22
mission acquisition files. *See* .ma files
mission files, 6-1
mission files. *See* .mi files
mission files. *See* .mlg files, M-20
mission log files (.mlg), 6-4, 8-1
missions
    aborting, 6-11, C-1 to C-2, I-1 to I-2, M-27
    adjusting cycle frequency, 4-1
    battery longevity, I-1 to I-2
    changes to, 6-1

countdowns to resuming, 8-2
end of, confirmation messages, 2-1
pre-written, 6-1
retrieving flight data from, 8-1 to 8-2
running, 10-1 to 10-2
sample of, B-4 to B-9, M-39 to M-43
stock, starting, 2-2
testing gliders before, 2-1 to 2-2
missions folder, 6-4, M-20
mkdir command, 6-10, M-26
m_leakdetect_voltage sensor, 1-24
m_leakdetect_voltage_forward sensor, 1-24
.mlg files, 6-4, 7-1, 8-1, M-20, M-21, M-31
m_lithium_battery_relative_charge command, 1-29
m_lithium_battery_relative_charge sensor, I-1
m_mission_start_time command, 7-4, M-33
mobile phone transmission towers, F-3
modem cables, F-1
modem parameter, M-17
modems
acoustic, 6-12 to 6-13, M-29
connections, M-9
FreeWave
configuration of, F-1 to F-3
connecting simulators with, M-11 to M-12
glider communications with, 5-1
simulating, M-14
telemetry, 1-22
Iridium
connecting simulators with, M-12
simulating, M-14
locations of, 1-5
on communications board assemblies, 1-5
radio frequency (RF)
configuration of, F-1 to F-3
glider communications with, 5-1, 6-3, M-19
location of, 1-5
telemetry, 1-22
modes
operation, FreeWave transceivers, F-1, F-2
setup, activating on FreeWave transceivers, F-1, F-2
verbose, 2-1
Motocross, B-1, M-35
Motorola, B-1, M-35
motors
testing, 2-1
motors, moving, 6-11, M-27
moving motors, 6-11, M-27
m_pitch sensor, 1-10, 1-20
m_present_time value, 7-2, M-31
m_raw_altitude sensor, 1-11

m_roll sensor, 1-20
m_science_readiness_for_consci command, 7-4, M-33
m_strobe_ctrl sensor, 1-31
mv command, 6-10, M-26
m_vacuum sensor, 1-17
m_water_depth sensor, 1-12

N

NAL Research, contact information, G-1
names of sensors, printing, 6-7, M-23
navigation devices, testing, 6-2
navigation, gliders, 1-3
.nbd files, 7-1, M-31
nbdlist.dat, 7-2, M-31
.nlg files, 7-1, M-31
no_electronics simulation level, M-13, M-17
noise problems with FreeWave transceivers, causes of, F-3
nose domes, architecture of, 1-4
nose end caps, shape of, 1-3
null modem cables, F-1
null_modem parameter, M-14, M-17

O

ocean pressure, zero, 6-11, M-27
oceans, deploying and recovering gliders in, 3-1
off setting, M-10
on setting, M-10
on_bench open argument, M-16
on_bench simulation level, M-13, M-16
operating systems. *See specific operating system names*
Operation mode menu, accessing, F-1, F-2
out of band abort, B-4, M-38
outdoors, conducting tests in, 2-1
overhead cranes, 3-1

P

packs, battery, 1-28 to 1-29
pager transmission towers, F-3
parameters, simulation, M-17
path command, 6-10, M-26
payload bays, architecture of, 1-5
payload computers, switchboards, 1-23
pbm reset setting, M-10
performance, science processors, 7-1, M-30
Persistor
    described, B-1, M-35
    hardware interface boards, 1-19
    software architecture, 6-11 to 6-14, M-27 to M-30
    See also GliderDOS
Persistor Instruments, B-1, M-35

Persistor Instruments Card Or Disk Operating System. *See* PicoDOS
phone transmission towers, F-3
phones, Iridium, 2-1, 5-1 to 5-2
pick points, 1-30, 3-1
Pico mode command, avoiding during deployment, 2-1
PicoDOS
    architecture of, 6-2 to 6-5, M-18 to M-21
    compability with DOS, B-1, M-35
    described, 6-1, B-1, M-18, M-35
    editing proglet.dat files, J-1 to J-2
    reasons to use, 6-5, M-21
    sending flight data files in, 8-2
    setting boot commands in, M-22
    testing Iridium in, 5-1
    when to enter, 5-1
PINs for unlocking Iridium SIM cards, G-2
pitch vernier, 1-4, 1-9 to 1-10
pitch, control of, 1-4
plots, alkaline battery voltage data, I-2
points, pick, 1-30
poles, hooks and lassos on, 3-1
ports, testing, 6-12, M-29
power on/off/science selection settings, M-10
power umbilicals, 1-26
power umbilicals, location of, 1-5
prefixes, masterdata, 6-7, M-23
pre-mission testing, 2-1 to 2-2
pressure transducers, 1-23 to 1-24
pressure transducers (micron), location of, 1-5
printing
    help, 6-7, M-23
    sensor names and values, 6-7, M-23
<probability> text line, M-14
processing, command line, 7-3, M-32
processors, science, 7-1 to 7-5, M-30 to M-34
proglet.dat, editing, J-1 to J-2
proglets.dat, 1-5, 6-12 to 6-13, M-28, M-29
programming languages, C, 6-1, B-1, M-35
prompt command, 6-10, M-26
propellers, 1-2
propulsion, gliders, 1-2
prune command, 7-4
prunedisk command, 6-10, 7-2, 7-3, M-26, M-31, M-32
PTT (Argos satellite platform transmitter terminal), 1-15 to 1-16, 1-19
pumps
    air, 1-5, 1-16 to 1-17, 1-19, 2-1
    ballast
        configuring, 1-7, 1-8
        described, 1-6, 1-8
        evaluating data, 1-7, 1-8
        sensors, 1-8, 1-9

testing, 1-7, 1-8
displacement, running, 1-7
service life of, 1-6
testing, 2-1
purchasing Iridium SIM cards, G-1 to G-2
purgelogs command, 6-10, M-26
put c_air_pump 0 command, 2-1
put c_gps_on commands, 2-1
put command, 6-10, 7-2, I-2, M-26, M-32
put c_recovery_on 1 command, 3-1
put sensor_name value command, 6-7, M-23

Q

quick references, downloading, K-1

R

radio frequency (RF) modems
configuration of, F-1 to F-3
glider communications with, 5-1, 6-3, M-19
location of, 1-5
telemetry, 1-22
raw voltages, 6-3, M-19
recovery systems, 1-30
recovery, gliders, 3-1, 4-1, I-2
reinitialization of sensor values, forcing, 2-2
removing wings, 1-3
rename command, 6-10, M-26
report - sensor_name command, 6-7, M-23
report ? command, 6-7, M-23
report + sensor_name command, 6-7, M-23
report ++ sensor_name command, 6-7, M-23
report all command, 6-7, M-24
report clearall command, 6-7, M-24
report command, 6-10, M-26
report list command, 6-7, M-24
reset setting, M-10
results from tests, troubleshooting, 1-6
RF modems. *See* radio frequency modems
rm command, 6-10, M-26
rmdir command, 6-10, M-26
RS232 cables, F-1
RUDICS, glider communications with, 5-2
run command, 6-10, M-26
run ini commands, 2-2
run status.mi command, 2-1, 2-2
run stock.mi command, 2-2

S

s command, 7-3, M-32
s_ prefix, 6-7, M-23

sacrificial anodes, 1-27
sam device, 6-12, M-28
sampling, 6-12, M-29
satellite service, establishing with Argos, H-1 to H-2
satellite telemetry, Iridium, 1-21 to 1-22
satellite transmitters
    PTT, 1-15 to 1-16, 1-19
saving energy during emergency recovery operations, 4-1
sbd command, 6-10, M-26
.sbd files, 6-4, 7-1, 8-1, M-20, M-21, M-31
sbdlist.dat, 6-3, 6-4, 7-2, M-19, M-20, M-31
SBE (Seabird Electronics) pumped CTDs, running dry, 1-14
#sci_ prefix, 6-7, M-23
sci setting, M-10
SciDOS commands, 7-2 to 7-3, M-31 to M-32
science bay computers, 1-13 to 1-14
science bay sensors, 1-13 to 1-14
science bays, simulation in, M-17
science computer. *See* Persistor
science data logging (SDL), 7-1 to 7-5, M-30 to M-34
science processors, 7-1 to 7-5, M-30 to M-34
science sensors, 1-5
science variable prefix, 6-7, M-23
sci_m_disk_free command, 7-4, M-33
sci_m_disk_usage command, 7-4, M-33
sci_m_free_heap command, 7-4, M-33
sci_m_min_free_heap command, 7-4, M-33
sci_m_min_spare_heap command, 7-4, M-33
sci_m_present_secs_into_mission command, 7-4, M-33
sci_m_present_time value, 7-2, M-31
sci_m_spare_heap command, 7-4, M-33
sci_water_cond sensor, 1-15
sci_water_pressure sensor, 1-15, 1-23
sci_water_temp sensor, 1-15
sci_x_disk_files_removed command, 7-4, M-33
sci_x_sent_data_files command, 7-4, M-33
scripts, conserving energy with, 4-1
SDL (science data logging), 7-1 to 7-5, M-30 to M-34
Seabird Electronics (SBE) pumped CTDs, running dry, 1-14
seas, deploying and recovering gliders in, 3-1
Seimac X-Cat PTT.*See* Argos satellite transmitter
send command, 6-10, 7-2, 7-3, 7-5, 8-2, M-26, M-32, M-34
sensor assemblies, CTD, 1-14 to 1-15
sensor boards, leak detect, 1-24
sensor commands, 6-7, M-23 to M-24
sensor names, printing, 6-7, M-23
sensor values, forcing reinitialization of, 2-2
sensors
    air bladders, 1-25
    air pumps, 1-17
    altimeters, 1-11 to 1-12

Argos satellite transmitter (PTT), 1-16
attitude, 1-19 to 1-20
attitude, location of, 1-5
ballast pump assemblies, 1-8, 1-9
battery packs, 1-29
burn wires, 1-26
checking for errors, 2-1
coulomb counter, 1-29
for monitoring battery voltage, I-1
glider, undefined, 1-6
global positioning system (GPS), 1-21
Iridium satellite telemetry, 1-22
labeling of, 6-1
leak detect sensor boards, 1-24
pitch vernier, 1-10
pressure transducers, 1-23
recovery systems, 1-30
science, 1-5
science bay, 1-13 to 1-14
simulators, 7-4 to 7-5, M-33 to M-34
vehicle controllers, 1-18
See also specific sensor names
sent command, 7-4
sentlogs folder, 6-4, M-21
sequence command, 6-10, M-26
serial cables, connecting simulators with, M-11
serial data connections. *See* science data logging (SDL)
serial numbers of FreeWave transceivers, adding to gliders, F-2, F-3
service life, deep and shallow pumps, 1-6
service life, pumps, 1-6
set command, 6-10, M-26
set in factory prefix, 6-7, M-23
setdevlimit command, 6-10, M-26
setnumwarn command, 6-10, M-26
setup mode, activating on FreeWave tranceivers, F-1, F-2
shallow pumps, service life of, 1-6
shapes, tail caps, 1-3
shoebox simulators. *See* simulators
short binary data files (.sbd), 8-1
short binary data files. *See* .sbd files, 6-4, M-20
SIM cards (Iridium), purchasing, G-1 to G-2
simul? command, 6-10, M-26
simul.sim, 6-3, M-13, M-14 to M-15, M-19
simulated state variables prefix, 6-7, M-23
simulations, process of, M-13 to M-17
simulators
connections for, M-8 to M-12
differences between gliders and, M-12
front panel settings, M-10
sensors added to, 7-4 to 7-5, M-33 to M-34
software control hierarchy, M-18 to M-30

slaves, communication between FreeWave transmitters and, F-1 to F-3
software
  architecture of
    GliderDOS, 6-5 to 6-6, M-21 to M-22
    masterdata, 6-6 to 6-11, M-23 to M-27
    overview, 6-1
    Persistor, 6-11 to 6-14, M-27 to M-30
    PicoDOS, 6-2 to 6-5, M-18 to M-21
    simulators, M-18 to M-30
  glider, downloading, D-1, M-44
speed, science processors, 7-1, M-30
srf_display command, 6-10, M-26
srtest.run, 6-3, M-19
stability, air bladders, 1-24
stacks, B-2 to B-4, M-36 to M-38
states, B-2 to B-4, M-36 to M-38
stock missions, starting, 2-2
Stratos, contact information, G-1
strobe command, 6-11
strobes, 1-31
surface expressions, GliderDOS, 6-5, M-21
switchboards, science and payload computers, 1-23
synchronous abort, B-3 to B-4, M-38
sync_time command, 6-11, M-26
system requirements, science processors, 7-1, M-30

T

tail caps, shape of, 1-3
tail cones, aft, 1-5
tail fins,fins, tail, 1-5, 1-26 to 1-27
talk arg command, 6-2
talk att command, 6-2
talk gps command, 6-2
talk help command, 6-3, M-20
talk iridium command, 5-1, 6-2
talk.run, 6-3, M-20
.tbd files, 7-1, M-31
tbdlist.dat, 7-2, M-31
tcm3 command, 6-11, M-26
telemetry
  Iridium satellite, 1-21 to 1-22
  radio frequency (RF) modems, 1-22
temperature, vacuum fluctuation and, 1-13
testing
  air bladders, 1-25
  air pumps, 1-17
  altimeters, 1-10 to 1-11
  Argos satellite transmitter (PTT), 1-15
  attitude sensors, 1-20
  ballast pump assemblies, 1-7, 1-8
  battery packs, 1-29

burn wires, 1-25
carbon fiber hulls, 1-13
communications for, 6-12, M-29
conductivity, temperature, and depth (CTD) sensors, 1-14
coulomb counter, 1-29
desiccants, 1-30
gliders, pre-mission, 2-1 to 2-2
global positioning system (GPS), 1-20, 2-1
hardware interface board, 1-19
hardware interface boards, 6-3, M-19
Iridium in PicoDOS, 5-1
Iridium satellite telemetry, 1-22
leak detect sensor boards, 1-24
motors, 2-1
navigation devices, 6-2
pitch vernier, 1-9 to 1-10
pressure transducers, 1-23
pumps, 2-1
radio frequency (RF) modem telemetry, 1-22
recovery systems, 1-30
science bay computers and sensors, 1-13
strobes, 1-31
switchboards, science and payload computers, 1-23
troubleshooting results from, 1-6
time command, 6-11, M-27
towers, pager and cellphone transmission, F-3
trajectory, gliders, 1-2
transceivers, FreeWave
configuration of, F-1 to F-3
connecting simulators with, M-11 to M-12
glider communications with, 5-1
simulating, M-14
transducers
pressure, 1-23 to 1-24
transducers (micron pressure), location of, 1-5
transmitters
Argos
data format of, H-2
location of, 1-5
platform transmitter terminal (PTT), 1-15 to 1-16, 1-19
X-cat, H-2
transmitters, FreeWave
telemetry, 1-22
transparency, science data logging (SDL), 7-2 to 7-3, M-32
troubleshooting
interference with FreeWave transceivers, F-3
mission aborts, C-1 to C-2, I-1 to I-2
test results, 1-6
turning off
air pumps, 2-1
bits, 6-3, M-19

turning on bits, 6-3, M-19
tvalve command, 6-11, M-27
type command, 6-11, M-27

U

u_ prefix, 6-7, M-23
u4talk.run, 6-13, M-29
u_abort_max_burn_time sensor, 1-26
u_abort_min_burn_time sensor, 1-26
u_alt_filter_enabled sensor, 1-11
u_alt_min_depth sensors, 1-11
u_alt_reduced_usage_mode sensor, 1-12
u_alt_reqd_good_in_a_row sensor, 1-11
uart drivers, 6-3, M-20
uarttest.run, 6-3, M-20
u_iridium_max_time_til_callback(sec) 1800.0 # value, 4-1
u_max_time command, 4-1
u_max_water_depth_lifetime sensor, 1-12
umbilicals, power, 1-5, 1-26
undervolts argument, I-1
unlocking Iridium SIM cards, G-2
u_sci_cmd_max_consci_time sensor, 7-5, M-34
u_sci_cycle_time command, 7-4, M-33
u_sci_dbd_sensor_list_xmit_control command, 7-4, M-33
u_science_send_time_limit_adjustment_ factor sensor, 7-5, M-34
u_science_send_time_limit_adjustment_factor (nodim) sensor, 7-5
use - device_name command, 6-8, M-24
use - Iridium command, M-14
use ? command, 6-8, M-24
use + dev_name command, 6-8, M-24
use all command, 6-8, M-24
use command, 6-8, 6-11, M-24, M-27
use none command, 6-8, M-24
user defined before run time prefix, 6-7, M-23
u_use_ctd_depth_for_flying sensor, 1-14

V

vacuum, temperature fluctuation and, 1-13
values
    default, finding, 1-6
    sensors
        commands for viewing and changing, 6-7, M-23
        forcing reinitialization of, 2-2
    See also specific value names
variables
    glider. *See* masterdata
    science, 6-7, M-23
    simulated state, 6-7, M-23
vehicle controllers, 1-17 to 1-18
vehicle status, gliders, 8-2
ver command, 6-11, M-27

verbose mode, 2-1
verifying device and sensor errors, 2-1
vernier, pitch, 1-4, 1-9 to 1-10
voltage
    batteries, I-1 to I-2
    raw, 6-3, M-19

W

warnings
    battery hazards, 1-29
    deploying and recovering gliders, 3-1
    running SBE pumped CTDs dry, 1-14
    when to enter PicoDOS, 5-1
water
    effects of currents on voltage, I-1
    setting currents to zero, 2-2
    testing gliders in, 2-2
weights
    ballast, location of, 1-4
    ejection, 4-1
    jettison, 1-5, 1-19, 1-25 to 1-26
where command, 6-11, M-27
whgpbm device, 6-12, M-28
whoru command, 6-11, M-27
whpar device, 6-12, M-28
why? command, 6-11, M-27
wiggle command, 6-11, M-27
wiggle off command, 2-1
wiggle on command, 2-1
wings
    architecture of, 1-6
    buoyancy of, 1-3
    composite, angle and replacement of, 1-3
    removing and installing, 1-3
wires, burn, 1-5, 1-19, 1-25 to 1-26
wiring diagrams, L-1

X

x_ prefix, 6-7, M-23
X-cat transmitters, H-2
x_science_logging_state command, 7-4, M-33

Y

yo, B-8

Z

zero ocean pressure, 6-11, M-27
zero, setting water currents to, 2-2
zero_ocean_pressure command, 1-23, 6-11, M-27
zmext.dat, 6-3, 6-13, M-19, M-29

Zmodem, 6-3, 6-11, 8-2, M-20, M-27
zr command, 6-11, M-27
zr.run, 6-3, 6-13, M-20, M-29
zs command, 6-11, M-27
zs.run, 6-3, 6-13, M-20, M-29

TELEDYNE
WEBB RESEARCH

A Teledyne Technologies Company

82 Technology Park Drive
East Falmouth, MA 02536 USA
+1 (508) 548-2077 (phone)
+1 (508) 540-1686 (fax)
www.webbresearch.com