

82 Technology Park Drive
E. Falmouth, Massachusetts 02536
Phone: 508.548.2077
Fax: 508.540.1686
glidersupport@webbresearch.com



Applies to: Electric gliders, type 30m, 100m, 200m and 1000m

Disclaimers and Battery Warning

Qualified personnel

Only trained and qualified personnel should operate and maintain the glider.

Teledyne Webb Research conducts regular training sessions several times a year. Glider users should attend a training session and understand basic glider concepts and terminology. Contact glidersupport@webbresearch.com for information regarding training sessions. Company policy is to fully support only properly trained individuals and groups.

Battery Warning

Gliders contain batteries comprised of alkaline manganese dioxide “C” cells.

There is a small but finite possibility that batteries of alkaline cells will release a combustible gas mixture. 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 mechanism, the gases can accumulate and an explosion is possible.

A catalyst inside the instrument recombines hydrogen and oxygen into H₂O, and the instrument has been designed to relieve excessive internal pressure by having the hull sections release.

Teledyne Webb Research knows of no way to completely eliminate this hazard. The user is warned, and must accept and assume responsibility for 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.

Teledyne Webb Research disclaims liability for any consequences of combustion or explosion.

Teledyne Webb Research (TWR) provides products with alkaline manganese dioxide (Duracell or equivalent) primary batteries. Some customers may choose to install other types of batteries. Any decision to use alternative batteries is made solely by the customer and is not the result of any endorsement or recommendation from TWR. Upon request, TWR may provide technical support, such as product energy usage, connector types, etc, to customers who wish to use alternative batteries. Any enclosure containing batteries presents inherent hazards. In ocean instruments, these hazards are aggravated by intrinsic risks of water leakage into the battery compartment. Teledyne Webb Research disclaims liability for any battery-related hazards including combustion or explosion.

DISCLAIMERS AND BATTERY WARNING	2
INTRODUCTION	8
FORMAT NOTES	8
SPECIFICATIONS	9
0 VEHICLE OPERATION THEORY	10
0.1 Forward Propulsion	10
0.2 Navigation and Flight	11
1 VEHICLE DESCRIPTION	11
1.1 Architecture	11
Nose Dome	11
Forward Hull Section	11
Payload Bay Mid Hull Section	11
Aft Hull Section	12
Aft Tail Cone	12
Wings	12
1.2 Specific Components	12
Ballast pump assembly 200 meter	12
Pitch Vernier	13
Altimeter	13
CTD	13
ARGOS Satellite Transmitter (PTT)	14
Catalyst	14
Air Pump System	14
Vehicle Controller	14
Hardware Interface Board	14
Attitude Sensor	15

GPS	15
Iridium Satellite Telemetry	15
RF modem Telemetry	15
Pinger (discontinued in 2007)	15
Science/Payload Computer Switch Board	15
Pressure Transducer	16
Leak detect	16
Air Bladder	16
Burn Wire	16
Jettison Weight	16
Power Umbilical	16
Antenna Fin/ Steering Assembly replaced by Digifin in 2008	17
Sacrificial anode	17
Batteries	17
2 OPENING PROCEDURE	18
2.1 Glider Hull Sections	18
2.2 Center Payload Bay	19
2.3 Ballast Pump Assembly	19
2.4 Nose Cone and Altimeter	20
3 CLOSING PROCEDURE	20
3.1 Nose Cone	20
3.2 Ballast pump assembly	20
3.3 Center Payload Bay	20
3.4 Glider Hull Sections	20
4 PRE MISSION TESTING	23
5 DEPLOYMENT AND RECOVERY	25

6	EMERGENCY RECOVERY	25
7	MAINTENANCE	26
	General	26
	O-rings	26
	Pressure transducer	26
	Burn Wire Assembly	26
	Dummy and Green Plug	26
	Sensors	27
8	GLIDER COMMUNICATIONS	27
	Direct	27
	RF modem (900 MHz Nominal 1 Watt configurable to .05 Watts)	27
	ARGOS (401 MHz ~1 watt)	27
	Iridium (~1600 MHz 1.1 watt)	28
	RUDICS	28
9	BALLASTING	28
	Calculating Weight	29
	Tank to Target Water	29
	External to Internal Weight	29
	Adjusting Weight	29
10	SOFTWARE ARCHITECTURE	30
	Picodos30	
	CONFIG Folder	31
	BIN Folder	31
	LOG Folder	31
	MISSION Folder	32
	SENTLOGS Folder	32

AUTOEXEC.BAT	32
GliderDos	32
Masterdata	33
Sensor Commands	33
Device Commands	33
10.1 Science Computer	35
BIN Folder	36
CONFIG Folder	36
AUTOEXEC.BAT	36
11 SCIENCE DATA RATE COOKBOOK	37
12 FLIGHT DATA RETRIEVAL	41
13 DOCKSERVER DATA VISUALIZER	43
14 SIMULATION	43
15 MISSIONS	47
MI Files47	
MA Files	48
Running Missions	48
a. Operating Systems	49
b. Code Design	49
c. Control: Stacks, States & Abort Sequences	50
d. General Control Structure	51
e. Abort Codes	52
f. Sample Mission and Comments	53
APPENDIX – GLIDER SOFTWARE WEB SITE	62
APPENDIX – WIRING DIAGRAM	63
APPENDIX – COMPASS	64

APPENDIX –Freewave Configuration	65
APPENDIX Iridium Service and SIM CARD	68
APPENDIX ARGOS Satellite Service and ID	68
APPENDIX ARGOS DATA FORMAT	70
APPENDIX – J25 TO DB9 TO DB25 WIRING	75
APPENDIX HOW TO DETERMINE MISSION LONGEVITY	76
APPENDIX COMMONLY PURCHASED SPARE PARTS	77
APPENDIX Ancillary Glider Equipment:	78
APPENDIX – BALLASTING AND H-MOMENT ADJUSTMENTS WORKSHEET	79
APPENDIX – How to edit a proglet.dat file:	85
APPENDIX – DBD_FILE_FORMAT	86
APPENDIX –Legacy Gliderview.exe	97
APPENDIX –Slocum Glider Do’s and Don’ts	100
APPENDIX – Storage conditions	101
APPENDIX – Returning equipment to factory	101
APPENDIX- Slocum Glider Operator Guide “quick reference”	101

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.

An innovative Autonomous Underwater Vehicle, the Slocum glider (AUVG), has two primary designs, the 200 meter coastal glider and 1000 meter glider. Each is specifically designed to work from 4 to 200 meters for the 200 meter and 40 to 1000 meters for the 1000 meter to maximize littoral or deep ocean capabilities. A third design in development is the long range Thermal glider. These platforms are a uniquely mobile network component capable of moving to specific locations and depths, occupying controlled spatial and temporal grids. Driven in a saw tooth vertical profile by variable buoyancy, the glider moves both 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 the regional or larger scale. The Slocum 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 bold and purple throughout this document.

Example:

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 enquire about access to these protected documents.

SPECIFICATIONS

Slocum 30/100/200 meter glider specifications:

Weight in air:	~52 Kg
Weight in water:	Neutrally buoyant
Hull Diameter:	21.3 cm / 8 3/8 Inch
Width including Wings	100.3 cm / 39 1/2 Inch
Vehicle Length:	1.5 meters
Depth Range:	4 - 200 meters (motors geared to depth rating)
Speed, projected:	0.4 m/sec horizontal
Energy:	Alkaline Batteries
Endurance:	Dependent on measurement and communication, type. 30 days
Range:	1500 km
Navigation:	GPS, internal dead reckoning, altimeter
Sensor Package:	Conductivity, Temperature, Depth,
Communications:	RF modem, Iridium satellite, ARGOS

Slocum 1000 meter glider specifications:

Weight in air:	~56 Kg
Weight in water:	Neutrally buoyant
Hull Diameter:	22 cm / 8 2/3 Inch
Width including Wings	100.3 cm / 39 1/2 Inch
Vehicle Length:	1.5 meters
Depth Range:	40 - 1000 meters
Speed, projected:	0.4 m/sec horizontal (Scalable??)
Energy:	Alkaline Batteries
Endurance:	Mission dependant
Range:	Mission dependant
Navigation:	GPS, internal dead reckoning, altimeter
Sensor Package:	Conductivity, Temperature, Depth,
Communications:	RF modem, Iridium satellite, ARGOS

0 Vehicle Operation Theory

The principle advantages of Autonomous Under water Vehicle Gliders (AUVGs) are:

- 1) Very suitable for long-range and endurance, if low to moderate speed is acceptable.
- 2) The saw tooth profile is optimal for both vertical and horizontal observations in the water column.
- 3) Regular surfacing is excellent for capturing GPS and two-way communication, no other navigational aids are required and the system is very portable.

0.1 Forward Propulsion

Gliders are unique in the AUV world, in that varying vehicle buoyancy creates the 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 (Fig 1 - representative of a 200 meter glider). Gliders require no propeller and operate in a vertical saw tooth trajectory.

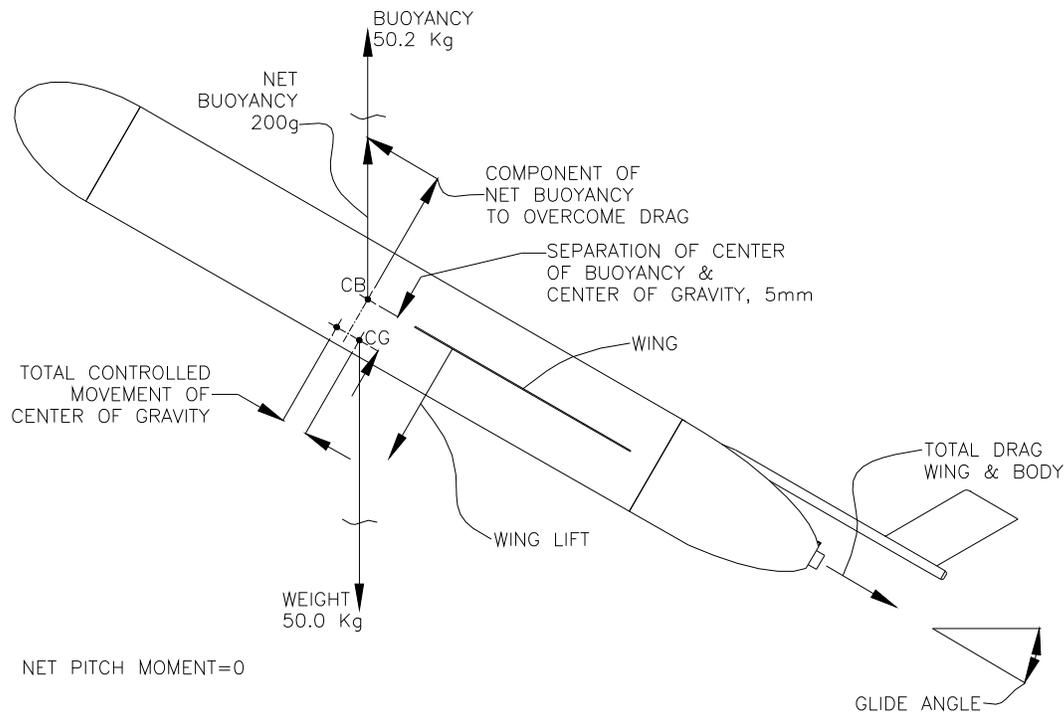


Fig 1. Force balance diagram of forces acting on Glider, angle of attack not included.

0.2 Navigation and Flight

The Slocum Battery Glider dead reckons to waypoints, inflecting at set depths and altitudes based on a text mission file. As set by the mission, the Glider periodically surfaces to communicate data and instructions and to obtain a GPS fix for location. Any difference in dead reckoning and position is attributed to current and that knowledge is used on the subsequent mission segment as a set and drift calculation.

1 Vehicle Description

1.1 Architecture

The Slocum Battery Glider is comprised of three main separate hull sections in addition to two wet sections located fore and aft. The 200 meter glider cylindrical hull sections are 21.3 cm OD 6061 T6 aluminum alloy chosen for simplicity, economy, and expandability. 1000 meter hull sections are 22 cm OD manufactured from composite material or 6061 T6 aluminum. The nose end cap is a machined pressure resistant elliptical shape, and the tail cap a truncated cone. Composite wings are swept at 45 degrees and are easily replaced using a quick release click in, click out system. Take care while removing and installing wings as they are not buoyant and will sink if dropped.

Nose Dome

Gliders manufactured before 2008 have a penetrator thru the front end-cap to the wet section that houses a 10 kHz transducer for Pinger. In addition, the nose dome has a hole on the centerline for large bore movement of water as is created by the ballast pump assembly. Although not of substantial volume and desire to keep reflectors away from the transducer, external trim weights can be added inside the nose dome for ballast trimming.

Forward Hull Section

This section houses the ballast pump assembly, pitch vernier mechanism, (altimeter electronics - was removed from the design in 2007), batteries, and provisions for ballast weights. Internal wiring connectors are mounted on the pump endplate. The large battery pack also serves as the mass moved by the pitch control. When installed the Pinger board is housed within the Forward hull section Bay.

Payload Bay Mid Hull Section

The payload bay is 8 3/8" diameter and 12" long with a nominal capacity of 3 to 4 kg. The payload was designed for easily removal and replacement for calibration needs or sensor type changes, allowing for great ease and flexibility to the user. It consists of two rings and a hull section. The front ring is typically ported for the CTD sensor assembly. To accommodate this section, a software interface exists in the main glider computer to allow a payload or science bay computer to be installed which can control the sensor packages and collect and store data. Any control system is applicable e.g. embedded processors, Persistor, PC104, etc. There are also provisions for ballast weight attachment points. With the exception of the wire harness and the tie rod that must run through the bay for connection from the aft to the forward section, this

volume is set aside for more energy, science or other payloads. H moment adjustments are made by moving weight, high or low, in this section of the vehicle (See Appendix L).

Aft Hull Section

This section houses the strong back chassis that ties the Glider together. On the bottom of the strong back chassis are; the ARGOS transmitter (moved to the lower tray in 2006), a catalyst, and the air pump system. In addition, the battery and internal wiring connector are located on the upper side of the strong back. An upper electronics chassis holds the Vehicle Controller, Hardware Interface Board, and the Attitude Sensor. GPS, Iridium, and RF modem engines are located on the lower electronics chassis tray. The Micron pressure transducer is ported thru the aft end cap and positioned remotely. The aft battery is located under the strong back and can be manually rotated for static roll offsets.

Aft Tail Cone

A faired wet area that houses the air bladder, steering assembly, burn wire, jettison weight, power umbilical, and has provisions for external trim weight and wet sensors. Protruding from the aft end cap through the tail cone is the antenna fin support. This boom is a pressure proof conduit for the antenna leads. The antenna fin is socketed into the support. Below the support is a protected conduit for the Steering Motor Linkage.

Wings

In all operations, particularly coastal work, there is a risk of entraining weed or debris on the wings or tail causing major degradation in gliding performance and for littoral gliders a sweep angle of 45 degrees or more is recommended. Horizontal tail planes are not required, pitch stability is provided by the wings which are mounted aft of the center of buoyancy. In the low Reynolds number regime in which the glider operates (approximately 30,000) their un-cambered (“razor blade”) wings are very suitable. The original glider design required a screw driver to secure the wings in place. The current design “clicks” into place. A quick release is located in the wing rail to the aft.

1.2 Specific Components

Ballast pump assembly 200 meter

A single-stroke piston design, using a 90-watt motor and a rolling diaphragm seal, moves 504 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 glider 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 is desired to pull 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. **It is important to note that the displacement pump should not be run without either external pressure or internal vacuum on the rolling diaphragm.**

Restated, the pump should be installed in the hull section and a vacuum drawn to 6inHg lower than external atmosphere. This ensures 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	Speed 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

Ballast pump assembly 1000 meter

The 1000 meter ballast pump is a rotary displacement design which moves oil from an internal reservoir to external bladder changing the vehicles buoyancy. While you will not damage the pump by adjusting the pump without a vacuum the 1000 meter pump will not properly draw the oil internally without a vacuum, so like the 200 meter glider it is advisable to only move the pump while the vehicle is under vacuum. A rotary valve is used to control flow of oil from bladder to reservoir.

Ratio	Pmax @ 7 amp	Rated Pressure	Speed no load
N/A	1240 dbar	1000 dbar	5 cc/sec

Pitch Vernier

Provided that the h moment is 6mm +/- 1, 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 8.4 Kg (10kg battery) battery pack fore or aft as a vernier. **The default pitch for descent and climb is 26 degrees.** The battery pack is put full forward during surfacing to better raise the tail out of the water for communications.

Altimeter

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

CTD

A typical sensor package contained in the payload bay on the glider is a Sea Bird non-pumped, low drag conductivity, temperature, and depth package. An appendage to the side of the payload bay, 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. Note that the CTD is not used for flight as the glider has an independent pressure sensor for dynamic flight control.

ARGOS Satellite Transmitter (PTT)

The Seimac X-Cat PTT or legacy Smartcat is used for recovery situations transmitting last known GPS positions when available. Service Argos also provides periodic surface location accurate to ~100 meters. See Appendix ARGOS Data

Catalyst

A catalyst is used to recombine Hydrogen and Oxygen into H₂O to reduce the risk of explosion. The reaction is exothermic and the catalyst may become hot. This item does not need periodic replacement. See the disclaimer at the beginning of this document.

Air Pump System

An air bladder in the flooded tail cone is used to provide additional buoyancy on the surface for bettering communications. It is inflated, using air from the hull interior, providing 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 inward via a latching valve for descent.

Vehicle Controller

[A Persistor CF1, based on a Motorola 68338 processor](#) is used to control the functions of the Glider. This board has low current consumption capability and supports the use of Compact Flash cards and miniature hard drives enabling large amounts of data to be stored. 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. 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 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, in memory, any number of pre-written missions (text files) that can be called or a new mission can be created, downloaded to the Glider via the RF or Iridium modem and run. Mission changes might include different inflect depths, new GPS waypoints, or turning a behavior on or off such as current correction.

Hardware Interface Board

The Persistor is mated to this driver board that interfaces to all of the sensors, communications, and drive mechanisms. See Appendix C Schematic and Wiring Diagram. The board runs on a nominal 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 hrs in the factory) is reset (**COP_tickled**) every time there is a GPS fix or a keystroke while in Glider Dos. Both of these situations indicate that the Glider is safely on the surface. If the timer elapses, however, the following items will come alive: Air Pump, ARGOS PTT, Pinger (if available), and the Burn Wire for the Jettison Weight. The 10 kHz Pinger (if available) will change to an 8 second duty cycle and at ~4.2ma (10ms/8 sec rate x 50watts/15v) will emit sound on a single 10 C-cell battery pack for ~60 days.

Attitude Sensor

The Precision Navigation compass and attitude sensor provides the bearing, pitch, and roll indications of the Glider. These inputs are used for dead reckoning the vehicle while under water. Recalibrating the compass, depending on the magnetic anomalies of the usage area, may at times be necessary. See Appendix D Compass Calibration.

GPS

The GPS is on during surfacing and it is used to locate the Glider's position and update the glider time if necessary. The output used is the RMC NMEA 0183 string, every 5 seconds.

Iridium Satellite Telemetry

The Iridium bi-directional satellite modem is on the lower electronics tray with a Low Noise Amplifying (LNA) switching board for the antenna, which is shared with the GPS.

RF modem Telemetry

Freewave 900 MHz radio modem is used for the local high-speed communications link to the Glider. It is connected to the console on the Persistor, permitting code load changes. See Appendix E: Freewave Configuration.

Pinger (discontinued in 2007)

The pinger is controlled by software in normal operation and by hardware during a hardware abort in which the Persistor has failed. The pinger will output various ping structures depending on the message being translated. The pinger transmits at 10 KHz, and every eight seconds it transmits the depth of the glider below the surface. If the glider is at the surface the pinger transmits, then transmits again 100mS later. If the glider is below the surface, the second transmission will be one second after the first, when the vehicle is at **u_pinger_max_depth**. The default value for **u_pinger_max_depth** is 100 meters. If this value is changed, the second ping will transmit when the vehicle dives the new value. ({ex. **put u_pinger_max_depth 25**}) This will set the Pinger maximum depth to 25 meters meaning that at 25 meters the pings will be spaced one second apart.) The following are transmitted once at the beginning of each task:

Vehicle starting to climb	5 Pings
Vehicle starting to dive	10 Pings
Vehicle at the surface	15 Pings
Vehicle software aborts	20 Pings
Vehicle hardware aborts	20 Pings

Science/Payload Computer Switch Board

A hardware relay on the glider control board is used to switch the RF modem communications to the science/payload computer to allow direct access through the software application consci.run on the Science bay Persistor. A disconnect of carrier detect for three seconds will revert the RF communications back to the Glider Controller Persistor. In the field, disconnecting power to the host side RF modem for three seconds will accomplish this.

Pressure Transducer

Micron 300 and 2000 PSIA strain gage transducers for 200 meter and 1000 meter gliders respectively are used for vehicle control and dead reckoning. Ported through the aft cap, the transducer is isolated by oil filled stainless tubing to prevent thermal shock. Changed to PEEK tubing and fittings in 2008.

Leak detect

Each glider is equipped with two leak detect sensor boards, the aft one is located on the bottom of the aft cap. The forward leak detect sensor is attached to the bottom of the front cap in the 200 meter glider and in the 1000 meter glider the sensor is attached to the front of the payload bay. The sensors will 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.

Air Bladder

A 1400cc bladder provides buoyancy and stability while the Glider is surfaced, lifting the antenna support out of the water. The bladder is filled via the Air Pump System. Although the bladder is rugged, care should be taken to have the Aft Tail Cowling in place when the bladder is filling. The bladder is then supported as it inflates until shut off by the pressure switch. Likewise, when removing the Aft Tail Cowling it is important to deflate the Air Bladder, as it will be hard up against the Cowling. See Opening Procedures for more details.

Burn Wire

The glider is equipped with an emergency abort system, a replaceable/rebuildable battery activated corrosive link that after approximately 15 minutes in salt water and 4 hours in fresh water will release a spring ejected Jettison Weight. The wire is 20 AWG Inconel held in a delrin bushing and mated and sealed to a single pin Mecca connector. 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. See Maintenance for more details.

Jettison Weight

A lead mass (470g) attached to Burn Wire Assembly by 300 lb test monofilament. When the Burn Wire is electrically corroded the Jettison Weight is forcibly ejected by a spring forcing the Glider to surface (within limits of mass lost).

Power Umbilical

An Impulse cable is used to switch or supply power to the Glider. When the (red band) Dummy Plug is inserted or the connector end is empty there is no power applied to the vehicle. This is done so that any person can be instructed to easily remove power from the system without special tooling. Further, for safety reasons no internal spark is generated as could be with an internal switch. To power the Glider on either use the provided external power cable or insert the (green band) shorting plug. The Umbilical is accessible external to the Glider Aft Tail Cone. See Appendix Wiring Diagram and Section Maintenance for Plug care. Voltage should not exceed 16 volts.

Digifin

A molded urethane self-calibrating tail fin is moved as a controlled plane acting as a rudder. Unlike its predecessor described below, it can be used to handle and manipulate the glider. It also contains the glider's antennas: ARGOS 401 MHz, Freewave RF modem 900 MHz, and combined GPS 1575 MHz and Iridium 1626 MHz

Antenna Fin/ Steering Assembly replaced by Digifin in 2008

A tail fin is moved as a controlled plane acting as a rudder. The steering motor and rotary potentiometer are located external to the pressure hull and are oil filled and pressure compensated. The maximum tail fin angle spans ± 35 degrees. The hinge is synthetic as the fixed portion of the fin contains antenna structures.

The tail fin houses three antennas: ARGOS 401 MHz, Freewave RF modem 900 MHz, and a patch with combined GPS 1575 MHz and Iridium 1626 MHz. The fin is bonded to a socket that is installed into the antenna support with dual radial o-ring seals. This provides a passage for the antenna cables and allows for easy replacement/upgrade of the Antenna Tail Fin module. The antenna should not be handled during launch and recovery. The tail boom attached to the fin and antenna is the desired location for manually manipulated the position of the glider during launch and recovery. Mention digifin, which can be handled during launch and recovery

Sacrificial anode

The outside of the aft end cap is fitted with a zinc sacrificial anode. This anode must have continuity to ground within the glider. The anode will need to periodically be replaced. Beware any scratches to the anodizing of the glider aluminum parts can be corrosion points. Scratches, which reveal bare aluminum with continuity, should be touched up with paint or in an emergency nail polish is effective. It is advisable to rinse the glider with fresh water each time it is exposed to salt water. The anode should be checked for continuity to ground on regular basis, to ensure proper protection against corrosion.

Batteries

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

<u>Location</u>	<u># of packs</u>
Pitch Battery	12
Aft Battery	10 - 12
Nose Batteries	1- 2

For power management typically all of the packs except one of the aft battery packs are tied into battery main. The one emergency pack is tied to Battery Backup and runs the Main Controller boards and performs the following functions only: Abort Timer, Burn Wire, ARGOS, and pinger (if available) in the event of Main power loss.

Desiccant

If possible the glider should be opened and sealed in a controlled, dry environment. A desiccant pack or several 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 atmospheric moisture from saturating it.

2 Opening Procedure

Only trained and qualified personnel should operate and maintain the glider.

Teledyne Webb Research conducts regular training sessions several times a year. Glider users should attend a training session and understand basic glider concepts and terminology. Contact glidersupport@webbresearch.com for information regarding training sessions. Company policy is to fully support only properly trained individuals and groups.

The section on opening and closing below will reference communications with the glider, please see the separate documents GMC or Dock Server User Guide and section 6 of this manual as a reference.

Teledyne Webb Research recommends use of Dockserver as your sole source of communication to the glider. However a properly configured Freewave modem connected to a terminal program such as Procomm or Hyperterminal can also be used for Glider communication if a Dockserver is unavailable.

Work on the glider is best done on a glider cart. The glider will lose its rigidity when taken apart and the sections will need to be supported.

2.1 Glider Hull Sections

Note: glider commands and definitions can be accessed by typing **?**. All glider commands in this manual will be referenced in bold and purple.

The glider should be powered down while performing maintenance.

- The Air Bag must be deflated in order to remove the Aft Tail Cowling. To deflate an inflated bladder from GliderDos, type the following command **put c_air_pump 0** and press enter. This will deflate the airbag. Then type **exit** and enter. This will shut down the glider.
- Turn power to the glider off by removing green plug or External Power Cable and replacing red plug.
- Remove the two 10-32 socket head cap screws (SHCS) and washers that hold the Aft Tail Cowling in place, using a 5/32 hex driver.
- Slide the cowling back, slightly separating the cowling to allow for the antenna tail fin.
- Remove the 7/16 MS vent/access plug on the starboard side of the Aft End cap with a 3/16 hex driver.

Note: If an internal vacuum is present, air will be heard rushing in. Properly stored with an internal vacuum air should not rush out, this may be a sign of released hydrogen gas – caution should be taken. If there is any concern, prior to opening the instrument, the internal pressure can be read from a terminal. To read the internal vacuum type the following command from

GliderDos **Report ++ m_ vacuum**. This command reports the vacuum scan cycle. Typing **Report clearall** Stops vacuum from reporting.

- Loosen the 10-32 SHCS on the Wing Rails with a 5/32 hex driver. This prevents the hull paint from being scratched, as the Wing Rails overlap hull sections.
- Insert the long handled 3/16 hex driver into the access port and engage the captured SHCS Tension Rod. Note: Take care to not damage the MS seal area or the threads with the side of the driver.
- Unscrew until the SHCS is disengaged from the Tie Rod Plate.

Note: Full disassembly will be described here, but one can also partially disassemble the Glider to access a particular module.

- Using the hull separation tool separate the center Payload Bay from the forward and aft hull sections.
- Unscrew the 63 pin CPC connector and unplug the battery from the forward end of the aft tray.
- Carefully withdraw the Aft End cap and Chassis from the Aft Hull. Set aside on a support stand to prevent tipping and/or rolling.
- Note or mark the rotation of the aft battery pack. To free the Aft Battery Pack from the Payload bay remove the 2 1/4 x 20 SHCS screws and washers, which hold the battery plate in place, using a 3/16" hex driver.
- Separate hull sections Unscrew the 28 pin CPC, unplug the battery, and disconnect the BNC (if applicable) connectors from the forward ballast pump assembly.
- Each section is now independent.

2.2 Center Payload Bay

- Remove the two aluminum hex bolts from the aft end ring of the center payload bay with a 1/2 inch wrench.
- Deep gliders: use a 3/32 hex wrench into battery standoff.
- Remove the guard plate from the payload bay by removing two Pan Head Phillips screws.
- Using the hull separation tool, separate the aft ring from the payload bay hull. Use caution when removing the ring as the weight bar may still be attached.
- Unplug the science sensor connectors.
- Carefully remove the center hull section from the front ring and set aside.
- This will expose the interior of the center payload bay.

2.3 Ballast Pump Assembly

- Looking into the open end of the forward hull section, there are two SHCS visible on the top of the pitch battery pack at mid length. Using a 5/32 long handled hex driver loosen both of these captured bolts.
- Withdraw the Pitch Battery Pack from the forward hull section, set aside.
- Place the forward hull section in a vertical stand (Nose down). Using the hull separation tool carefully remove the forward hull section from the ballast pump.

2.4 Nose Cone and Altimeter

- Remove four SHCS from the Nose Cone with a 7/64 hex driver.
- Remove Nose Cone.
- Remove four SHCS from the Altimeter with 9/64 hex driver
- Unscrew Altimeter pigtail from bulk head.
- Older gliders have hardwired altimeters, DO NOT REMOVE

3 Closing Procedure

Care must be taken to inspect and lubricate the O-rings and clean the O-ring surfaces. See Maintenance. When tightening the nuts and bolts do not over tighten and apply proper torque values where applicable.

3.1 Nose Cone

- Replace Altimeter and Nose Cone
- Replace four SHCS from the Nose Cone.

3.2 Ballast pump assembly

- Place the Ballast pump assembly in a vertical stand (Nose down).
- Paying attention to the centerline marks, Slide the forward hull over the Ballast pump assembly and seat it on the forward end cap.
- Lay the piston displacement pump and forward hull section horizontally on the cart.
- Check that the Forward End cap is seated square in the Forward Hull Section.
- Slide the Pitch Battery in from the aft end of the Forward Hull Section.
- Looking into the open end of the forward hull section, there are two SHCS visible on the top of the pitch battery pack at mid length.
- Using a 5/32 long handled hex driver engage and tighten both of these captured bolts.

3.3 Center Payload Bay

- Paying attention to the centerline marks, slide center hull Section over the payload bay and seat squarely on the front ring. Use caution to prevent damage to sciences sensors.
- Reconnect science sensor wiring.
- Install the aft end ring and weight bar assembly(s) over the payload bay and seat squarely on the hull section.
- Replace aluminum chassis bolts from the aft end ring using a 1/2 inch wrench. Do not over tighten.
- Deep gliders: use 3/32 hex wrench into battery standoffs.
- Replace the guard plate

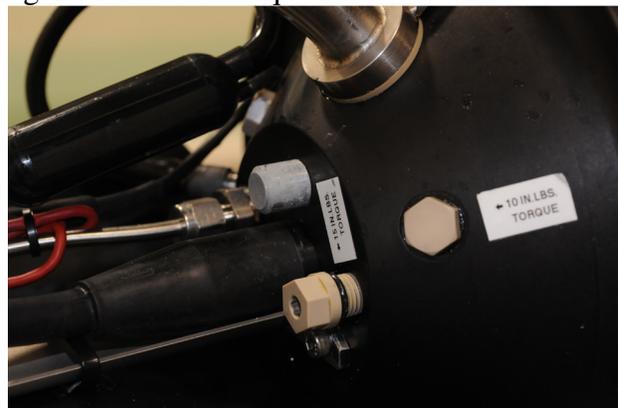
3.4 Glider Hull Sections

- Ensure that power is not applied to the vehicle and that the red stop plug is installed
- Place the three hull sections on the cart in the appropriate fore and aft position keeping the top centerline marks up and in line.

- At the forward end of the center payload reconnect the battery connector, the BNC transducer connector (if applicable), and the CPC connector using the alignment marks. Note: Take care in engaging connectors, as the pins are delicate.
- Slide the Center Payload Bay and the Forward Hull together taking care not to pinch any wires and to ensure free movement of the battery connector wire, as it must move with pitch adjustments.
- Position aft hull so that the centerline marks are matched with the Center Payload Bay leaving an approximate 3-inch gap between the aft and center hull sections.
- Install the aft battery pack and slide up to aluminum chassis bolts of the Center payload bay.
- Slide the aft battery pack up to the aluminum chassis bolts of the center payload bay. Realigning to noted rotational position of the aft battery pack, replace and tighten the two 1/4x20 SHCS with a 3/16" hex driver. Note: The hull sections must be aligned parallel and at the same height to allow the hulls sections to fit together.
- Ensure that all of the connectors on the aft chassis are connected and seated properly.
- Slide the Aft End cap and Chassis into the Aft Hull.
- Take care, that the tension rod runs through the tie rod guide tube in the center payload bay. This aligns the rod and protects payload components.
- Reconnect the aft battery connector and the CPC connector and wire bundle. Take care in engaging connectors, as the pins are delicate.
- Slide the center and forward hull sections together until the tension rod engages with the Tie rod plate in forward hull section.
- Using the 3/16-inch hex t-handled torque wrench tighten the Tension Rod to 15 in/lbs. Use caution that the O-rings are seating properly and that no wires are being pinched. Take care to not damage the MS seal area or the threads with the side of the driver.
- The hull sections should be tight and square to the end caps. The rings and hull sections with the centerline marks should be in alignment.
- Refit wing rails with SHCS using a 5/32 hex driver.
- Use a vacuum pump with an evacuation tool and place the 7/16 MS Plug on the hex driver
- Seal the evacuation tool over the aft end cap evacuation port shown by the red arrow
- Evacuate the glider to 6 inches of mercury for 200 meter glider and 7 inches of mercury for a 1000 meter glider and screw in the MS plug using the 15 inch lb torque handle.



Vacuum gauge and tool



M/S plug not seated

-.

.



Vacuum drawn and M/S plug seated

- Power the Glider with an external power cable (15 volts DC) or insert the green Power Plug.
- Use Dockserver or a terminal emulator connection to a Freewave modem (115,200K baud, no parity, 8 data bits, and 1 stop bit) to establish communication with the glider.
- Press **control C** take control of the glider in GliderDos after the glider responds with:
- *SEQUENCE: About to run initial.mi on try 0*
 - o *You have 120 seconds to type a control-C to terminate the sequence.*
 - o *The control-P character immediately starts the mission*
 - o *All other characters are ignored*
- Type the commands:
 - **lab_mode on**
 - **callback 30**
 - **ballast** –wait fo motors to become idle.
 - **Report ++ m_vacuum**
 - **Note**
One + symbol will display any time the measurement changes.
Two ++ symbols will display every time it is measured ~every 2 seconds.
The above is true of any measured sensor.

```

peter
pongo
potame
pytheas
sim012
sim015
sim_026
simbond
trieste_1
unit_087
unit_085
unit_097
unit_098
unit_103
unit_104
171.36 41 sensor: m_vacuum = 6.5 inHg
175.57 42 sensor: m_vacuum = 6.5 inHg
179.54 43 sensor: m_vacuum = 6.5 inHg
183.74 44 sensor: m_vacuum = 6.5 inHg
187.73 45 sensor: m_vacuum = 6.5 inHg
191.95 46 sensor: m_vacuum = 6.5 inHg
195.92 47 sensor: m_vacuum = 6.5 inHg
199.97 48 sensor: m_vacuum = 6.5 inHg
Ctrl-C
lab_mode on
ballast
report ++ m_vacuum

```

Glider Terminal screenshot

- Adjust vacuum accordingly by applying more vacuum or slightly unscrewing the 7/16 MS plug allowing air back into the hull until the vacuum reads 6 inches of mercury \pm .3 for a 200 meter glider and 7 inches of mercury for 1000 meter glider.
- Ensure MS plug is tight using the 15 inch lb torque handle.
- Note: This is done at room temperature with the piston displacement pump in ballast position. The vacuum will subsequently change with piston movement, air bladder inflation, and hull temperature – all items that change the internal volume of air or its pressure. The vacuum serves many purposes: a leak test, providing seating of the O-rings, force holding the Glider together, a differential pressure on the displacement piston rolling diaphragm to eliminate bunching, and the air return method for deflating the air bag

To terminate the vacuum report:

- Type the command **report clearall** to stop reporting.
- Type the command **exit**, after a few seconds the glider will report:

YOU MAY NOW REMOVE POWER FROM THE GLIDER

- Remove power from the glider by removing green plug and install the red stop plug.
- Slide aft cowling around the antenna fin and engage on the aft end cap. Replace the two 10-32 SHCS and washers that hold the aft tail cowling in place using a 5/32 hex driver – do not over tighten.

4 Pre mission testing

These procedures should be followed for qualification of a glider prior to delivery, or with new or modified software.

On the beach, boat or bench:

Power on glider and enter glider dos by typing **ctrl-c**

Type **put c_air_pump 0** to stop the air pump from running.

Type **callback 30** to hang up the iridium phone.

Test gps by typing **put c_gps_on 3**. This will put gps communication 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**

Test motors by typing **lab_mode on** and then **wiggle on** and run for 3-5 minutes to check for any device errors or other abnormalities. Type **wiggle off** to stop wiggling.

If no errors are found, type **lab_mode off** to return to the GliderDos prompt.

(Always ensure that the glider is not in Pico or Lab Mode before deploying it in the water)

Type **run status.mi** and confirm that all sensors are being read. Mission should end “mission completed normally”.

Load glider into the boat and head out toward the first waypoint or deployment location. Note: These tests need to be done outside as the gliders needs a clear view of the sky to get a GPS fix and to make iridium communications.

In the water:

If possible it is advisable to attach a line with flotation to the glider before putting it in the water. However if you have great confidence in the ballasting and are an experienced user you may proceed without a floatation device.

Once the glider is in the water type run **status.mi** once again.

Run ini0.mi

Does 1 yos, dive to 3m
Fixed pitch and fin

Run ini1.mi

Does 3 yos, dive to 5m, climb to 3m
Heading North, pitch at += 20 degrees

Run ini2.mi

Goes to a waypoint 100m south of dive point
Does yos, dive to 5m, climb to 3m

Run ini3.mi

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

If you are performing this mission on a line with floatation please ensure line length is sufficient or modify yo depth.

When the glider returns to the surface evaluate if it is ok to proceed by examining data.

If you have not removed the line from the glider, do it now.

From the GliderDos prompt type **exit reset**. This will force a re-initialization of all of the sensor values. It is advisable to do an exit reset after removing the buoy but not necessary.

When the glider re-boots type a **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.

Type run **glmipc.mi** to begin the glmipc mission or run desired mission.

5 Deployment and Recovery

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

Deployment conditions and craft will vary. The glider 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 by tail tube not the antenna unless the glider is fitted with the Digifin. A glider with a Digifin installed can be manipulated by the palm sized, horizontal antenna area.

During recovery the pick point or cart can be useful tools to manipulate and provide support while moving the glider onboard. A hook or lasso on a pole has also been used to manipulate the glider while in the water. Remember to not use the antenna as a handle unless the glider is fitted with a Digifin.

Photographs of deployment and recovery by cart can be found in the Appendix Operators Guide.

6 Emergency Recovery

Contact glidersupport@webbresearch.com for specifics but some or all of the following may need to be done in an emergency recovery scenario:

Callback time can be increased significantly by changing in the glider Autoexec.mi:
sensor: u_iridium_max_time_til_callback(sec) 1800.0 # 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 glider dos from 900 to 3600. This will increase how often the glider will cycle into a mission and try to call in to save energy while stuck at surface.

Consider running special scripts to conserve energy

A pilot might change to a **callback 30** script

Contact service Argos and turn on Argos ALP - all location processing.

Determine best-known position with available data.

7 Maintenance

General

Rinse glider with fresh water after exposure to salt water. Minor scratches to paint and anodizing should be touched up with automotive paint or nail polish.

O-rings

O-rings should be inspected for cleanliness, nicks, and slices. O-ring surfaces should also be inspected for scratches, dents, and cleanliness. [Parker fibrous o-lube 884-4](#) Petroleum Naphthenic Oil and Barium Soap is recommended.

Pressure transducer

Rinse well with fresh water after each salt-water deployment

Burn Wire Assembly

If in the event a Burn Wire has been used and must be replaced:

- The Air Bag must be deflated in order to easily remove the Aft Tail Cowling. In GliderDos **put c_air_pump 0**. See Glider Communication & Sensor Command.
- Remove the two 10-32 SHCS and washers that hold the Aft Tail Cowling in place with a 5/32 hex driver.
- Slide the cowling back and around Antenna tail fin.
- Disconnect the single pin Mecca connector.
- From the single pin male Mecca connector side, remove the burn wire bushing in the jettison weight tube.
- Do not discard the used burn wire assembly, as they can be rebuilt at the factory.
- Remove the e-ring from the new burn wire assembly complete with jettison weight attached.
- Feed the single pin Mecca connector through the aft end of the Jettison weight tube out the hole on the forward end.
- With one hand push the Jettison Weight into the tube compressing the jettison weight spring.
- At the same time feed the Mecca wire through the hole until the burn wire bushing appears.
- With the face of the burn wire bushing beyond the edge of the hole, slide it slightly sideways so that it is resting on the crossbar and does not fall back through the hole.
- Replace e-ring on the burn wire bushing seating it fully.
- Allow the burn wire bushing to fit back through the hole. It will be stopped by the e-ring on the face of the crossbar.
- Reconnect to the single pin Mecca connector. Use O-lube lubrication.
- Replace Aft End Cowling. See Closing Procedure.

Dummy and Green Plug

- Use O-lube lubrication or Silicone Spray.
- Keep Contact Pins clean.

Sensors

Individual sensors may have special needs. See manufacturer recommendations.

8 Glider Communications

The Glider is intended to be used in conjunction with Dockserver, see the GMC/Dock Server Users Guide for information on communication to the glider while using Dock server.

Direct

The glider control Persistor is set to communicate at 115,200, N, 8, 1.

Provided that the RF modem system is working, direct wire communication is not necessary. If needed:

- Open the Glider.
- Remove the aft end cap and chassis from aft hull. (See opening procedure)
- On the glider control board remove connector J25 and replace with a patch cable to computer with terminal emulator running. See Appendix G - Wiring Diagram.
- Disconnect the ballast and air pump connectors and apply power to the Glider. (This is necessary because the air bladder requires vacuum and the aft cowling to provide resistance to switch off the air pump at the proper differential pressure.
- Communication should commence.

RF modem (900 MHz Nominal 1 Watt configurable to .05 Watts)

[A Freewave modem](#) should be paired with the vehicle using Point to Point (with Repeaters if necessary), the correct Ids in Call Book, and matching frequency keys. See Appendix E: Freewave Configuration.

- Connect the Freewave to Dockserver or to computer running terminal emulation.
- With power applied to both the Freewave and the Glider, communication should commence.

Changing Settings on Glider Freewave

- Open the Glider. See Opening Procedure.
- Remove the Aft End cap and Chassis from Aft Hull.
- On the Freewave board, remove connector and replace with a patch cable to computer with terminal emulator running 19,200, N, 8, 1.
- Take the reset line to ground. See Appendix E: Freewave Configuration, configuration and patch cable wiring.
- Change appropriate settings.

ARGOS (401 MHz ~1 watt)

ARGOS messages are nominally transmitted from Glider while powered up on the surface. See Appendix Argos-data-format.txt for documentation of message transmitted.

Iridium (~1600 MHz 1.1 watt)

Iridium is generally used in the absence of the Freewave. The phone number is configured in the Autoexec.mi file. In GliderDos, Iridium always tries to call the given number; however, while running a mission, Iridium will call only if Freewave communication is not available. Similarly, during data transfer, the data will be transferred via Iridium, only if Freewave is not available. The option to force transfer through Iridium while Freewave is connected is also available.

To test Iridium in Picodos, use **talk iridium**. (refer to Picodos section of manual)

- **Control-C** pause **Control-C** to leave talk.
- 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** Is used to answer an Iridium phone call
- **ath** Is used to hang up after terminating an Iridium call. Alternately, a control-C can be use to hang up the phone

Rudics

In 2008 Rudics capabilities were introduced. Refer to section 12 of the GMC Manual:
<ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/gliderMissionControl/Documentation/gmcUserGuide.pdf>

9 Ballasting

The goal of ballasting is to adjust the mass of the Glider to result in the glider being neutrally buoyant and properly trimmed at the surface of the operation site water with the Displacement Pump set to 0 cc and the Pitch Vernier set to 0 inches and the air bladder deflated. Static roll should be set to 0 degrees. The command **ballast** from Lab mode will move these motors to the proper position.

- Close Glider with an internal vacuum of 6 (+/- 0.3) inHg (7 inHg for 1000 meter gliders). See Closing Procedure.
- Power the Glider with either an external power cable (15 volts DC) or insert the Green Plug.
- If not there already, **boot app <>** to get to GliderDos.
- From GliderDos type **callback 30** then **Lab_mode on**
- Type Ballast. This will set the ballast pump to 0, the pitch vernier to 0 and deflate the air bladder
- If external power was used type exit and pull the plug and replace with a Red Dummy Plug when the glider states it is ok to remove power.
- If internal power is being used the Glider may remain on. Keep in mind that if there is no GPS fix or cop_tickle from the keyboard (any keystroke) within 18 hours the Hardware Abort will commence. See Hardware Interface Board.
- Insert the wings into the wing rails. If your tank does not provide you with the space, you may lay stacked Wings on top of the Aft Hull Section aligning the holes of the Wings with the aft holes of the Wing Rails. Hold in place with a wrap of electrical tape around the Hull and Wings.
- If Glider is too light, add external washer to the exterior of the hull in the areas where one can make weight adjustments. See Ballasting, Adjusting Weight. These tend to be at the hull intersections at the Forward and Aft End caps and at the Payload Bay.

- If Glider is too heavy, attach spring scales at the hull intersections at the Forward and Aft End caps to determine balance and overall excess weight.
- Record the results and compare against target surface water. The mass Glider has just been measured to make it neutrally buoyant in the tank, one then needs to compare the tank to the target and add or subtract that calculated weight to derive the final mass of the Glider.
- To achieve the proper neutral pitch the weights may need to be shifted fore or aft internally. The Pitch Vernier will take care of some offset provided that the h moment is ideally set to 6 mm.

Calculating Weight

Tank to Target Water

Density and temperature adjustments from one body of water to another:

Weight to adjust glider = Displacement of glider * ((Thermal coefficient of expansion * (Target water temperature - Tank water temperature) * Target water density) + (Target water density / Tank water density - 1))

Given:

Displacement of glider = Air weight of glider.

Thermal coefficient of expansion of aluminum = 70E-6.

Density in grams/liter.

Weights are in grams.

Temperatures are in Celsius.

External to Internal Weight

In ballasting the vehicle in a tank it is often desirable to lay external weight on the hull to trim the overall weights and moments. These weights displace water and it should be accounted for prior to putting the equivalent weight internal to the vehicle.

Stainless Steel: External air weight * .875 = Internal weight to be added.

Lead: External air weight * .912 = Internal weight to be added.

Adjusting Weight

The capabilities to adjust weight and balance on the Glider are lead weights, lead shot in Ballast Bottles, Batteries, and Payload. Sites include:

- Nose Cone: external washers tie wrapped to the Transducer Cable.
- Forward Hull Section: provisions for two 2 X 5 C-cell Battery Packs, two Ballast Bottles
- Payload mid Hull Section: shifting payload such as science packages, batteries, etc., a top and bottom position for Weight Bars that have tapped holes for positioning of punched lead rounds, and provisions for a Ballast Bottle.
- Aft Hull Section: removal or addition of batteries from the Aft Battery Pack, provisions for a Ballast Bottle.
- Aft Tail Cone: external weight attached to various places in the wet area.

10 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 which allows one to re-play flight dynamics or 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 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.

Hierarchy of software control

PicoDos	Persitor operating system
GliderDos	Glider operating system
masterdata	Defines all “sensors” or maybe better defined as Glider variables (~1400)
longterm.dat	Maintains sensors/variables on power cycle
Autoexec.mi	Defines glider specific variables
.mi files	Mission files defines mission variables
.ma files	Mission acquisition file defines 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 is as follows:

talk gps	Talk gps turns the GPS on and displays the NMEA output; this may also be used to acquire a full almanac. Leaving the GPS on for in excess of 15 minutes will refresh the almanac.
talk att	Legacy: Talk att turns the attitude sensor on and displays the pitch, roll, heading and temperature output; this may also be used to calibrate the compass. See Appendix D. (Does not work with TCM3 Compass)
talk arg	Legacy: Talk arg turns the Argos transmitter on and displays the output. This command only works with the older Smartcat style PTT

talk iridium Talk IRIDIUM turns the Iridium modem on and allows the user to manually make or receive a call.

Structured in Picodos or GliderDos are the following folders and contents:

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

- AUTOEXEC.MI Configuration file for calibration constants and factory settings.
- CONFIG.SCI Specify which sensors are sent to the glider and when.
- 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 Can convert the glider into several versions of a simulator. See Simulation. **(Note this file must be deleted prior to actual flights)**
- ZMEXT.DAT Automatically puts transferred files in the correct directory from any other directory.

BIN Folder

Where Pico executable programs are kept. These are all run in PicoDOS.

- ADTEST.RUN Displays and updates raw voltages for A → D devices. Used in calibration procedures.
- ALLOFF.RUN Is 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 over to the Payload/Science Computer for direct access and code loads. Note: A loss of carrier detect on the Glider side will automatically switch back to Glider Controller.
- SRTEST.RUN Allows switching on/off of select bits. 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** will give a list of parameters.
- UARTTEST.RUN Testing of uart drivers for programming.
- ZR.RUN and ZS.RUN Required for z-modem send and receive transfers.

LOG Folder

Where the mission derived data is stored. Types include:

- .DBD Dinkum Binary Data, all sensors turned on for recording are stored in this type file. See Appendix K
- .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.

MISSION Folder

Missions are stored here as *.mi files. They are text files that when run by the glider determines the behavioral parameters.

SENTLOGS Folder

The storage of .dbd .sbd .mlg files from LOG folders that have been successfully sent.

AUTOEXEC.BAT

Typically this has:

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

GliderDos

The operating shell GliderDos is a superset of Picodos. It is an application that performs most of the Picodos functions and has knowledge of the Glider as explained under Software Architecture. Typing **help** or **?** will list the functions available. Sensors make up all of the variables in the glider these sensors are defined in masterdata. Behaviors then use these values to operate the vehicle. The glider will list all sensors names with the command **list**.

GliderDos is an application that is loaded onto the Persistor (glider.app). Configured correctly it will boot up and call an Autoexec.mi file that has all of the Glider's calibration coefficients. Certain devices are set automatically: Ballast pump assembly full extension, Pitch full forward, Air Pump on, ARGOS on, Freewave on, and GPS on to ensure best possible surface expression. The reasons to go to Picodos are to: load new source code for GliderDos and to work in the file structure without the Device Drivers being called. The glider should never be deployed will 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 following basic control commands

- 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 need to be set in Picodos

Masterdata

Masterdata contains all of the sensors or variable definitions and default values. Masterdata cannot be changed as a text file as the mission files can be. 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/src/code>

prefix meanings:

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_ Do not ever set this. Typically computed at run-time.

s_ simulated state variables

Sensor Commands

Sensors can be changed on the GliderDos command line. Commands are as follows:

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 will allow one to see what devices are installed and in use. During GliderDos operation if a device is installed and receives 2 errors it will be taken out of service. Presently, this number is upped to 20 during a mission for most devices. Errors are primarily that the driven device is not moving, thus it is taken out of service as a protective measure.

use ?	Prints help
use	Lists all devices
use + dev_name	Puts device(s) in service
use - dev_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 command help will list all commands available to the user:

Full help menu and definition

- * see Appendix command examples for examples of this command
- *** not often used by average user

ATTRIB	[+ - RASH] [d:][p][name]	
ballast	BALLAST ? ; for help	
boot	[PICO][PBM][APP]	
callback	<minutes til callback>	
capture	[d:][p]fn [/Dx/B/N/E]	*
CD	Change Directory	
CHKDSK	[d:][p][fn] [/F][/I]	***
CLRDEVERRS	zero device errs	
consci	[-f rflirid] ; console to science	
COPY	source dest [/V]	
CP	<src_path> <dest_path> ; copy a file system branch	
CRC	Compute CRC on memory	
DATE	[mdy[hms[alp]]] /IEUMCP]	
DELLOG	ALL MLG DBD SBD	
DEL	[drv:][pth][name] [/P]	
DEVICES?	print device driver info	
DF	Print disk space used and disk space free	
DIR	[d:][p][fn] [/PWBLV4A:a]	*
DUMP	file[start[,end]]	***
ERASE	[drv:][pth][name] [/P]	***
exit	[-nofin] [poweroff reset picolpbm]	
GET	GET <sensor name>	
HARDWARE?	[-v] ; Hardware config	
HEAP	Report Free Memory	
HELP	Print help for commands	
HIGHDENSITY	HIGHDENSITY ? ; for help	
LAB_MODE	[onloff]	
LIST	;display all sensor names	
loadmission	loads mission file	
logging	onloff ; during GliderDos	
LONGTERM_PUT	LONGTERM_PUT <sensor name> <new value>	
LONGTERM	LONGTERM ? ; for help	
LS	[path] ; list a file system branch	
MBD	MBD ? ; for help	
MKDIR	[drive:][path]	
MV	<src_path> <dest_path> ; copy a file system branch	*
PATH	Show search path	***

PATH	[[d:]path[;...]] [/P]	***
prompt	[text] [/P]	***
PRUNEDISK	Prune expendable files to free space on disk	
PURGELOGS	Deletes sent log files	
PUT	PUT <sensor name> <value>	*
RENAME	[d:][p]oldname newname	*
REPORT	REPORT ? ; for help	
RMDIR	[drive:][path]	
RM	<path> ; delete a file system branch	***
run	[mission_file] ; runs it	
SBD	SBD ? ; ? for help	
SEND	[-f={rf}] {irid} [-num=<n>] [-t=<s>] [filespec ...]	*
sequence	SEQUENCE ? ; do this for help	
SETDEVLIMIT	devicename os w/s w/m	*
SETNUMWARN	[X] ; set max dev warnings to X	
SET	[var=[str]] [/SLFE?]	***
SIMUL?	print desc of what is simulated	
SRF_DISPLAY	SRF_DISPLAY ? ; for help	
sync_time	[offset] ; Syncs system time with gps time	
tcm3	TCM3 ? ; for help	
TIME	[hh:mm:ss [alp]] [/M/C]	
tvalve	[upchargeldown][backward]	***
TYPE	[drv:][pth][name]	
USE	USE ? ; do this for help	
VER	Firmware versions	
WHERE	prints lat/lon	
whoru	Vehicle Name:	
WHY?	[abort#] ; Tells the reason for an abort	
wiggle	[onloff] [fraction] ;moves motor	
ZERO_OCEAN_PRESSURE	re-calibrate(zero) ocean pressure sensor	
ZR	Zmodem Rec: zr ? for help	
ZS	Zmodem Send: zs ? for help	

10.1 Science Computer

From Picodos, type: **consci** or from GliderDos type **c** to transfer to communicating with the science Persistor.

Structured in Picodos of the science Persistor are the following folders and contents:

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 "proplet" for each device connected to the science computer. Some of the devices are as follows:

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 Photosynthetically Active Radiation
whgpbm	WHOI Glider Bathy-Photometer
hs2	HobiLab HydroScat2 Spectral Backscattering Sensor
bam	Benthos Acoustic Modem

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

BIN Folder

Where Pico executable programs are kept. These are all run in Picodos.

- AMCONNECT. 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 and ZS. RUN Required for z-modem send and receive transfers.

CONFIG Folder

- APPCMD.DAT Simulates functions in the science bay. See Simulation. **(Note this file must be deleted prior to actual flights)**
- ZMEXT.DAT Automatically puts transferred files in the correct directory from any other directory.
- PROGLET.DAT This contains configuration of the science computer sensors and defines which sensors are installed in the Payload bay.

AUTOEXEC.BAT

This contains the path to the executable and the Picodos prompt.

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

11 Science data logging RELEASE_7.0

***** Soundbite for production release 7.0 *****

NOTE: This release contains science data logging, a major change to the way the glider stores and uploads data. Users, especially sites who have implemented their own shoreside data handling systems, should understand the following information before upgrading.

**** NOTE for legacy glider users: To use this release you should have an upgraded 1MB RAM Persistor for your science processor, and a large CF card on science. Please contact glidersupport. ****

Science Data Logging (SDL) - INTRO

This is an architectural change to address the limited throughput of the serial data connection (the so-called 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 which could be thus transferred. In the new system using SDL, the science sensors do not have to be transmitted over the clothesline, eliminating the bottleneck.

SDL - SCOPE AND CAVEATS

To fully implement this release, it is important to upgrade both the glider processor (to 7.0) and the science processor (to 3.0). This is absolutely necessary because the baud rate at which the clothesline is operated has been changed (from 9600 to 4800). However, with science 3.0 it is still possible to fall back to the old method of operation (sans SDL), though this is not recommended.

Only release 7.0 and forward 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.

SDL - PERFORMANCE

Since science isn't doing much else EXCEPT to collect and log the data, even though by default it is running at reduced CPU speed, it is able to maintain a high cycle rate. The cycle time is independent of the glider cycle time, and by default is set to 1 second. Of course, actual throughput attainable does depend on installed sensor load and sensor data stream parsing overhead.

SDL - DETAILS

Science now has a parallel logfile type for each logfile type on the glider:

```
glider -> science
-----
.dbd  -> .ebd
.mbd  -> .nbd
.sbd  -> .tbd
.mlg  -> .nlg
```

and each pair is formatted the same (e.g., .ebd is formatted same as .dbd).

Accordingly there are some new config files on science which also parallel like config files on the glider:

```
glider          -> science
-----
\config\mbdlist.dat -> \config\nbdlist.dat
\config\sbdlist.dat -> \config\tbdlist.dat
\config\highdens.dat -> \config\highdens.dat
```

and again each pair is formatted the same.

Just as m_present_time should be present in both mbdlist.dat and sbdlist.dat on the glider, sci_m_present_time should be present in both 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 var known to science, as seen by science)
heap
list (all vars known to science, with their values)
prunedisk
put (note: think about whether it propagates to glider side or not)
send
```

SDL - TRANSPARENCY

Whenever logging is turned on on the glider, a clothesline message is sent to turn it on on 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 of data files to the dockserver has been made 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 3 .tbd files, finding no match for *.sbd, and the glider will send 3 .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. The total number of files sent by two two processors together in this case will be 6.

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

SDL - EXCEPTIONS TO TRANSPARENCY

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

SDL - 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.

Sensors added:

c_science_send_all(bool), glider, default 0, tells science whether to send all vars, or just a few
m_mission_start_time(timestamp), glider, in order to propagate to science
m_science_readiness_for_consci(enum), glider, tells if ready or, if not, why not
sci_m_disk_free(Mbytes), science, how much space currently free on science
sci_m_disk_usage(Mbytes), science, how much space currently used on science
sci_m_present_secs_into_mission(sec), science, analog of m_present_secs_into_mission on science
sci_m_free_heap(bytes), science, analog of m_free_heap on science
sci_m_min_free_heap(bytes), science, analog of m_min_free_heap on science
sci_m_min_spare_heap(bytes), science, analog of m_min_spare_heap on science
sci_m_spare_heap(bytes), science, analog of m_spare_heap on science

sci_x_disk_files_removed(nodim), science, count of files removed by last prune cmd on science
sci_x_sent_data_files(nodim), science, count of files successfully xmitted by last science send cmd
u_sci_cycle_time(secs), glider, default 1.0, tells science how fast to run
u_sci_dbd_sensor_list_xmit_control(enum), glider, default 0 = always transmit header, to tell science what to do
x_science_logging_state(enum), glider, tells what science logging state is
u_science_send_time_limit_adjustment_factor(nodim), glider, default 0.5, see below

NOTE: 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 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.

If you need to run a huge send command for a large number of files, you can temporarily raise u_sci_cmd_max_consci_time to a potentially huge number for this purpose (there are 3600 seconds in a hour, 86,400 seconds in a day). You should exercise caution before doing this in the water, and you should take care to restore the normal setting after doing the long send.

Alternatively, you can just do repetitive sends until all files have been transferred.

12 Science data rate cookbook

****As of March 2010 the science data rate cookbook should no longer be needed. The introduction of Science data logging in Version 7 code and greater resolves the issue, see section 11. For releases 6.38 and earlier the below cookbook should be consulted for gliders with heavy science demands.****

A text document fully describing the original problem and the solutions is outlined below and can be found at:

<ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/doco/science/science-datarate-cookbook.txt>

Overview:

The problem:

A glider is unable to process all of the data in a timely fashion from the science bay when a number of science sensors are collecting large amounts of data simultaneously. The observed symptom is the periodic science_super ODDITY: Input ringbuffer overflow. In the time between these oddities, **M_SCIENCE_CLOTHESLINE_LAG(sec)** continues to climb as the glider falls further and further behind until eventually a ring buffer overflow occurs.

Glidens configured with heavily loaded science bays can:

1. Live with current behavior, i.e., get all the data for a while and remove the **M_SCIENCE_CLOTHESLINE_LAG** in post processing.
2. Reduce the data in time, i.e. increase
behavior: sample
b_arg: intersample_time
3. Reduce the amount of data, i.e. decrease
c_XXX_num_fields_to_send(nodim) for all the proglers to the minimum required number of output sensors.

The documentation found at website above quantifies the glider's ability to consume science data and give some guidelines for configuring the glider to get what portions of the payload data that is most important to the user.

13 Flight Data Retrieval

A separate document [GMC \(Glider Mission Control\) User guide](#) will instruct the user in use of Glider Terminal a Java application and one of the Dockserver applications.

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

The Dockserver can automate file retrieval, data storage and allow the user to display data and glider locations for easy viewing. Dockserver applications also allow for ftp manipulation of data.

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 the user to suit their particular mission needs.

This text will guide a user in retrieval of 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 Iridium service. You will be able to retrieve all *.dbd, *.mlg, and *.sbd files using a the Freewave modem.

- **.dbd** Dinkum Binary Data, all sensors (or variables) are recording and stored in this type file. See Appendix K. .dbd are large files in most cases it would be undesirable to transmit these files during a mission especially over iridium.
- **.sbd** Short Binary Data, records only those sensors specified in SBDLIST.DAT to reduce communication time. Users will 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:

- Type **s *.dbd** or **s *.sbd**, or **s *.mlg** or **s *.mbd**
- The 30 most recent files will be sent of the type specified. Remember it is not desirable to send .dbd files over Iridium.
-

To retrieve glider data while the glider is not running a missions from a GliderDos prompt:

-
- Type **send *.dbd**, **send *.sbd**, **send *.mbd**, **send *.mlg**, or **send *.***
- **Send *.*** sends all of the files of type **.sbd**, **.mbd**, **.dbd** and **.mlg**

To send specific files in Picodos or Glider Dos

Use Zmodem to send the files desired **zs <filename>**

Vehicle Status – Glider on surface counting down to resume mission.

Immediate commands available during surface mission paused dialog.

- Ctrl-r** resumes glider mission, if so programmed
- Ctrl-c** aborts the mission, closes files, and remains on surface in glider DOS
- Ctrl-e** extends surface time by 5 minutes before resuming mission
- no entry will resume the mission in the seconds specified
- Ctrl-p** Starts mission immediately

Ctrl-f re-read ma files
C switch to communications with the science bay Persistor
! followed by a allowable glider dos commands to runs a subset of glider DOS commands (commands listed as lower case in help menu not available during a mission.

14 Dockserver Data Visualizer

Version 6.33 Ashumet and later of Dockserver include a Data Visualizer. See Dockserver Guide for instruction of use. The Data Visualizer allows the user to plot all sensors from any data file transmitted to the Dockserver. See Appendix Gliderview for legacy data processing method.

15 Simulation

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

There are various ways to operate the glider in a simulated environment. This 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.

There are various levels of simulation:

(NOTE: MAJOR DAMAGE WILL BE CAUSED TO THE GLIDER IF THE SIMULATION FILE CONTAINS {no_electronics} or {just_electronics} WHEN USING A FULL GLIDER.)

no_electronics	Persistor alone - no glider board. - Pocket simulator
just_electronics	No Hardware, no motor, etc, just the electronics board and Persistor - Shoebox simulator
on_bench	This is a complete glider on the bench, i.e. not In water.

The level of simulation is set when the glider 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. A glider should never be deployed with a simul.sim file remaining in the config directory.

Iridium phone simulation

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. Normally this would be an Iridium satellite modem, although any Hayes-compatible modem should work.

null_modem: A null modem cable connects the Iridium connector on the electronics board to a terminal emulator. All modem responses such as "OK", "CONNECTED 4800", etc. 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.

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 typically requires a switch. On revision E and later glider control boards connect a wire between J25-6 and J25-7, switching Freewave power to CD input. Disconnecting this for at least 3 seconds simulates unplugging an actual Freewave modem. This is useful in testing code which 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_device: altimeter <min mt> <max mt> <probability> <bad value>
Make the altimeter return the specified bad value sometimes.
<bad value> The bad value to return

bad_device: attitude <min mt> <max mt> <probability> <bad value>
Make the attitude return an error sometime
<bad value> The error code to return, Ex.

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

Make the gps return a fix an added error:

<ini_err_meters> The error 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>

Make the pitch_motor appear to stall (jam) sometimes.

Should generate a motor not moving warning.

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

Make the buoyancy appear to stall (jam) sometimes.

Should generate a motor not moving warning.

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

Make the buoyancy overheat bit come up sometimes.

bad_device: memory_leak <min mt> <max mt> <probability> <bytes to leak>

Consume 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 simdrv 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 returning <bad value> as the drift sometimes.

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 sub cycle 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: simdrv.

The code is located in code/simdrv.c.

In general what we do at each of the levels:
not simulating

 Computes all the S_XXX variables.

 Inputs to A/Ds are NOT computed.

Note that the driver is consuming resources even though the results are not being used. On a real long-term mission you probably want to put the following in autoexec.mi or your mission file:

 USE simdrv 0

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 following argument may be added:

on_bench open

When operating a full glider on the bench WITH A DUMMY BOUYANCY PUMP MOTOR to cause the simulation code to simulate the vacuum reading to prevent vacuum aborts.

!!!! DAMAGE TO THE GLIDER WILL RESULT IF THE GLIDER IS OPERATED ON_BENCH, WITH THE GLIDER OPEN AND THE VEHICLE BOUYANCY 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.

 Just_electronics

 Computes all the S_XXX variables.

 Supplies simulated inputs to all A/Ds and shift registers

 Simulates all of the uarts EXCEPT science and Iridium

 Simulates modem responses if "null_modem" specified

 Removes Iridium from service if neither "null_modem" or "modem" is specified.

no_electronics

Requires only a Persistor.

Computes all the S_XXX variables

Supplies simulated inputs to all A/Ds, shift registers, and uarts.

To begin a mission one has to set the origination point of the glider by:

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

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

*A .mi file is available and will be included in production releases of code 6.34 that will allow a user to define all the simulated values and set them by using the **loadmission** command.*

SIMULATION IN THE SCIENCE BAY

```
config\appcmd.dat
```

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

```
-nog    No glider is attached
```

```
-sim    No sensors attached, simulate them
```

```
-echo_uart Show send/rcv lines from sensor uarts
```

```
-echo_cl Show send/rev lines from clothesline (glider)
```

16 Missions

Missions can be loaded to or drawn off the glider by using FTP and referring to section 3, of the GMC or Dock Server Users Guide. Alternatively below is the method used to manipulate files directly while using a terminal program and the Freewave modem.

To load a mission file into the glider, you must first have ready the mission file on your computer. Open a Glider Terminal and apply power to the glider. If the glider boots to GliderDos, exit to a GliderDos: prompt by using **control-c**. It is advisable to load all missions before starting field operations. However, when necessary, you can still load missions while in GliderDos using the same procedure.

At the C:/ prompt type

```
C:/>DOCKZR mission name.mi
```

The appropriate mission or file must first be in the to-glider directory on the ftp utility.

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 files, *.ma files.

One important mi file to be aware of is the Autoexec.mi mission file which resides in the config directory. The Autoexec.mi controls many of the individual settings particular to each glider. This is where calibrations for motors are stored, where the phone number for iridium dialing to a Dockserver is stored and where the glider draws its name.

MA Files

The *.ma files called by *.mi files can contain behaviors that are often modified, 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_107.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 insure that everything is running properly.

- Connect the Freewave radio to the antenna and a computer.
- Run the terminal program on you computer.
- Power the glider by inserting the Green Band Plug.
- Check that glider is not under **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.)
- Run Status.mi and check everything is working properly, check you have a GPS fix, check the level of batteries, check no errors appears on the screen on you computer.

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

APPENDIX – Code Theory and Operation

The brain of the SLOCUM AUGV 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 vehicle also contains a separate Persistor for the collection and logging of scientific data (science computer) in addition to CTD measurements, which are logged by the glider computer.

a. Operating Systems

The operating system for the Persistor CF1 is **Picodos** (**P**ersistor **I**nstruments **C**ard **O**r **D**isk **O**perating **S**ystem). Picodos is smaller than, but very similar to DOS, and many DOS commands will work in Picodos. Uploading of new Glider Controller code, new mission files, and retrieval of data files are all done in Picodos.

GliderDos is an application that is loaded into the Persistor. This resident operating system is a super-set 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 SLOCUM vehicle via Motocross.

b. Code Design

The user commands the SLOCUM glider by writing **mission files** (text files with a .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 themselves, 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.

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

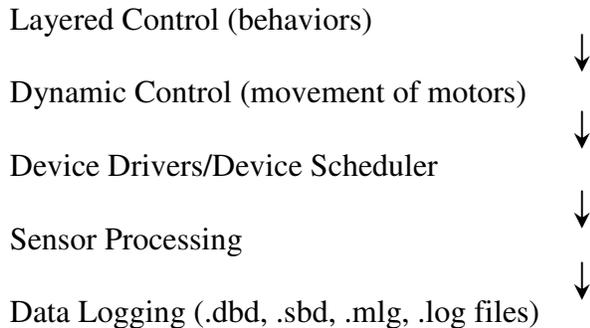
<ftp://ftp.glider.webbresearch.com/glider/windoze/production/src/code/masterdata>

Only experienced users should manipulate the masterdata file located on your glider. 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 over written; this is done at glider start up. 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 over written 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 SLOCUM 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 glider movement. Once layered control determines this, Dynamic Control moves the motors, inflates/deflates the bladders, etc., to initiate the movement or **behavior** commanded under the layered control. In order to do this, Dynamic Control uses a specific set of Device Drivers to perform the movements. Sensor Processing involves the algorithm calculations to assign values (1 or 0) to the **sensors**. data Logging is self explanatory, except that values of all sensors (i.e.: variables), instruments, gps fixes, etc are actually logged.

c. Control: Stacks, States & 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-sequences.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

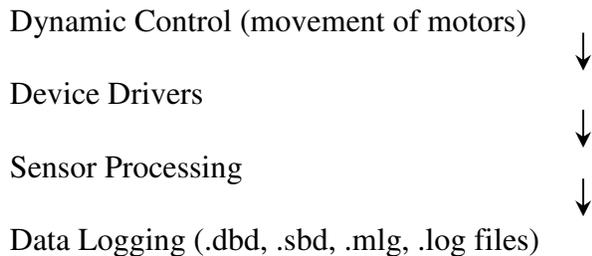
Towards the end of MASTERDATA is a list of numbers and their ‘assigned’ actions. This list is named ‘beh_args.h’ and can be found in the C code. This list primarily deals with the two *b_arg* statements: **b_arg: start_when** and **b_arg: stop_when**, and determines when a particular behavior becomes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

d. General Control Structure

For a typical glider deployment, layering behaviors in a pre-determined 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 kinds of aborts, which the glider can trigger, to get to the surface. They are *synchronous abort*, *out of band abort* and a *hardware generated abort*. Each type of abort utilizes different control layers to perform the 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:



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* stands for 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 below:

e. Abort Codes

- 3: MS_NONE
- 2: MS_COMPLETED_ABNORMALLY
- 1: MS_COMPLETED_NORMALLY
- 0: MS_IN_PROGRESS
- 1: MS_ABORT_STACK_IS_IDLE
- 2: MS_ABORT_HEADING_IS_IDLE
- 3: MS_ABORT_PITCH_IS_IDLE
- 4: MS_ABORT_BPUMP_IS_IDLE
- 5: MS_ABORT_THRENG_IS_IDLE
- 6: MS_ABORT_BEH_ERROR
- 7: MS_ABORT_OVERDEPTH
- 8: MS_ABORT_OVERTIME
- 9: MS_ABORT_UNDERVOLTS
- 10: MS_ABORT_SAMEDEPTH_FO
- 11: MS_ABORT_USER_INTERRUPT
- 12: MS_ABORT_NOINPUT
- 13: MS_ABORT_INFLECTION
- 14: MS_ABORT_NO_TICKLE
- 15: MS_ABORT_ENG_PRESSURE
- 16: MS_ABORT_DEVICE_ERROR
- 17: MS_ABORT_DEV_NOT_INSTALLED
- 18: MS_ABORT_WPT_TOOFAR

*Abort Codes #5 and #15 are used with the Thermal Gliders only.

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 2 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.

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 2 hours or 16 hours, set by a jumper in the glider control board and referred to as COP_TICKLE (COP = Computer Operating Properly).

f. 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 mfiles/surfac10.ma. This is a new feature of software in which the '10' in
enum (see notes below for more on b_arg enum values) tells GliderDos to surface only if
the conditions in the .ma file are met. The corresponding .ma file is displayed below:
behavior_name=surface
# surface-20deg.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>
```

#This allows for 'generic' missions to be written with new conditions for b_arg's to be inserted. A more practical (or obvious) use of this feature is to insert a behavior_name = goto_list which contains inserted waypoints for the glider to go to. Then, you will not have to go into the actual mission file to change the waypoint coordinates; instead you can just insert a new .ma file on the glider with the waypoints.

```
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
  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 mafeiles/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 mafeiles/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 way point

behavior: surface

```
b_arg: args_from_file(enum) 10
# read from mfiles/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 mfiles/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.

```
#behavior: surface
# b_arg: args_from_file(enum) 10 # read from mfiles/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 it's 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 that the glider will glide to.

behavior: goto_list

b_arg: args_from_file(enum) 10

read from mafiles/goto_110.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_110.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: yo
  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 as long after waiting for 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 most of the input sensors on

behavior: sensors_in

<0 off, 0 as fast as possible, N, sample every N secs

b_arg: c_att_time(s) -1.0

b_arg: c_pressure_time(s) -1.0

b_arg: c_alt_time(s) -1.0

b_arg: u_battery_time(s) -1.0

b_arg: u_vacuum_time(s) -1.0

b_arg: c_profile_on(s) 0.0

b_arg: c_gps_on(bool) 0.0 # Special, 1 is on, 0 is off

APPENDIX – GLIDER SOFTWARE WEB SITE

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

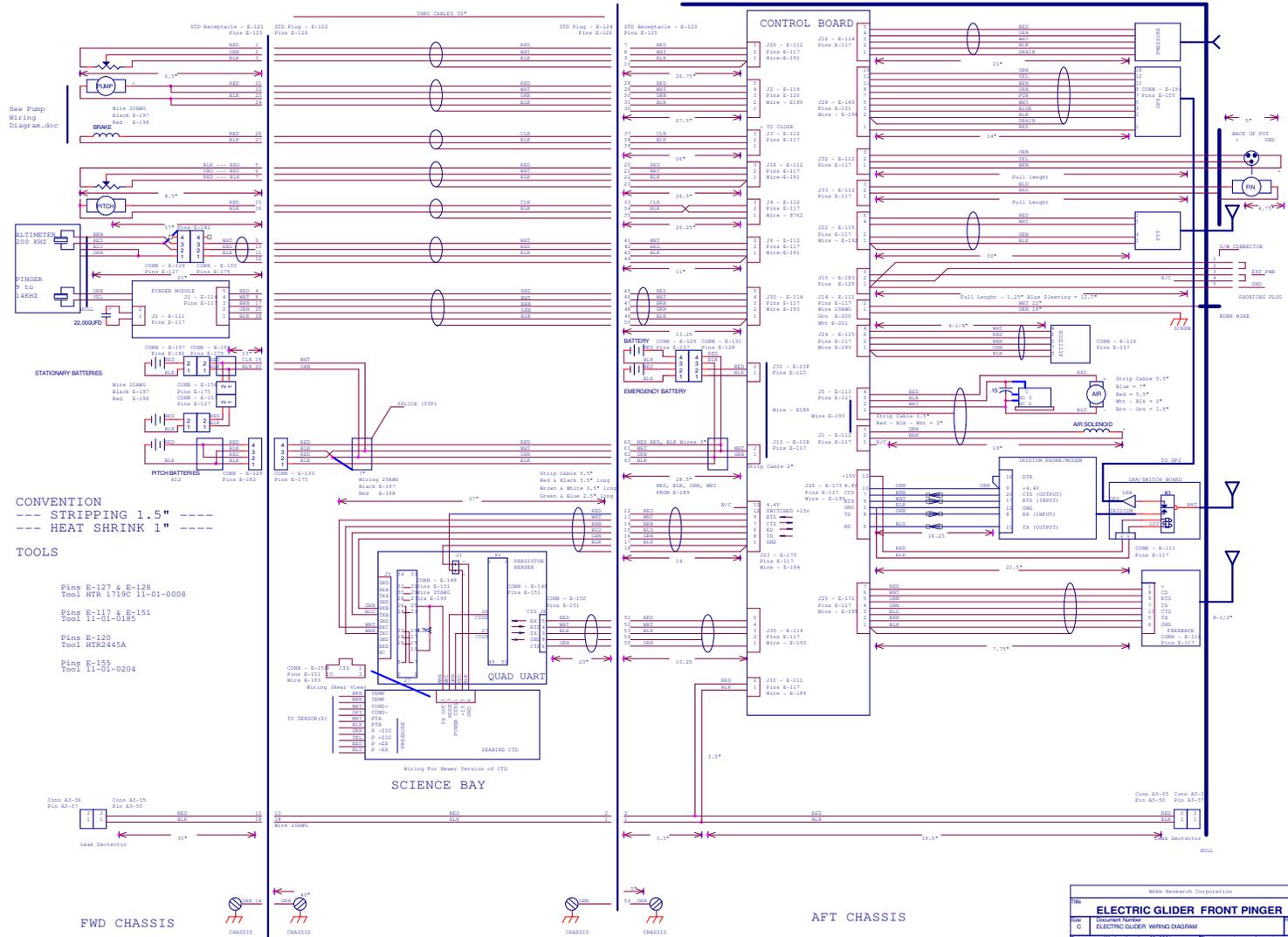
A user name and password will be provided or can be requested. TWR will require organization, name, phone number and email addresses of each person using access. To access documents log in at:

<https://dmz.webbresearch.com/>

The Glider Software Web Site is as follows:

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

APPENDIX - WIRING DIAGRAM



APPENDIX – COMPASS

Present compass is the TCM3. Talk command is not available. Recalibration is a three point system. Contact Glidersupport@webbresearch.com if you need assistance in recalibration of this sensor. To determine if the factory configured compass is a Tcm3 confirm presence of the following text in Autoexec.mi:
installed attitude_tcm3

Original glider Compass

The CAL3 calibration requires you to take one measurement, turn the system exactly 180 degrees, take another measurement, then turn the system upside-down exactly, and take a third measurement. The compass is taking two measurements on each axis with the directions reversed between them. The average of the two measurements is the hard iron vector that the compass needs to correct.

Calibration

Power the Glider on either use an external power cable (15 volts DC) or insert the (green band) Green Plug. Start communication with the vehicle and make sure the glider is in Picodos. Type **talk att** and hit enter. If the compass is displaying data, type **h** and **enter** until the output stops.

Position the glider on a flat surface heading north.

Type **CAL3** and hit Enter.

The compass will take a measurement and prompt you to turn the system 180. It is very important that this be as exact as possible.

Turn the glider exactly 180 and press the keyboard's *Space bar* to continue.

The compass will take a measurement then you will be prompted to turn the system upside-down. It is very important that the vertical axis in the new position be exactly 180 degrees from the original position.

Turn the system upside-down and press the keyboard's *Space bar* to continue.

The compass will take the third measurement and calculate the hard iron correction.

Configuration

Make the following configuration changes to the compass. Each of the following will need to be typed separately followed by the enter key. Note that if the data is entered properly, the compass will only respond with a carriage return, but if the data is entered improperly the compass will respond with E010 error. Simply re-enter the data. Typing the function then **?** will report the stored value (*ex. ec=?*).

ec=e compass enabled

ep=e pitch enabled

er=e roll enabled

et=e temperature enabled

ed=e Enable Magnetic Distortion Alarm

damping = d Disable Dampening

fast = d Disable fast sampling

clock = 5 Set clock to 5Hz

- a. Type *go* and hit *enter*; confirm that the data is streaming, then type Control C twice to exit.

APPENDIX –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 function on a 9-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.

Set-up glider shore side Freewave

The following is the procedure setting up the glider shore side Freewave. This mode allows for a shore side Freewave to communicate with several Slaves.

1. Connect the transceiver to the serial port of your computer through a serial cable.
2. Open up a Hyper Terminal session.
 - Use the following settings in connecting with hyper terminal
 - 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 get into the Operation Mode menu.
 - Press 0 to set the radio as a point to Point to Point Master.
 - Press Esc to get back to 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 that the radio is attached.
 - Press 1 to set the radio communication baud rate to 115,200.
 - Press Esc to get back to Main menu.
6. At the Main Menu, press 3.
 - Set the Frequency Key
 - Press 0 to set to change the Frequency Key
 - Press 5 to set to change the Frequency Key to 5

- Press Esc to get back to Main menu.

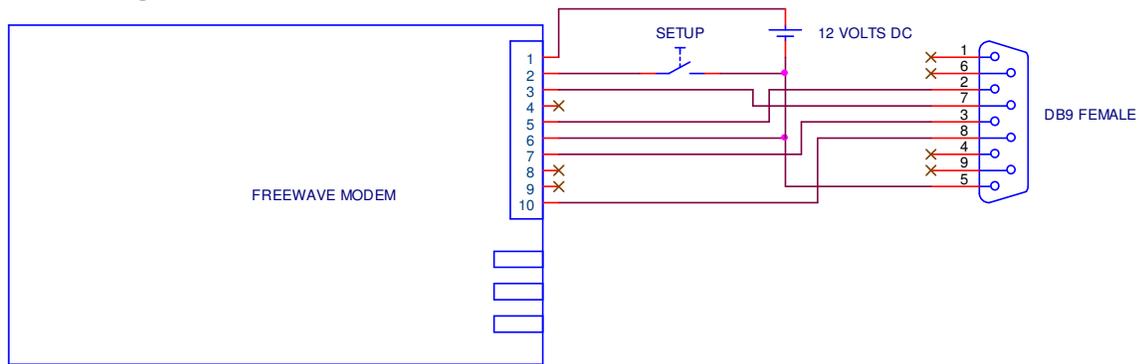
7. At the Main Menu, press 2.

- 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 in which this master will communicate with.
- 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 get back to Main menu.
- Press Esc to exit set up.

Set-up glider Freewave slave (Internal to the glider)

The following is the procedure setting up the glider slave Freewave. This mode allows for a slave to communicate with several shore side Freewaves.

1. Connect the transceiver to the serial port of your computer through a serial cable, per the drawing below.



2. Open up a Hyper Terminal session.

- Use the following settings in connecting with hyper terminal
- 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 get into the Operation Mode menu.

- Press 1 to set the radio as a Point to Point Slave.
- Press Esc to get back to 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 that the radio is attached.

- Press 1 to set the radio communication baud rate to 115,200.
- Press Esc to get back to Main menu.

6. At the Main Menu, press 3.

- Set Frequency key.
- Press 0 to set to change the Frequency Key
- Press 5 to set to change the Frequency Key to 5
- Press Esc to get back to Main menu.

7. At the Main Menu, press 2.

- Press 0 (*0 through 9 may be used*). To add the serial number of the shore side Freewave.
- Press C to select the glider in which this slave will communicate with.
- Press 0 to select entry 0, 1 to select entry 1, etc. or A for all radios on the list.
- Press Esc to get back to 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 6-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 omni directional antennas with cable lengths ranging from 3 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 2 feet in antenna placement can solve noise problems. In extreme cases, such as when the transceiver is located close to Pager or Cellular Telephone transmission towers, the standard and cavity band pass filters that Freewave offers can reduce the out-of-band noise.

APPENDIX Iridium Service and SIM CARD

To obtain a Iridium Sim Card you will need to locate and choose a provider. Iridium charges can be a significant expense and 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 Stratos:

<http://www.stratosglobal.com/StratosGlobal.cfm>

[1] 954-217-2265.

CLS America or ARGOS

<http://www.clsamerica.com/>

Another provider, NAL Research offers competitive rates:

<http://www.nalresearch.com/Airtime.html>

Another provider:

<http://www.infosat.com/>

support@infosat.com,

1-800-207-55759

Specify "**data only**" service. No equipment 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 prior to shipment. Teledyne Webb Research will need the actual card and the unlocking pin. This is so the card can be unlocked and the PIN code deactivated permanently.

Note: Iridium usage based on one of our users averages 90 minutes per day per glider using some of the data reduction techniques that we provide. This is roughly \$108/day cost. Plan on 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.

APPENDIX ARGOS Satellite Service and ID

Glider users must provide IDs in both decimal and hexadecimal
Standard repetition rate 90 seconds.

To establish ARGOS Service:

Submit a Program Application Form to the 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 ARGOS Technical Information Form:

Processing: A2 (hex output)
Results format: DS
of Platforms : # of IDS
Lifetime of Platform ; # 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 request 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

Note: It is vital to ensure that IDs are not already in use. Please verify this.

Note: ARGOS ID Format Change

Since 1999, CLS ARGOS has been preparing to change the format of platform IDs.

~2003 Service Argos began offering 2 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. Note this will change the format of ARGOS data. For gliders with 20 bit ids and X-cat transmitter a user should refer to legacy ARGOS Data format for decoding. For gliders with 20 or 28 bit ids and Smart cat transmitters use rev1 format for decoding.

Customers may need to revise their data processing software accordingly.

Teledyne Webb Research can provide data format information upon request. In the event your glider is missing and Argos transmissions are needed to locate the vehicle contact Service Argos and ask that ALP processing (All Location Processing) be activated.

APPENDIX 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 transmitter type refer to the Autoexec.mi and the following sensor and sensor notes:

sensor: f_argos_format(enum)

Below is the data format for gliders with X-cat ptt's and 28 bit Argos Ids.

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

file: argos-data-format-rev1.txt

This defines the 31 byte packet of data sent to argos by the glider. For a description of the legacy 32 byte format see argos-data-format.txt.

NOTE WELL: If you change what's in the packet, you'll need to modify:
code\argos.c:construct_31b_data_to_xmit_to_argos()
code\prntargo_rev1.c
If you change an encoding scheme, you'll have to modify:
code\argoscode.c
code\argosdecode.c

23-Mar-05 pfurey@DinkumSoftware.com Initial
27-Apr-05 pfurey@DinkumSoftware.com Fixed typo.

byte offset in decimal	sensors	Meaning
0-3	M_PRESENT_TIME	Present time (at start of cycle, secs since 1970)
4-11	M_GPS_LAT/LON	Current location and age of fix in minutes
12-19	M_GPS_INVALID_LAT/LON	Last unvalidated fix and age in minutes

20-27	M_GPS_TOOFAR_LAT/LON	Last "too-far" fix (from dead reckoning point) and age in minutes
28,29	X_WATER_VX/VY	Water current components in LMC.
30		8 bit checksum of bytes 0-29

Present time:

Seconds since 1970 encoded in 4 bytes (32 bit unsigned long)

Locations and age:

The same encoding scheme is used for current location, last invalid, and last too-far fixes.

The location consists of a latitude and longitude. The age is in minutes from "now", the "now" must be extracted from the argos email header.

A latitude or longitude is encoded in 3 bytes. The age is encoded in two bytes.

0,1,2	latitude
3,4,5	longitude
6,7	age in minutes

The input datum format on the glider is:

[-] [D] DDMM.MMMM

where the Ds are degrees and Ms are minutes.

The latitude or longitude is rounded to nearest 1/100 of a minute (about 20 meters), multiplied by 100, and stored as a signed integer.

i.e. 4235.12354 would be stored as
423512 in 3 bytes

The age in minutes is stored as an unsigned 2 byte integer. An all ones value 0xFFFF (hex) means either that the location was never computed -or- it was computed more than 65K minutes ago.

Water current:

The water current is encoded in 2 bytes, the units are m/sec and are in Local Mission Coordinates where +Y is magnetic north. The velocity components are scaled by 1/0.05 prior to encoding and then scaled by 0.05 when decoded, and have a valid range of +/- 12.8 m/sec.

8 bit checksum of bytes 0-29:

Used for error checking.

Alg: sum the bytes and takes one's complement
(ie; negate and add 1 (~sum+1))

The following data format will be transmitted with the discontinued Smart-cat transmitter

file: argos-data-format.txt

This defines the 32 byte packet of data sent to argos by the glider.

NOTE WELL: If you change what's in the packet, you'll need to modify:
code\argos.c:construct_data_to_xmit_to_argos()
code\prntargo.c
If you change an encoding scheme, you'll have to modify:
code\argoscode.c

06-Sep-00 tc@DinkumSoftware.com Initial
07-Sep-00 tc@DinkumSoftware.com Added sensors

byte offset in decimal	sensors	Meaning
0-7	M_GPS_LAT/LON	Current location and age of fix
8-15	C_WPT_LAT/LON	Current waypoint and when started for it
16-23	X_LAST_WPT_LAT/LON	Last waypoint achieved and when
24,25	X_MISSION_NUM	Current or Last Mission number
26	X_MISSION_STATUS	Curr or last mission status
27	X_OLD_MISSION_STATUS_1	status of mission before last
28	X_OLD_MISSION_STATUS_2	status of mission, two missions ago
29	X_OLD_MISSION_STATUS_3	status of mission, three missions ago
30,31		16 bit crc of bytes 0-29

Locations and age:

The same encoding scheme is used for current location, last waypoint, and next waypoint.

The location consists of a latitude and longitude. The age is in seconds from "now", the "now" must be extracted from the argos email header.

A latitude or longitude is encoded in 3 bytes. The age is encoded in two bytes.

0,1,2 latitude
3,4,5 longitude
6,7 age in seconds

The input data format on the glider is:

```
[-][D]DDMM.MMMM
```

where the Ds are degrees and Ms are minutes.

The latitude or longitude is rounded to nearest 1/100 of a minute (about 20 meters), multiplied by 100, and stored as a signed integer.

i.e. 4235.12354 would be stored as
423512 in 3 bytes

The age in seconds is stored as an unsigned 2 byte integer. An all ones value 0xFFFF (hex) means either that the location was never computed -or- it was computed more than 65K seconds ago.

Current or Last Mission number.

The mission number is YYDDDxx

YY year

DDD day in year

xx the xx'th mission of the day

This is encoded into two bytes as an unsigned integer.

The MSB is stored first, followed by LSB

DDDxx

Note: Once must examine Curr or last mission status to determine if that mission is still running -or- it terminated and glider is sitting at GliderDos prompt.

Mission status:

A single byte representing the status of a current or prior mission:

These are decimal numbers that are taken from command.h. It is likely that command.h will be changed and this document won't.

So consult code\command.h for the final word:

```
MS_IN_PROGRESS=0,
```

```
MS_ABORT_STACK_IS_IDLE=1, // layered_control()
```

```
MS_ABORT_HEADING_IS_IDLE=2, // layered_control()
```

```
MS_ABORT_PITCH_IS_IDLE=3, // layered_control()
```

```
MS_ABORT_BPUMP_IS_IDLE=4, // layered_control()
```

```
MS_ABORT_THRENG_IS_IDLE=5, // layered_control()
```

```
MS_ABORT_BEH_ERROR =6, // layered_control() behavior  
entered error state
```

```
MS_ABORT_OVERDEPTH =7, // behavior abend
```

```
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
```

	29	Mission never ran (-3)
MS_NONE		
	30	not used (-2)
MS_COMPLETED_ABNORMALLY		
	31	Mission completed normally
(-1) MS_COMPLETED_NORMALLY		

16 bit crc of bytes 0-29:

Used for error checking. See code\crc_16.c for algorithm.

It was taken from:

C Programmer's Guide to Serial communications, 2nd Edition
page 779

Joe Campbell

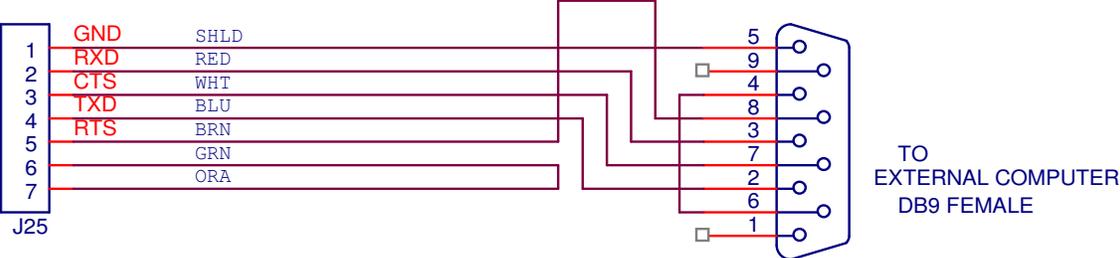
Sams Publishing

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 cmd prompt to work properly. Please contact glidersupport@webbresearch.com if you are having trouble using this tool while searching for your glider.

APPENDIX – J25 TO DB9 TO DB25 WIRING



DB9	Signal	DB25
1	DCD	8
2	RD	3
3	TD	2
4	DTR	20
5	SGND	7
6	DSR	6
7	RTS	4
8	CTS	5
9	RI	22

APPENDIX HOW TO DETERMINE MISSION LONGEVITY

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

A user should monitor the battery voltage from the glider surface dialog or included in the data stream by monitoring the following masterdata sensor.

sensor: m_battery(volts)

When plotting m_battery a user should note that voltage drops during heavy current usage – ie buoyancy pump adjustment at depth are expected and normal.

The glider will begin aborting missions for under voltage at 10 volts per the abort behavior below:

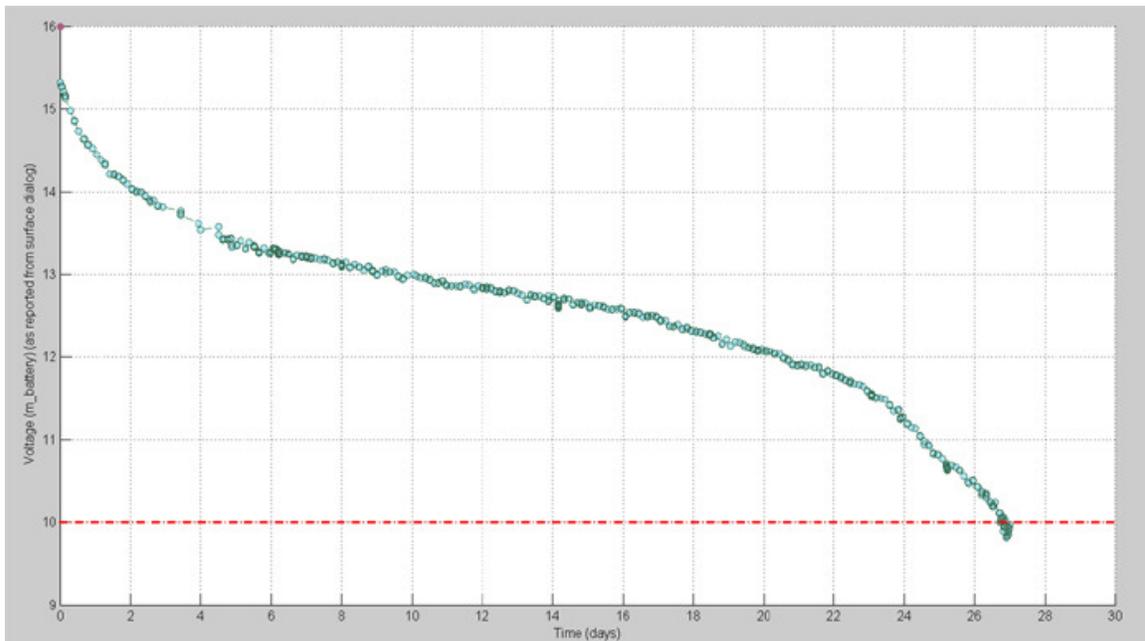
b_arg: undervolts(volts) 10.0 # < 0 disables

The following document can be used to estimate battery longevity:

Electric Glider Mission Spreadsheet.xls

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. A user 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 a user finds themselves 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.

APPENDIX COMMONLY PURCHASED SPARE PARTS

Commonly replaced parts include:

Battery packs

Note: The Teledyne Webb Research design uses alkaline manganese dioxide batteries (Duracell). It does not contain lithium primary batteries, which are regulated as hazardous materials per international law (UN3091). Transportation and disposal of batteries is greatly simplified by avoidance of lithium batteries. Some customers install lithium batteries, which increases endurance.

Typical glider usage with alkaline manganese dioxide batteries averages 600 – 700 hours of operation per battery pack. This is approximately one month of flight. This calculation is highly dependent on the sensor package carried by the glider; battery usage increases with increased sensor payload. Contact Teledyne Webb Research at glidersupport@webbresearch.com for a .xls spreadsheet for estimating battery longevity.

Glider Battery:

Full set includes: (1) Pitch Battery Set, (1) Aft Battery set, (2) Nose battery sets.
Credit issued with return of battery cores.

The following parts require occasional replacement:

Anode (ASSY G-540)

Burn Wire Assembly (ASSY G-123)

Note: This is the ejection weight, which will release if the glider fails to surface per planned mission. It will need to be replaced each time it is activated.

Wing (G-237D)

Hull O-rings (G-024) Parker only 2-265-N674-70

Dessicant

APPENDIX Ancillary Glider Equipment:

Ballast Tank; min. size: 8' (2.5m) long X 4' (1.2m) X 3' (1m)

This is for manual set-up of glider.

A way to get the glider in, and out of the tank; i.e. an overhead **winch**, a low enough tank to go over the side, or some lifting device.

Vacuum Pump; Thomas, model # 2688CE44, available from Grainger (Grainger item # 5Z350) or equivalent, to pull the Glider vacuum.

Gram Scale; 0kg-2kg to measure internal ballast

Hanging Gram scale to measure weight of glider in ballast tank

Lead Shot or ballast material

Iridium account (sim card); To enable satellite communications
See appendices for further information.

ARGOS; Account, and ID

See appendices for further information.

Land phone line; To receive Iridium satellite calls or introduced in 2008 Rudics is available for data transfer.

APPENDIX – BALLASTING AND H-MOMENT ADJUSTMENTS WORKSHEET

Glider Name: _____

Date: _____

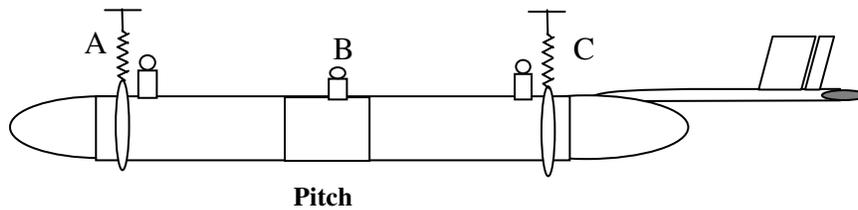
Glider Displacement: _____ kg

Tank Water Temp: _____ **Target Water Temp:** _____

Tank Water Density: _____ **Target Water Density:** _____

	Tank (fresh water)	Target	Adjustment	Weight Change
Salinity (pss)	0			
(T)emperature (°C)			Displacement * $70 \cdot 10^{-6}$ * (target T – Tank T) * Target D	
Pressure (psi)	0			
(D)ensity g/L	1000		Displacement * [(target D / Tank D) - 1]	

Current Weight Configuration



	Section A	Section B	Section C
Spring Scales			
Weights Added			

Static Roll: _____

X Ballast weight constants

S. Steel 0.912			
Lead 0.875			

= Total weight change

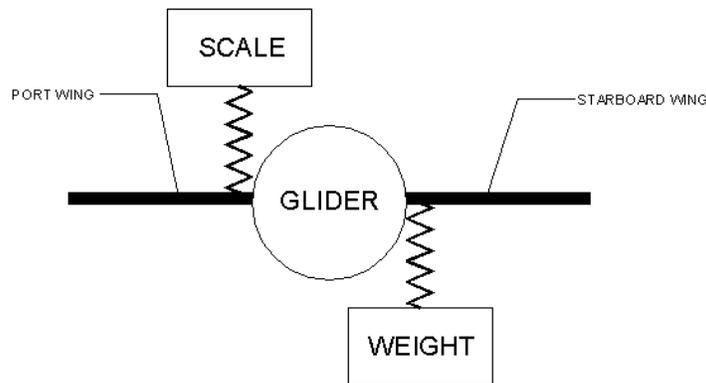
Final Weight Configuration.

Weight Section A	Weight Section B	Weight Section C

APPENDIX – CALCULATING THE H-MOMENT

This factor accounts for buoyancy provided by water on material.

- ✓ Place well-ballasted glider with wings in a ballasting tank.
- ✓ Measure the static angle of roll. An inclinometer may be used.
- ✓ Hang a known amount of weight (~300 g) from the starboard wing rail.
- ✓ Attach a spring scale to the port wing rail.
- ✓ Measure the weight change on the spring scale.
- ✓ Measure the angle of roll that the glider undergoes due to the addition of weight. An inclinometer may be used.
- ✓ Remove the added weight, measure it, and multiply by 0.912 if using Lead weight or by 0.875 if the weight used is stainless steel.



Weight added to starboard wing rail. (W)g _____

Weight shown on spring scale. (S)g _____

The difference in angle of rotation prior to weight added and the angle of rotation after to weight added. _____

The diameter of the Hull . (H) 107mm

The glider displacement. (D) _____ kg

$((S+W)*H)/(D*1000*TAN(W*(\pi()/180)))$

H-distance (specification = 6mm +/- 1mm). _____

In order to lower the H-distance, it will be necessary to add weight to the weight bar in the payload bay and the reverse is necessary to increase the H-distance.

Appendix Command examples

attrib:

```
GliderDos I -3 >attrib +r c:\config\autoexec.mi  
[Makes it read only]  
[in the current directory you don't need the C:\ part]
```

```
GliderDos I -3 >attrib autoexec.bat  
A R C:\AUTOEXEC.BAT  
[shows attributes of autoexec.bat]
```

capture:

```
GliderDos I -3 >capture c:\config\simul.sim  
Ready, <CTRL-C or BREAK> to end  
on_bench  
[hit control-C here]  
Complete
```

dir:

```
[plain]  
GliderDos I -3 >dir  
Volume in drive C is NONAME  
Volume Serial Number is 7B17-2E37  
Directory of C:\CONFIG\  
  
.  
..  
CONFIG.SCI  
CONFIG.SRF  
DELLOG.DAT  
HIGHDENS.DAT  
LOGIN.EXP  
LONGTERM.DAT  
MBDLIST.DAT  
SBDLIST.DAT  
SENSDATA.DAT  
ZMEXT.DAT  
AUTOEXEC.MI  
SIMUL.BAK  
SIMUL.SIM  
  
13 file(s) 56,503 bytes  
2 dir(s) 125,399,040 bytes free
```

[verbose]

GliderDos I -3 >dir /v

Volume in drive C is NONAME

Volume Serial Number is 7B17-2E37

Directory of C:\CONFIG\

File Name	Size	Allocated	Modified	Attrib
.	<DIR>		09-04-08 10:27a	D
..	<DIR>		09-04-08 10:27a	D
CONFIG.SCI	449	2,048	07-29-08 3:45p	A
CONFIG.SRF	41	2,048	07-29-08 3:45p	A
DELLOG.DAT	442	2,048	07-29-08 3:45p	A
HIGHDENS.DAT	549	2,048	07-29-08 3:45p	A
LOGIN.EXP	1,222	2,048	07-29-08 3:45p	A
LONGTERM.DAT	857	2,048	09-27-06 6:12p	A
MBDLIST.DAT	295	2,048	07-29-08 3:45p	A
SBDLIST.DAT	480	2,048	07-29-08 3:45p	A
SENSDATA.DAT	46,207	47,104	07-29-08 3:45p	A
ZMEXT.DAT	1,361	2,048	07-29-08 3:45p	A
AUTOEXEC.MI	4,541	6,144	04-12-08 12:59p	R A
SIMUL.BAK	49	2,048	01-21-08 7:23p	A
SIMUL.SIM	10	2,048	09-04-08 3:12p	A

13 file(s) 56,503 bytes
2 dir(s) 75,776 bytes allocated
125,399,040 bytes free
128,116,736 bytes total disk space, 2% in use

[tighter specs]

GliderDos I -3 >dir c:\state

Volume in drive C is NONAME

Volume Serial Number is 7B17-2E37

Directory of C:\STATE

.	<DIR>		09-04-08 10:29a
..	<DIR>		09-04-08 10:29a
LONGTERM.STA	555		09-04-08 2:50p
PFSTDERR.TXT	0		09-04-08 2:54p
PFSTDOUT.TXT	776		09-04-08 2:54p

3 file(s) 1,331 bytes
2 dir(s) 125,399,040 bytes free

GliderDos I -3 >**dir c:\config*.mi**

Volume in drive C is NONAME

Volume Serial Number is 7B17-2E37

Directory of C:\CONFIG\

AUTOEXEC.MI 4,541 04-12-08 12:59p

1 file(s) 4,541 bytes
0 dir(s) 125,399,040 bytes free

rm:

GliderDos I -3 >**rm c:\old_app**

Path C:\OLD_APP

-- delete entire branch (10 seconds to reply)? (Y/N) -> y

Deleting branch at path C:\OLD_APP ...

put:

GliderDos I -3 >**put c_science_all_on -1**

1899 79 sensor: c_science_all_on = -1 secs

rename:

GliderDos I -3 >**rename config\simul.sim simul.bak**

GliderDos I -3 >**dir config**

Volume in drive C is NONAME

Volume Serial Number is 7B17-2E37

Directory of C:\CONFIG

.	<DIR>	09-04-08 10:27a
..	<DIR>	09-04-08 10:27a
CONFIG.SCI	449	07-29-08 3:45p
CONFIG.SRF	41	07-29-08 3:45p
DELLOG.DAT	442	07-29-08 3:45p
HIGHDENS.DAT	549	07-29-08 3:45p
LOGIN.EXP	1,222	07-29-08 3:45p
LONGTERM.DAT	857	09-27-06 6:12p
MBDLIST.DAT	295	07-29-08 3:45p
SBDLIST.DAT	480	07-29-08 3:45p
SENSDATA.DAT	46,207	07-29-08 3:45p
ZMEXT.DAT	1,361	07-29-08 3:45p
AUTOEXEC.MI	4,541	04-12-08 12:59p
SIMUL.BAK	49	01-21-08 7:23p

12 file(s) 56,493 bytes

2 dir(s) 126,162,944 bytes free

send:

GliderDos I -3 >**send -f=irid *.sbd**

Enumerating and selecting files

About to send 17 files

[...]

GliderDos I -3 >**send -f=rf *.sbd**

Enumerating and selecting files

About to send 17 files

Prechecking is not necessary for this invocation

[...]

SHUFFLING FILES.....

Sent 17 file(s):

c:\logs\00310000.SBD c:\logs\00300020.SBD c:\logs\00300019.SBD

c:\logs\00300018.SBD c:\logs\00300017.SBD c:\logs\00300009.SBD

c:\logs\00300008.SBD c:\logs\00300000.SBD c:\logs\00290001.SBD

c:\logs\00290000.SBD c:\logs\00280000.SBD c:\logs\00270000.SBD

c:\logs\00260000.SBD c:\logs\00250000.SBD c:\logs\00240000.SBD

c:\logs\00230000.SBD c:\logs\00220000.SBD

SUCCESS

[end of a bad transfer]

Sent 0 file(s):

FAILURE xmit_to_host(): results=1

Error sending files

[-num option sets max num files]

GliderDos I -3 >**send -num=2 *.dbd**

Enumerating and selecting files

About to send 2 files

[...]

[-t sets max time]

GliderDos I -3 >**send -t=60 *.dbd**

Enumerating and selecting files

About to send 1 files

[...]

setdevlimit:

GliderDos I -3 >**setdevlimit attitude 10 20 5**

Setting limits for ATTITUDE:

os : # times device can be put back into service: 10

w/s: # warnings/segment before error: 20

w/m: # warnings in last minute before error: 5

APPENDIX – How to edit a proglet.dat file:

From Picodos or GliderDos, type: **consci** to transfer to communicating with the science Persistor. While in a mission type **C**

Structured in Picodos or Sci Dos of the science Persistor are the following:

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.

Type **cd config**

Type **dir**

Confirm proglet.dat is in config directory.

To transfer the proglet.dat from the config folder to Dockserver type **zs proglet.dat**

Preserve original proglet.dat by changing name. Type **rename proglets.dat proglets.org**

Edit proglet.dat in "from glider" directory. Below is an excerpt from proglet.dat of commenting out an Aanderaa sensor 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)  
  
#-----
```

Is edited as follows to remove 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)
```

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

Move the edited proglet.dat file to the To glider directory on the Dockserver.

Confirm that glider is still in Scidos (There is a 1200 second default timeout value). **cd** to config directory if necessary. Type **dockzr proglets.dat**. When transfer is complete Type **type proglet.dat**. The new file will now be displayed to screen. Confirm edits.

Type **quit** to exit science computer.

If in mission type **!use - science_super** wait for science to power down. Then type **!use + science_super**

If not in a mission Type **exit reset**

Proceed.

APPENDIX – DBD_FILE_FORMAT

This document specifies the Dinkum Binary Data format. This is the format of the data transmitted from Teledyne Webb Research Glider to a host computer for processing.

Table of contents

- >>>HISTORY and RATIONALE
- >>>MISSION NUMBERING SCHEME
- >>>DBD FORMAT
- >>>HOST SIDE PROCESSING
 - >>>>>rename_dbd_files:
 - >>>>>dbd2asc:
 - >>>>>dba_time_filter
 - >>>>>dba_sensor_filter
 - >>>>>dba2_orig_matlab:
 - >>>>>dba2_glider_data
- >>>DBA FORMAT
- >>>DATA SIZE REDUCTION RESULTS
- >>>FUTURE PLANS
- >>>HISTORY OF RELEASED VERSIONS

>>>HISTORY and RATIONALE

The Glider originally used an OBD (Odyssey Binary Data) format. This consisted of an ASCII header and binary data representing "sensor data". On every cycle, 6 bytes were transmitted for every sensor that was updated with a new value: 2 bytes of sensor number and 4 bytes of floating point data representing the sensor.

This format had a couple of problems:

1. Once on the host, one couldn't tell whether a sensor was updated with the same value -or- it wasn't updated. This presented a severe problem for artificial, i.e. non real-world, sensors.

2. The data compression wasn't optimal:

- a. Sensors with small dynamic range, e.g. 0 or 1, still required 4 bytes to transmit.
- b. The overhead of two bytes for each sensor number was excessive.

To address these needs, the DBD (Dinkum Binary Data) format was invented. It also consists of:

An ASCII header A list of sensor names, units, and number of bytes encoded in. Binary data representing "changed" sensors on each cycle.

The binary data is encoded quite differently. At the beginning of every cycle, two bits are transmitted for EVERY sensor. These two bits encoded one of three states:

- The sensor wasn't updated
- The sensor was updated with the same value
- The sensor was updated with a new value.

This "cycle state" is followed by the binary values (at 1, 2, or 4 bytes) for only those sensors that were updated with a new value.

This adds some fixed overhead per cycle of transmitting 2 bits for every sensor (56 bytes currently). If 13% or more of the sensors change on every cycle, the new format results in less data and fully communicates the state of the sensors. In a one hour simulated mission, 16% of the variables changed per cycle.

See DATA SIZE REDUCTION RESULTS for final numerical comparisons.

>>>MISSION NUMBERING SCHEME

A mission is uniquely identified by 2 different numbers:

unique_mission_number 0-9999
mission_segment 0-9999

Each mission is given a unique number. Is it incremented every time a mission is started over the life of the vehicle. Any mission can be broken into a number of segments. The initial segment of a mission is segment 0.

A new segment can be created any number of ways:

- The mission is interrupted and resumed by operator at a keyboard.
- The glider surfaces to transmit data.
- The glider decides to do so because the current segment is to "big".

The most visible impact of segments relates to log files. When a segment is ended, all the log files are closed and flushed to disk. When a new segment is created, new mission files are opened with a different file.

There is a bunch of strings/filenames related to mission_number/segment. There are a lot of constraints on these filenames. Primarily, the Persistor only has an 8.3 file system, i.e. filenames can only be 8 chars with a 3 char extension.

In the old *.OBD file days, a mission was identified as "Z0122202.xxx"

- Z first letter of vehicle name
- 01 The last two digits of year
- 222 The day in the year
- 02 The (n-1) mission of day, e.g. 3rd

This didn't let us handle segments or many different vehicle names. So to get around these limitations, we have two "names" for each log file.

An 8.3 compatible name and a long file name. The 8.3 name simply needs to be unique on the glider's file system, once it is transferred to another computer with a reasonable file system, it gets renamed to the long file name. There are host side tools to do this automatically. See rename-dbd-files.

The 8.3 name is:

mmmmssss.XXX

where mmmm is the unique mission number

ssss is mission segment number

After 10,000 missions or segments, they wrap around.

The long filename looks like:

zippy-2001-222-04-05

- zippy vehicle name
- 2001 The year AT START OF MISSION
- 222 day of year AT START OF MISSION
- 04 the (n-1)th mission of the day
- 05 the segment of the mission

Both the 8.3 and full names are stored in all the data files.

When used for labeling, a string of the form is used:

zippy-01-222-04-05(0123.0005)

The long-term memory of unique_mission_number/segment is kept in disk file "state/mis_num.txt". You never need delete this file, it periodically gets trimmed in length. The format is:

```
mmmm ssss YY DDD UUUU zippy-01-222-04-05
```

where mmmm is unique mission number

ssss is mission segment number

zippy... is the long base filename

YY year of mission start

DDD day of year of mission start

UUUU mission number of the day

>>>DBD FORMAT

The details of the DBD Format:

<<AN ASCII HEADER>>

<<A sensor list in ASCII>> UNLESS FACTORED

<<A known bytes binary cycle>>

<<A data cycle with every sensor value transmitted>>

<<data cycles>>

.....

<<end of file cycle>>

<<AN ASCII HEADER>>

An example ASCII header is shown below. It consists of whitespace separated (key,value) pairs:

```
dbd_label: DBD(dinkum_binary_data)file
encoding_ver: 4
num_ascii_tags: 14
all_sensors: T
the8x3_filename: 00410000
full_filename: zippy-2001-115-2-0
filename_extension: dbd
mission_name: SASHBOX.MI
fileopen_time: Thu_Apr_26_07:35:08_2001
total_num_sensors: 216
sensors_per_cycle: 216
state_bytes_per_cycle: 54
sensor_list_crc: 5A87DC29
sensor_list_factored: 0
```

The meanings of the fields:

dbd_label: Identifies it as a DBD file
 encoding_ver: What encoding version. Version 0
 is reserved for the future development
 of an OBD to DBD translator.
 num_ascii_tags: The number of (key,value) pairs
 all_sensors: TRUE means every sensor is being transmitted
 FALSE means only some sensors are being
 transmitted, i.e. this is an *.SBD file
 the8x3_filename: The filename on the Glider.
 full_filename: What the filename should be on the host.
 filename_extension:
 mission_name: The name of the mission being run.
 fileopen_time: Human readable date string
 total_num_sensors: Total number of sensors in the system
 sensors_per_cycle: Number of sensors we are transmitting/cycle
 state_bytes_per_cycle: The number of "state bytes" sent per
 cycle. These are the state @ 2 bits/sensor.
 sensor_list_crc: CRC32 of the section <<A sensor list in ASCII>>.
 sensor_list_factored: 1 if the section <<A sensor list in ASCII>>
 is present, 0 if factored.

<<A sensor list in ASCII>>

This section is not present in every file.

It is present if sensor_list_factored (see above) is 0, and it is not present if sensor_list_factored is 1.

If it is not present, the topside processing software has to find the correct <<A sensor list in ASCII>> by inspecting other files, looking for one with the same sensor_list_crc, and a sensor_list_factored of 0.

Factoring is controlled by the value of the sensor u_dbd_sensor_list_xmit_control.

If present:

One line is sent for EVERY sensor, regardless of whether it is being transmitted. An example list:

```

s: T 0 0 4 f_max_working_depth m
s: T 1 1 4 u_cycle_time s
s: T 2 2 4 m_present_time s
.....

```

The "s:" marks a sensor line.

The next field is either:

- T sensor being transmitted
- F sensor is NOT being transmitted

The next field is the sensor number, from 0 to total_num_sensors-1

The next field is the index, from 0 to sensors_per_cycle-1, of all the sensors being transmitted. -1 means the sensor is not being transmitted. if all_sensors is TRUE, then this and the prior field will be identical.

The last numerical field is the number of bytes transmitted for each sensor:

- 1 An integer value
- 2 An integer value
- 4 A float value

The last two field are the sensor name and it's units.

<<A known bytes binary cycle>>

Three known binary values are transmitted. This allows the host to detect if any byte swapping is required:

- s Cycle Tag (this is an ASCII s char)
- a one byte char
- 0x1234 two byte integer
- 123.456 four byte float
- 123456789.12345 eight byte double

<<A data cycle with every sensor value transmitted>>

This is like a regular data cycle, but every sensor is marked as updated with a new value. This represents the initial value of all sensors. See <<data cycle>> for the format.

<<data cycle>>

The data cycle consists of:

- d Cycle tag (this is an ASCII d char)
- <state bytes> there are state_bytes_per_cycle of these
- <sensor data> 1,2,4, or 8 bytes for every sensor that was
- updated with a new value.
- <sensor data>

The state bytes consist of 2 bits per sensor. The MSB of the first byte is associated with the first transmitted sensor. The next two bits, with the next sensor, etc. Any unused bits in the last byte will be 0.

The meanings of the two bit field for each sensor:

MSB	LSB	
0	0	Sensor NOT updated
0	1	Sensor updated with same value
1	0	Sensor updated with new value
1	1	Reserved for future use

<<end of file cycle>>

X cycle tag (a single ASCII X char)

There may very well be data in the file after this last cycle. It should be ignored. The persistor currently has a real annoying habit of transmitting a bunch of Control-Z characters after the end of valid data. This ought to be fixed.

>>>HOST SIDE PROCESSING

>>>>>rename_dbd_files:

A program, rename_dbd_files, will rename the transmitted mmmmssss.dbd files to their full filenames. Versions of this program exist for Windows and Linux.

It takes its arguments from the command line AND from stdin if -s is on the command line.

Usage: rename_dbd_files [-s] <file> ... <file>

The names of all renamed files are echoed to stdout. Any non-dbd file or a *.dbd file that has already been renamed will be silently ignored. So, "rename_dbd_files *" works just fine. For windows users, I might suggest:

```
dir /b *.* | rename_dbd_files -s
```

>>>>>>dbd2asc:

A single program, dbd2asc, will read a DBD (or SBD) file(s) and convert them to a pure ASCII format. Versions of this program exist for Windows and Linux.

<ftp://ftp.glider/glider/windoze/production/windoze-bin/>

<ftp://ftp.glider/glider/linux/production/linux-bin/>

Usage: (this may be dated)
dbd2asc [-s] <filename> ... <filename>

Builds a list of files (DBD or SBD) from all the filenames on the command line and from stdin if -s is present.

The data from all the files are merged and written to stdout in an ASCII format. See next section, >>>DBA FORMAT, for a description.

The intent is to provide a series of filters, which are piped together to produce the desired results.

The following will process all the DBD files in a directory:

Windows: dir /b *.dbd | dba2asc -s

The following will process all the SBD files from a given mission:

Linux: ls cassidy-2001-193-28-*.sbd | dba2asc -s

>>>>>dba_time_filter

Reads dinkum binary ascii data from stdin (output of dbd2asc) Throws away some data based on time Writes the remainder ad dinkum binary ascii data to stdout

Usage:

dba_time_filter [-help] [-epoch] earliest_included_t latest_included_t

All records between earliest_included_t and latest_included_t (inclusive) are output. All others are discarded. The time is normally based on mission time (M_PRESENT_SECS_INTO_MISSION). If -epoch is present, time is based on seconds since 1970 (M_PRESENT_SECS)

A -tf is added to the filename of the output header, e.g.

filename: zippy-2001-222-04-05 --> zippy-2001-222-04-05-tf

>>>>>dba_sensor_filter

Reads dinkum binary ascii format from stdin (output of dbd2asc) Excludes the data of sensors NOT listed on the command line or in the -f <sensors_filename> Writes the data of remaining sensors in dinkum binary ascii format to stdout

Usage:

dba_sensor_filter [-h] [-f <sensors_filename>] [sensor_name_0 ... sensor_name_N]

Accepts a .dba file from stdin. The data corresponding to sensors listed on the command line or in the -f <sensors_filename> are written to stdout as a .dba file. All other sensor data is discarded. Sensor names in -f <sensors_filename> should be line or space delimited.

A -sf is added to the filename of the output header, e.g.

filename: zippy-2001-222-04-05 --> zippy-2001-222-04-05-sf

>>>>>dba2_orig_matlab:

Reads dinkum binary ASCII data from stdin (output of dbd2asc) and writes two matlab files (your filenames will vary):

zippy_2001_104_21_0_dbd.m
zippy_2001_104_21_0_dbd.dat

The output format is identical to the initial matlab files produced in the development of the Teledyne Webb Research Glider.

To use the file, from matlab, execute:

zippy_2001_104_21_0_dbd.m

Thus a typical usage is:

dbd2asc 0041001.dbd | dba2_orig_matlab

As other data processing needs arise, additional filters can be written. See >>>FUTURE PLANS for some that have been conceived, but not implemented yet.

>>>>>dba2_glider_data:

Reads dinkum binary ASCII data from stdin (output of dbd2asc) and writes two matlab files (your filenames will vary):

zippy_2001_104_21_0_gld.m
zippy_2001_104_21_0_gld.dat

The .m file contains the same information as the .m file produced by dba2_orig_matlab but formatted as a matlab struct. In addition, this struct contains the segment filenames corresponding to the input .dba file's composite .dbd files, and it contains struct elements representing all of the .dba header keys and values. The .dat file is the same as that produced by dba2_orig_matlab.

The generated .m and .dat files are for consumption by the Matlab Glider_Data application.

was not updated that cycle.

The labels in an ascii header may have a number of Xs in them when multiple data sets are merged. Basically any characters in the ascii header fields which are different in any of the input DBD files are replaced by an X. An example may help:

```
dbd2asc cassidy-2001-193-28-0.dbd cassidy-2001-193-28-1.dbd ...
      cassidy-2001-193-28-2.dbd cassidy-2001-193-28-3.dbd
```

Would result in the following header:

```
dbd_label: DBD_ASC(dinkum_binary_data_ascii)file
encoding_ver: 0
num_ascii_tags: 17
all_sensors: 1
filename: cassidy-2001-193-28-X
the8x3_filename: 0088000X
filename_extension: dbd
filename_label: cassidy-2001-193-28-X(0088000X)
mission_name: BOX.MI
fileopen_time: Fri_Jul_13_2X:XX:XX_2001
sensors_per_cycle: 249
num_label_lines: 3
num_segments: 4
segment_filename_0: cassidy-2001-193-28-0
segment_filename_1: cassidy-2001-193-28-1
segment_filename_2: cassidy-2001-193-28-2
segment_filename_3: cassidy-2001-193-28-3
```

The Xs in the header represent characters that are different in the four input DBD file headers.

>>>DATA SIZE REDUCTION RESULTS

On a simulated one hour mission:

the OBD format generated 381 Kbytes/hour

the DBD format (transmitting 4 bytes for every sensor)
generated 278 Kbytes/hour. A 27% improvement.

The DBD format (transmitting a variable number of bytes per sensor) generated 289 Kbytes/hour. Note that this is bigger than the 4 bytes/sensor. It was the same mission, but longer. I can only assume that the extra mission time involved actions that resulted in more variables changing. This should probably be investigated.

On a simulated five hour mission:

The SBD sensor list from New Jersey '00 (24 sensors) produced data at the rate of 50Kbytes/hour.

APPENDIX –Legacy Gliderview.exe

Is legacy data visualization tool and is minimally supported. Data server described above is the preferred method for data viewing.

GliderView.exe may be downloaded from:

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

It is a self-extracting archive and it will self extract to the current directory.

GliderView.exe is a single self-extracting archive of everything needed to run the Glider View application on Win2K, NT 4.0 or WinXP.

Procedure to install and run:

- 1) Create an empty working directory, which you do not care if it is filled with a tree containing some tens of megabytes of files.
- 2) Copy GliderView.exe to that directory.
- 3) Run GliderView.exe from that directory.
- 4) Again from that directory, run the file go.bat, which will have been created in that directory in step 2.

You can do all this either from a command window, or graphically from Windows Explorer.

The batch file will set GLIDER_PARENT_DIR and PATH appropriately for you, but only locally (it won't upset your global variables), and run Glider View.

Two data files are included in subdirectory "data", created in step 2. These are purely for testing.

1152x864 resolution is required to see the display properly.

Don't press the data buttons at the bottom until after you have loaded some data.

1. Loading new data:

1. Go to *Data* menu and click on the 'load data' option.
2. Chose the data file you wish to load. This file can be a 'sbd' or a 'dbd' file. This file can also be a short-name format (ex: 01230001.sbd or dbd) or a long-name format (glider-2003-196-0-0.dbd)
3. After clicking ok, the data is loaded and ready to view.
4. If you wish to load a single segment of the data set, uncheck the 'All Segments' option in the *Data* drop-down menu. Then 'load data'.

- **Sensor Filter:**

The 'sensor filter' option allows the user to choose a specific set of sensors to process into a data file. Clicking on 'sensor filter' will cause the program to look for a file called 'sensor_list_file.txt' in the current data directory. If this file exists, and contains a list of the wanted sensors, delimited by either spaces or new lines, the file will be processed and the data displayed. If the file does not exist, the user will be prompted to choose the location of the sensor_list_file.txt. Only the sensors in the file will be processed and displayed.

Example of file contents:

m_depth m_present_secs_into_mission

c_ballast_pumped

OR

m_depth

m_present_secs_into_mission

c_ballast_pumped

- **Time Filter:**

The 'time filter' option allows the user to determine a time range during the mission to look at. This time is specified by the user through a dialog box when the 'time filter' option is used. The time range must be specified in number of seconds from the beginning of the mission.

Note: If the 'All Segments' option is left checked in the *Data* drop-down menu, then any segment clicked on will process **all** data segments.

2. Viewing data and clearing data:

1. Viewing Data

1. Now that the data is loaded, common engineering and science data can be viewed by simply pushing the buttons that correspond to the desired elements. This option is usually most helpful when viewing 'dbd' files.
2. To plot two different sensors against each other, high-light the wanted sensors in the list box, using the 'CTRL' key for the second selection, and press the '**Plot**' button. This will activate the '**invert y-axis**', the '**Swap x y**' and the '**Rad to Deg**' checkboxes that you can click to help view your data in a more reasonable format. You cannot choose more or less than two sensors for this function.
3. The list box at the right of your screen contains all commanded, measured and science sensors. You can choose up to 14 of those sensors to plot against time with the '**plot vs. time**' button. Multiple sensors are chosen by holding down the 'CTRL' key and choosing the required sensors.
4. The '**Map it!**' button will give you a map plot of your current glider position and the waypoints that the glider is heading to. If the required

sensors do not exist in the sbd file that you are viewing, you will be unable to map glider position.

5. The '**Track it!**' button is a new feature and still needs a lot of work. In theory, this button will launch a new window that suspends the operation of Glider View until it has been closed. This new window, running the 'Track_glider' program, will allow the user to automatically monitor a capture file from the Freewave or Iridium output of a glider and display the data. To run this feature, press '**Start**' in the 'Track glider' window, chose the capture file required, then press ok. To stop operation, press the toggle button '**Stop and Save**', save your data to the desired filename, then you can either restart or '**Close**' the window. Glider View operation should resume once you are done tracking data.
6. The '**Sensor:**' editbox allows you to type in a sensor not shown in the list box and to plot it vs. time. To find the sensor, there exists a list of all sensors in the help menu. To plot only the sensors in the box, chose the None variable in the list box. To plot more than one sensor including the sensors in the sensor box, use the 'CTRL' key to choose none in addition to the sensors desired from the list box.

2. Clearing Data

It is not necessary to clear data each time before loading new data. Clicking the 'clear data' option will simply clear all plots, list boxes and remove any displayed data.

3. The help menu:

The help menu is self explanatory and launches help information in the Matlab help window in HTML format.

4. The Fly-through option:

This option is designed to allow the user to view a previously run mission. The '**Fly-through**' button is only activated for DBD file-types. Clicking on the button will start a fly-sequence of the loaded mission. A 3D plot of the flight path in latitude, longitude and depth is displayed in the main plot. A second plot shows the glider adjusting heading and pitch according to recorded data and a third compass plot displays glider heading and magnitude of the horizontal speed of the glider. To increase the mission replay time, increase the number in mission speed from 1 to the desired number. To do this you must first stop the mission, make your change, and then start the mission again. To move forward into a mission, use the slider to jump ahead in the mission. This option can only be used once. This will change in future releases. The '**Pause**' button pauses the fly-through. The user cannot continue any other functions while in pause mode. The '**Stop**' button stops fly-through and any other functions can be used.

APPENDIX –Slocum Glider Do’s and Don’ts

THINGS TO NEVER DO WITH A GLIDER

Never power up a glider without a vacuum

Never run a simulation on a glider other than “on_bench”

Never deploy a glider in simulation

Never pick a glider up by the rudder/fin (digifin can be handled)

Never deploy a glider in “boot pico”

Never exit to pico during a deployment.

Never power a glider with more than 15v DC from an external power supply.

Never deploy a glider in lab_mode

Never perform the top of a yo below 30 meters (with 100 or 200 meter glider)

THINGS TO DO WITH A GLIDER

Do secure it properly in crate with all 3 straps

Do use fresh desiccants on each deployment

Do monitor internal vacuum before launch (less vacuum indicates a leak; positive pressure may indicate dangerous gas accumulation)

Do simulate missions before launch

Do test Iridium and ARGOS telemetry before launch

APPENDIX – Storage conditions

For optimum battery life, storage temperature range is +10 to +25 degrees C. When activated, the glider should be equilibrated at a temperature between -2 and +54 degrees C.

APPENDIX – Returning equipment to factory

Before returning equipment, contact TWR for RMA.

Email: Glidersupport@webbresearch.com.

All returns from outside USA, please specify our import broker:

Consignee: Teledyne Webb Research
82 Technology Park Dr
East Falmouth, MA 02536 USA

Notify: DHL-Danzas Freight Forwarding Agents
Attention: Ellis Hall, Import Broker
Phone (617) 886-6665, FAX (617) 242-1470
500 Rutherford Avenue, Charlestown, MA 02129 USA

Note on shipping documents: US MADE GOODS

Please note:

All crates large enough to hold a human must be metal banded.

TWR recommends shipping by air only (and strongly discourages truck transport over long distances as this frequently causes damage).

CAUTION: If the glider was recovered from the ocean, it may contain water, which presents a safety hazard due to possible chemical reaction of batteries in water, which may generate explosive gases (see Battery Warning and Disclaimers). In this case, be sure to remove the seal plug to ventilate the instrument.

APPENDIX- Slocum Glider Operator Guide “quick reference and checklists”

To download or review the most recent Operators Guide visit:

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