Slocum G2 Glider Operators Training Guide

Revised July 2014

Teledyne Webb Research
49 Edgerton Drive
North Falmouth, MA 02556
USA
+1 (508) 563-1000
www.webbresearch.com

TELEDYNE
WEBB RESEARCH
Everywhereyoulook™

The material in this document is for information purposes only and is subject to change without notice. Teledyne Webb Research assumes no responsibility for any errors or for consequential damages that may result from the use or misrepresentation of any of the material in this publication.

FreeWave is a registered trademark of FreeWave Technologies. Iridium is a registered trademark of Iridium Communications Inc. Persistor and PicoDOS are registered trademarks of Persistor Instruments, Inc.

Contents

Introduction

This document is a field guide and reference documentation for use in preparing and deploying Teledyne Webb Research's Slocum G2 Gliders.

Please also refer to the complete *Slocum Glider Operators Manual* at:

https://datahost.webbresearch.com/files.php?cwd=/glider/production/doco/MANUAL

Datahost is an authorized user restricted site. You will need to register for access.

Glider Operation and Maintenance Training

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@teledyne.com for information about training sessions. Our company's policy is to fully support only properly trained individuals and groups.

Internet Resources

Software distribution:

https://datahost.webbresearch.com/files.php?cwd=/glider/production

Slocum Glider Operators Manual:

https://datahost.webbresearch.com/files.php?cwd=/glider/production/doco/MANUAL/Slocum_G2_Glider_Operators_Manual.pdf

GMC Users Guide (Dockserver Manual):

https://datahost.webbresearch.com/files.php?cwd=/glider/production/doco/gmc/gmcUserGuide.pdf

Glider service bulletins:

https://datahost.webbresearch.com/viewtopic.php?f=13&t=205&p=553&hilit=bulletins#p553/

Glider code update procedure:

https://datahost.webbresearch.com/files.php?cwd=/glider/production/doco/updating-all-glider-software.txt

Masterdata:

https://datahost.webbresearch.com/files.php?cwd=/glider/production/masterdata

Notes and Warnings

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

➡

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

CAUTION

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

⚠

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

Format Notes

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

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

When this handout is 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:

https://datahost.webbresearch.com/

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

Customer Service

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

Notes for Ballasting and Lab Tests

If the glider is not already closed up with a proper vacuum, you will need to do this before you can apply power to the glider. To do this pull the glider together with the tie rod using the provided 24"-long T-handle hex wrench until the hulls have come together. Set the torque to 15 in/lbs using the torque handle and long extension provided. With the vacuum tool and the long T-handle, put a vacuum on the glider. Your target is 6" Hg (7" Hg for 1000 m), but it is best to pull a vacuum higher than this as you can bleed some off when the glider is powered on. Once this is accomplished, and the MS plug is in place, you may apply power. The glider will power on and go through its normal start up routine.

When you see:

```
SEQUENCE: About to run initial.mi on try 0
```

you have 120 seconds to type a control-C to terminate the sequence.

- The control-P character immediately starts the mission.
- All other characters are ignored.

Type `CTRL-C`. This will give you a GliderDOS prompt. From the GliderDOS prompt:

1. Type `callback 30`. This will hang up the Iridium phone for 30 minutes. You can enter any value for callback from 1 to 30. Alternately you can type `use -iridium` to take the iridium out of service until you are done with your testing.

NOTE If you do this, remember to type `use + iridium` when you are finished to put the iridium back into service.

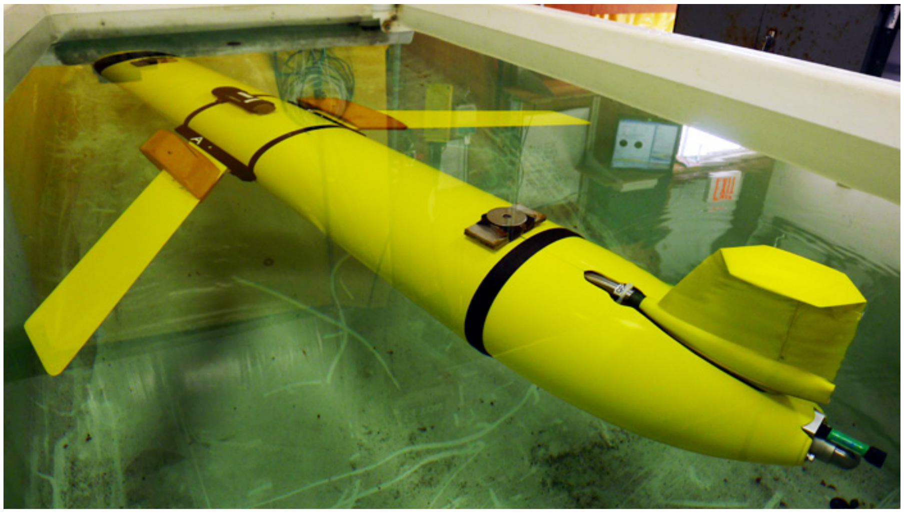
2. Type `lab_mode on`. This puts the glider in lab mode and will prevent the glider from trying to run its default mission.

3. Type `ballast`. This deflates the air bladder, sets the pitch motor to zero, and the sets the buoyancy pump to zero.

WARNING If any device is removed from service during the time the glider is in the ballast tank the glider will move the buoyancy engine to full displacement and the bladder will inflate. The most common reason for this is the attitude sensor being taken out of service due to local magnetic fields. A user should periodically check that all devices are in service by typing "use". If a device is out of service the user should determine if it is critical for ballasting and type ballast again when satisfied that the required devices are in service.

4. Type `report ++ m_vacuum`. This displays the vacuum inside the glider every time the sensor updates. If the vacuum is already at 6" Hg (7" Hg for 1000m), you are done (+/- 0.2). If not, you need to adjust the vacuum. If the vacuum is higher than the target you can bleed air into the system now. If it is lower than the target, air can be removed from the system using the vacuum tool while reporting vacuum values.
5. Type `report clearall`. This stops outputting the vacuum value.
6. Put the aft cowling on the glider. If you are connected via an external power supply, you may need to power down by typing `exit` before installing the cowling. Repower, if necessary, and follow steps 1-3. You are now ready to put the glider into the ballast tank.
7. You will need to get CTD data from the glider so that you can make your final weight adjustment calculations from the ballast tank to real conditions. To do this, turn on all sensors in the science bay by typing the individual commands below or by typing `loadmission sci_on.mi`
   - `put c_science_all_on 0 (off = -1)`. This will tell the science computer to sample all science sensors as fast as possible.
   - `put c_science_on 3 (off = 1)`. This will display that data to the screen.

- `put c_science_send_all 1 (off = 0)` to send science to flight Persistor.



8. Note in the above graphic we see a glider ballasted well for the tank. This means no pitch or roll, and the top of the fin is almost breaking the surface while the glider is commanded to the `ballast` position.
9. Pick out the water temperature and conductivity. Enter this data in the appropriate blocks (tank values) on the Ballasting and H-Moment Checklist (see page 47). It is important to remember that you need to make the glider neutral in the tank and calculate the H-moment before you make a mass adjustment for your operational area. After measuring H moment remember to enter the temperature, density and salinity for your target water into the appropriate blocks to get your total weight change from tank to real world conditions.

   To calculate the H-moment with the glider neutral in the tank:

   Type `report ++ m_pitch m_battpos`

   This displays the pitch and the position of the pitch battery of the glider in radians and inches respectively every time the sensor updates.
10. Follow the instructions for calculating the H-moment on the Ballasting and H-Moment Checklist (see page 47).

Common Lab Commands

To do this:	Type this:
Exit lab mode	While in `lab_mode on`, type `lab_mode off`. **NOTE: Never launch the glider in `lab_mode`.**
Zero motors and deflate the air bladder	`ballast` **NOTE: Never launch the glider in `ballast`.**
Stop Iridium phone calls	`use - iridium` or `callback 30` **NOTE: Never type `use - iridium` when in water**
Report a sensor as fast as possible	`report ++ (any_masterdata_sensor)` Example: `report ++ m_battery`
Change a sensor	`put (any_masterdata_sensor)` Example: `put c_fin 0` (zeros the fin after a wiggle)
Turn off all reporting	`report clearall`
Exercise the ballast pump, pitch motor, and fin motor	`wiggle on` (only available in lab_mode)
Stop exercising the motors	`wiggle off` (only available in lab_mode)
Tell the science computer to sample all science sensors as fast as possible	`put c_science_all_on 0 (off = -1)`
Display science data to the screen	`put c_science_on 3 (off = 1)`
Send science data to the flight Persistor	`put c_science_send_all 1 (off = 0)`
Apply power to the glider in an open state (no vacuum)	Follow these steps before powering down and opening the glider: • Type `exit pico`. This will bring you to a PicoDOS prompt. • Type `boot pico` to set the glider to boot into PicoDOS. If the ballast pump is already all the way forward or the pump is unplugged the application can run on the bench without a vacuum. Running the ballast pump without a vacuum can damage the forward rolling bellafram. • Type `app -lab` from PicoDOS to enter straight into lab_mode on. When you are finished, close the glider, apply the vacuum, and type boot app to set the glider to boot the application. You must always make sure the glider is set to boot app before doing any in the water tests.
Cycle default settings	`exit reset`
Remove green plug or power supply; install red plug	`exit` and wait for the prompt

Pre-mission Checkouts

These procedures should be followed to qualify a glider so that it can be launched for a mission. TWR can provide current versions of the Functional Check-out Procedure and Pre-Deployment Procedure by request.

On the Beach, Deck and/or outside at the Lab

NOTE When a glider is qualified on the beach, deck, and/or at a lab, it must be outside with a clear view of the sky.

1. Power on the glider.
2. When prompted, type `control-C` to exit to GliderDOS.
3. From the GliderDOS prompt, type `callback 30` to hang up the Iridium phone.
4. Type `lab_mode on`.
5. Type `put c_gps_on 3`.
6. Confirm the GPS.

   In the example string below the highlighted character should change from V to A.

   ```
   gps_diag(2)cyc#:538|GPRMC,161908,A,5958.3032,N,
   7000.5568,W,0.000,343.9,190808,0.3,W|
   ```

   After a number of **A** responses, type `put c_gps_on 1` to stop the screen display.
7. Type `wiggle on` and run for 3-5 minutes for shallow pumps to check for any device errors or other abnormalities. For deep pumps `report ++ m_de_oil_vol` and ensure full retraction and extension. Type `wiggle off` to stop wiggling.
8. Type `report ++ m_vacuum`. (Remember, the vacuum can fluctuate with the temperature.)
9. Type `report ++ m_battery`.
10. Type `report clearall`.

11. If no errors are found, type `lab_mode off` to return to the GliderDOS prompt.

NOTE Make sure that the glider is not simulating or in boot Pico or lab mode before deployment.

12. Purge the log directory, and send the logs over FreeWave or `dellog`. (This can take a long time if there are a large number of files and they will be lost, so the best practice is to purge and archive the log files in the lab.)
13. Type `run status.mi` and confirm that all sensors are being read. The mission should end with this message: “mission completed normally.”

NOTE The following sensor may not update during running status.mi (this is OK): `surface_2: Waiting for sensors to report. ERROR behavior surface_2: Timed out waiting for input sensors:ERROR behavior surface_2:    Sensor NOT reporting: m_raw_altitude`

14. Let the glider connect to the Dockserver and send .sbd over Iridium files. If not connected, type `callback 1` to force the Iridium to call in one minute once connected.

    Here is an example of forcing Iridium while the FreeWave is present:

    GliderDOS I -3 >`send -f=irid *.sbd -num=2` (This sends the two most recent .sbd files over Iridium. Be patient, because the Iridium is slow, and currently there is no positive feedback over FreeWave).

Science Sensor Checkout

1. Type `loadmission sci_on.mi` and `loadmission sci_off.mi` if available or type the individual commands below:
2. Type `put c_science_all_on 0 (off = -1)`. This tells the science computer to sample all science sensors as fast as possible.
3. Type `put c_science_on 3 (off = 1)`. This displays that data to the screen.
4. Type `put c_science_send_all 1 (off = 0)` to send science to the flight Persistor.

5. Verify that science output seems reasonable for all sensors installed. TWR will provide check out procedures for some sensors by request to glidersupport@teledyne.com

Transporting the glider

1. Ensure that all of the cart straps, crate straps, and locks are used, or load the glider into the boat and proceed to the first waypoint or deployment location. See the sections, "Deploying the Glider" on page -12 and "Recovering the Glider" on page -15. Just prior to deployment install wings and ensure green plug is seated well and tucked into cowling

In the Water

1. Attach a 10 meter line, preferably neutrally buoyant line, with a buoy to the glider before putting it in the water. If you have **great confidence** in the glider's ballasting you may choose to not test on the line.
2. Once the glider is in the water, type `run status.mi` again.
3. Run ini0.mi per below. If further evaluation is required run one or more of the following missions while on station until satisfied that the glider is ballasted and operating normally.
   - `run ini0.mi`—Does a single yo to a maximum depth of 3 meters and a minimum depth of 1.5 meters. This uses a fixed pitch battery and fin position.
   - `run ini1.mi`—Does 3 yos to the north, diving to 5 meters and climbing to 3 meters. The pitch should be +/- 20 degrees.
   - `run ini2.mi`—Goes to a waypoint 100 meters south of the dive point, diving to 5 meters and climbing to 3 meters. The pitch should be +/- 20 degrees.
   - `run ini3.mi`—Goes to a waypoint 100 meters north of the dive point, diving to 5 meters and climbing to 3 meters. The pitch should be +/- 20 degrees.
4. Send the files locally and/or by Iridium. Confirm the flight data and desired flight characteristics of ini missions run. If necessary, turn flight control over to the Dockserver over Iridium.
5. If you have not removed the buoy and the line from the glider, do so now.
6. From the GliderDOS prompt, type `exit reset`. This forces reinitialization of all of the sensor values.

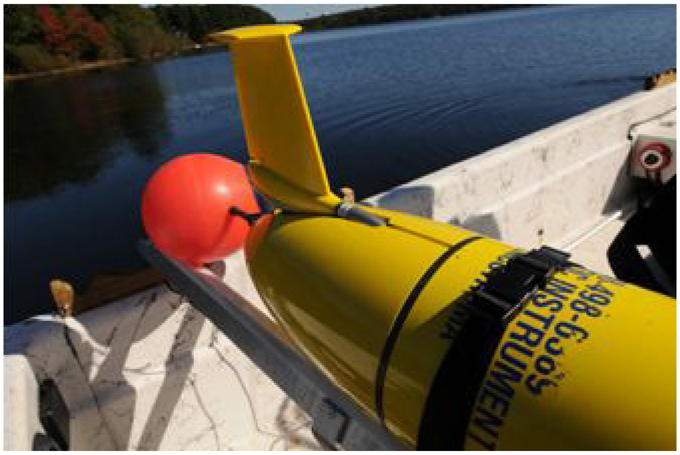
7. When the glider reboots, type `control-C`, when prompted, to bring up the GliderDOS prompt. Then type `loadmission waterclr.mi` to zero any built-up water currents that are remembered long term.
8. Type `run stock.mi` or the equivalent `.mi` to begin the desired mission.

Deploying the Glider

Deployment at sea can be dangerous, and the welfare of crew and glider handlers should be considered while at the rail of a ship. From a small boat the glider cart can be used to let the glider slip easily into the water. Remove the nose ring quick pin and let the ring fall forward, if necessary entirely remove nose ring. Unclip the fastening strap from the center section of the cart/glider before deployment.

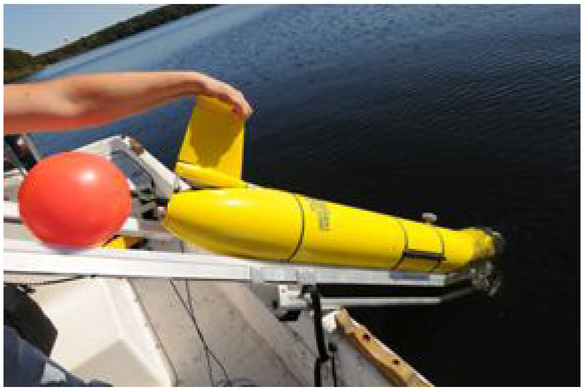
For larger boats, the pick point affixed to the payload bay can be used to lower and raise the glider with a crane or winch from the vessel to the water.

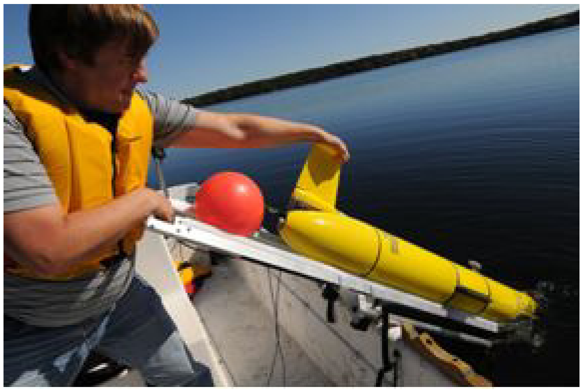
Glider with the Buoy and Rope Ready for the First Deployment

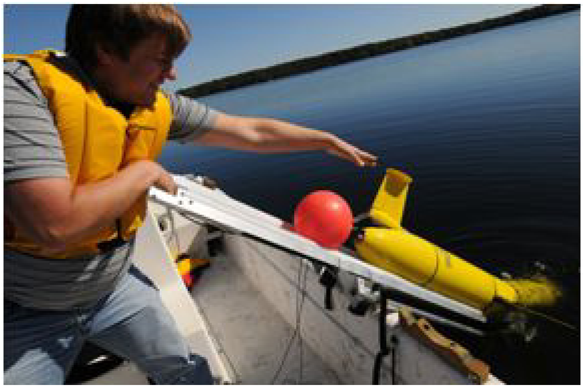


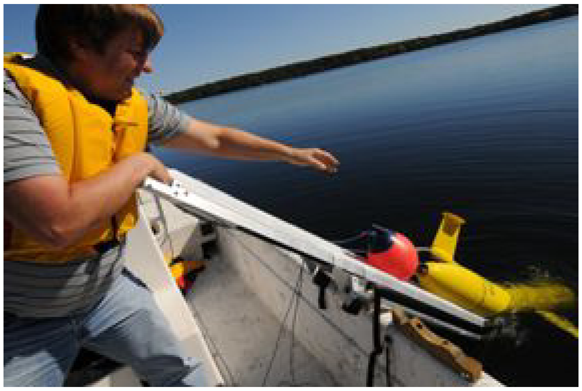
NOTE In the deployment sequence below, the digifin can be handled.

CAUTION Sensors protruding from the vehicle are prone to damage during deployment and recover. Take steps to ensure no damage is done.



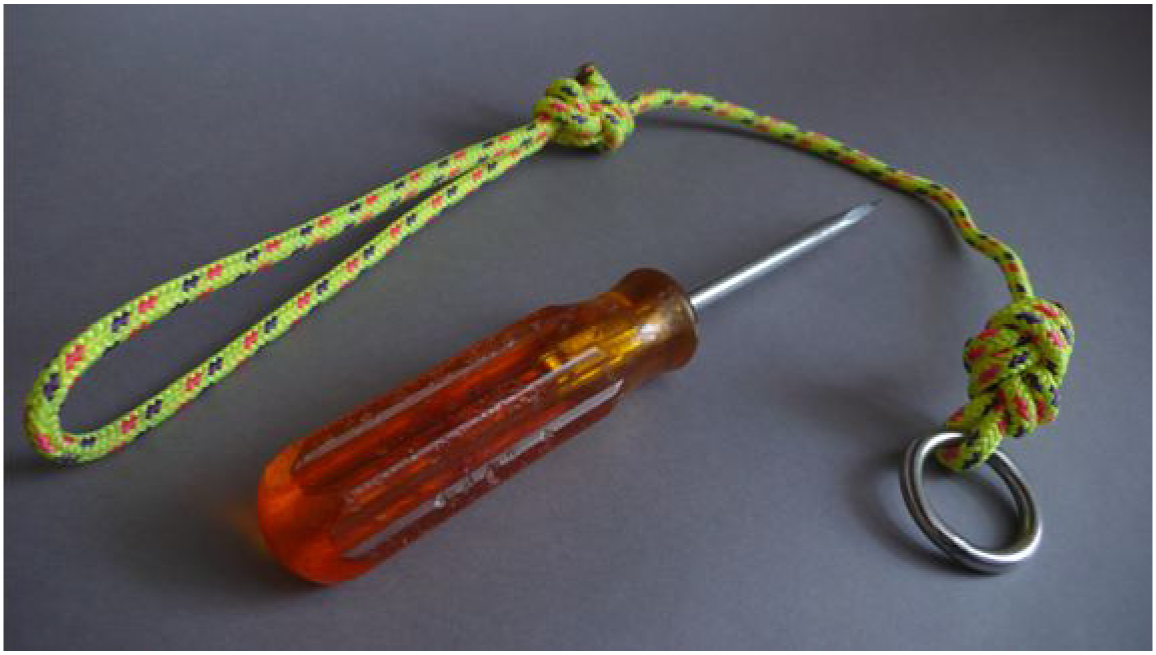


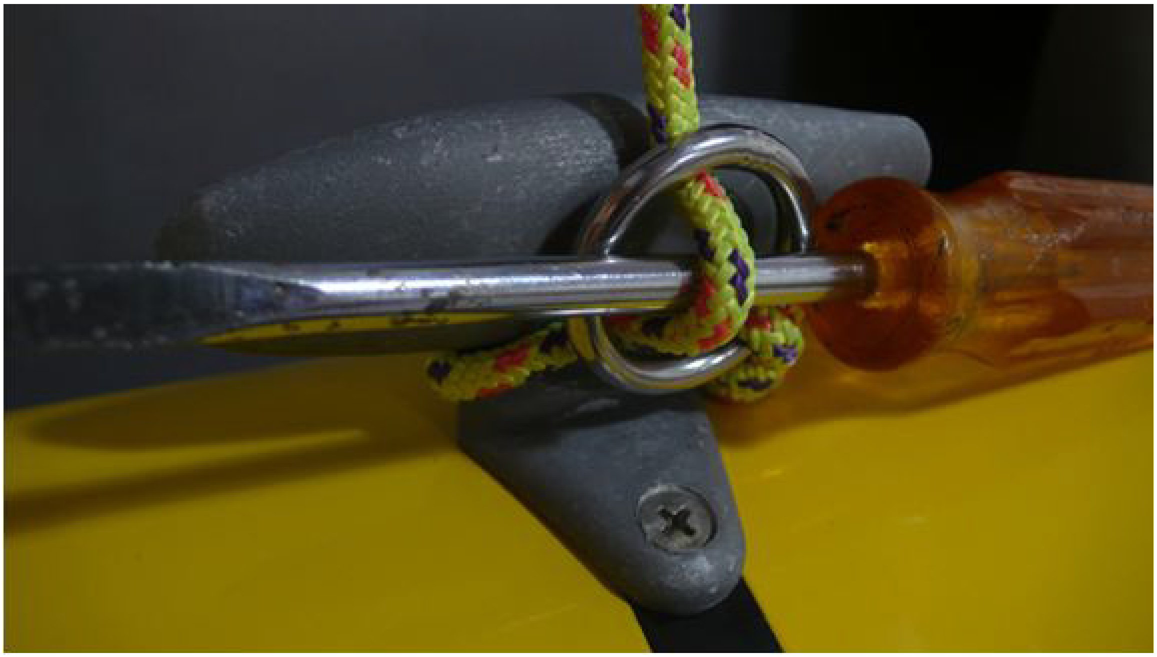




Large Ship Deployment

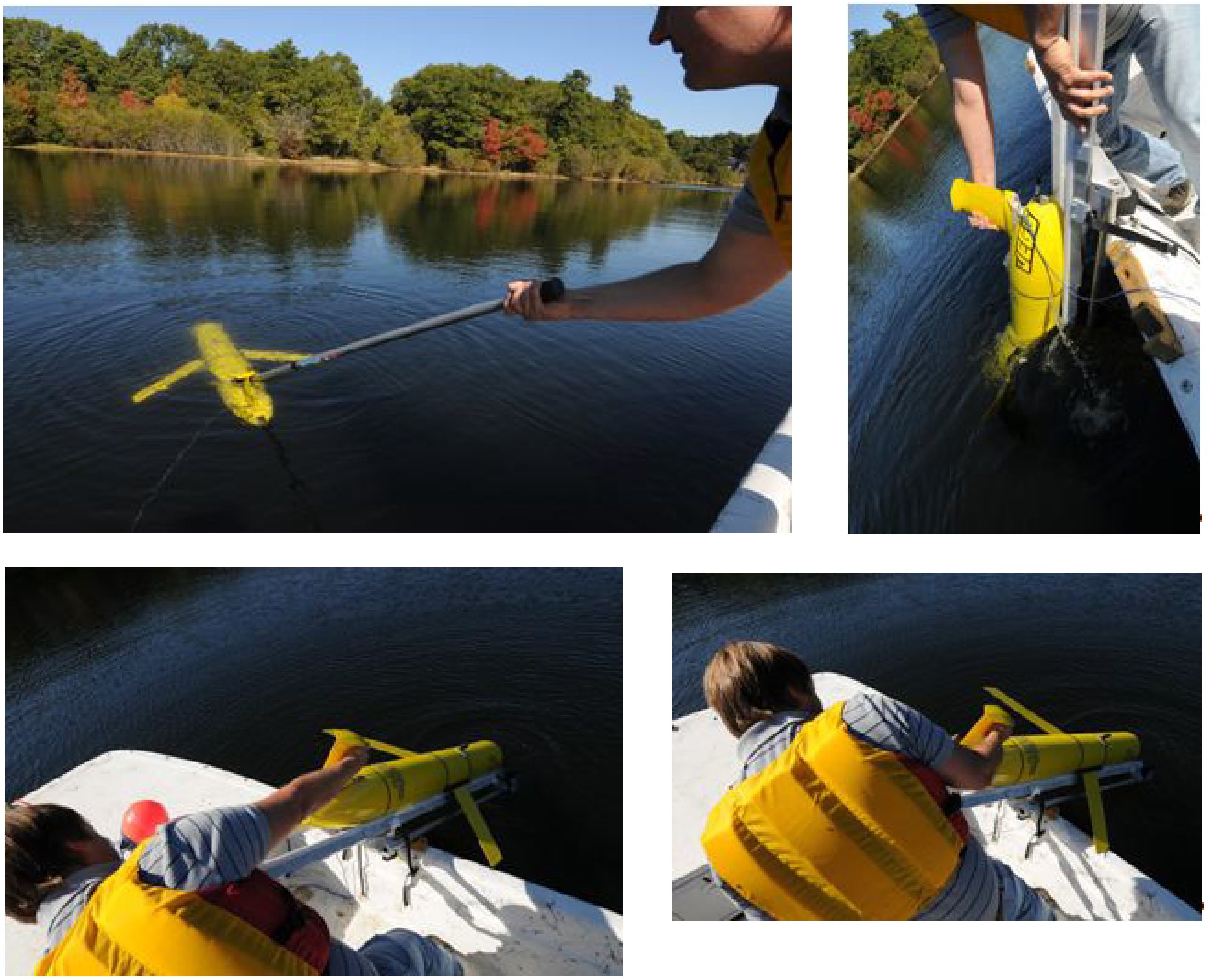
A quick release system using the pick point can be fashioned from supplies found on most vessels, as illustrated in the following two images.





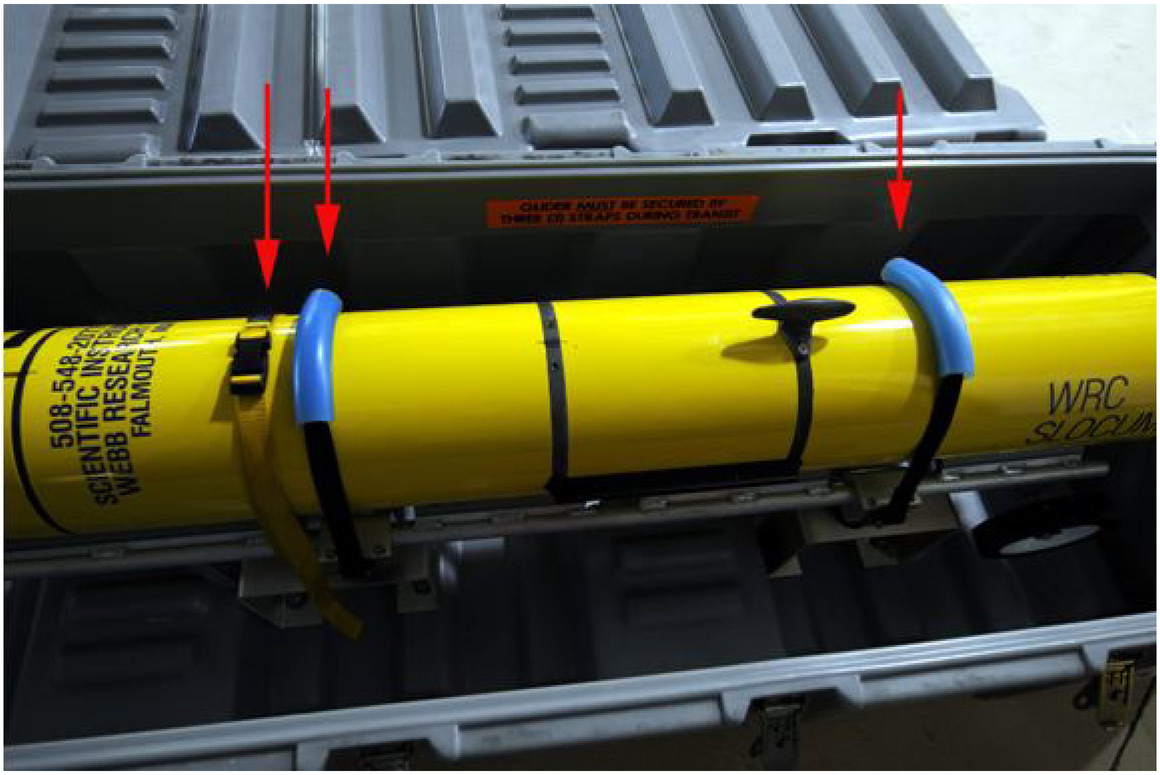
Recovering the Glider

NOTE A boat hook can be used to manipulate the glider in the water.



Lower the cart into water, and manipulate the glider by the digifin into position on the cart. Lift and tilt the glider onto the ship’s deck.

Packing the Glider



Ensure that all three straps are secure (two crate straps and one cart strap). If extra supplies are included in the crate, ensure that they will not interfere with the fin or become dislodged during transit.

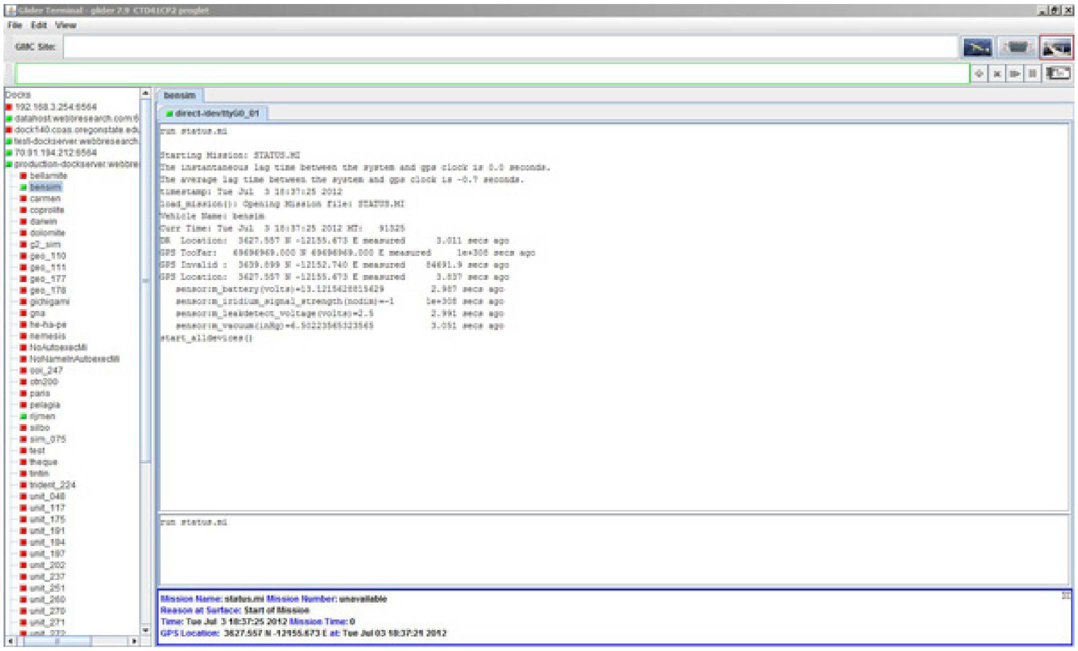
Dockserver

Dockserver is the name of the laptop or rack-mounted Linux CentOS 6 or Redhat 6 based PC provided with a glider. The applications (also named Dockserver and Dataserver) must be launched from desktop icons to provide full Dockserver functionality.

Glider Terminal

This is the primary interface through the Dockserver to the glider.

- Top panel
  - Dockserver site manual entry
  - Script functionality
  - Terminal and ports perspective toggle and remote glider notification tabs.
- Left panel: active docks and gliders
- Middle right panel: communication from the glider
- Bottom right panel: communication to the glider
- Optional bottom right panel: mission status



```
Docks
192.168.3.254:6564
datahost.webbresearch.com:6
dock140.coas.oregonstate.edu
test-dockserver.webbresearch
70.91.194.212:6564
production-dockserver.webbre
    bellamite
    bensim
    carmen
    coprolite
    darwin
    dolomite
    g2_sim
    geo_110
    geo_111
    geo_177
    geo_178
    gichigami
    gna
    he-ha-pe
    nemesis
    NoAutoexecMi
    NoNameInAutoexecMi
    ooi_247
    otn200
    paris
    pelagia
    rijmen
    silbo
    sim_075
    test
    theque
    tintin
    trident_224
    unit_048
    unit_117
    unit_175
    unit_191
    unit_194
    unit_197
    unit_202
    unit_237
    unit_251
    unit_260
    unit_270
    unit_271
    unit_272
```

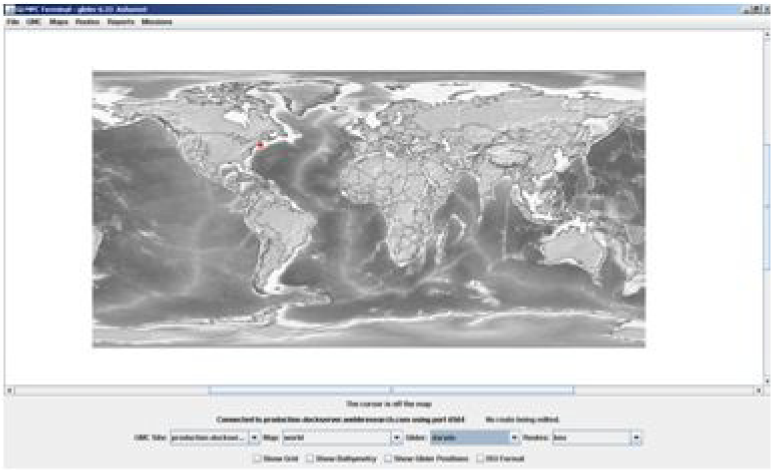
```
run status.mi

Starting Mission: STATUS.MI
The instantaneous lag time between the system and gps clock is 0.0 seconds.
The average lag time between the system and gps clock is -0.7 seconds.
timestamp: Tue Jul  3 18:37:25 2012
load_mission(): Opening Mission file: STATUS.MI
Vehicle Name: bensim
Curr Time: Tue Jul  3 18:37:25 2012 MT:    91525
DR  Location:  3627.557 N -12155.473 E measured      3.011 secs ago
GPS TooFar:   69696969.000 N 69696969.000 E measured     1e+308 secs ago
GPS Invalid :  3639.899 N -12152.740 E measured    84691.9 secs ago
GPS Location:  3627.557 N -12155.473 E measured      3.037 secs ago
   sensor:m_battery(volts)=13.1215628815429        2.987 secs ago
   sensor:m_iridium_signal_strength(nodim)=-1      1e+308 secs ago
   sensor:m_leakdetect_voltage(volts)=2.5          2.991 secs ago
   sensor:m_vacuum(inHg)=6.50223565323565          3.051 secs ago
start_alldevices()
```

```
run status.mi
```

Mission Name: status.mi Mission Number: unavailable
Reason at Surface: Start of Mission
Time: Tue Jul 3 18:37:25 2012 Mission Time: 0
GPS Location: 3627.557 N -12155.673 E at: Tue Jul 03 18:37:25 2012

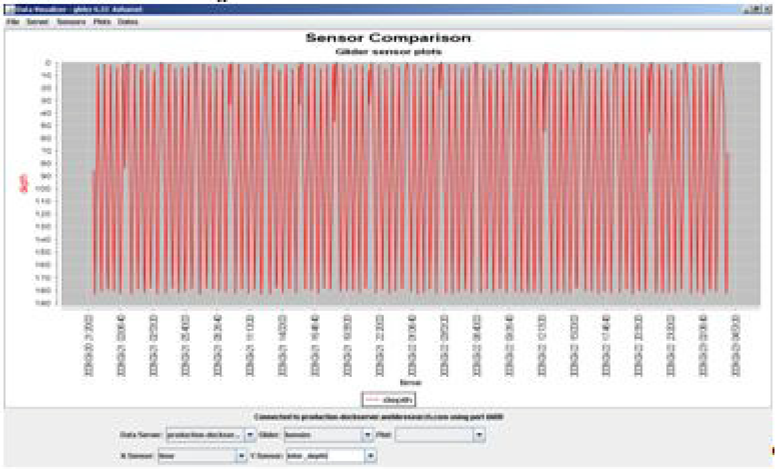
Glmpc Terminal

This real-time interface displays custom maps in JPG format and allows click-through uploading of waypoints during live missions.


File GMC Maps Routes Reports Missions
The cursor is off the map
Show Grid
Show Bathymetry
Show Glider Positions
ISO Format

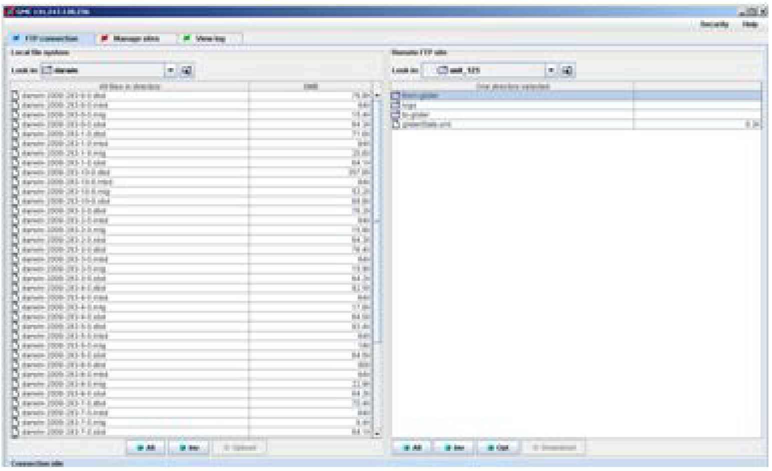
Data Visualizer

The Data Visualizer allows pilots to plot all glider data as it is received by the Dockserver. The Data Visualizer server must be running on Dockserver to view data remotely. Launch the Data Visualizer with the desktop icon on the Dockserver.



Dockserver FTP Utility

Whenever new files are sent to Dockserver, you must disconnect and reconnect to refresh the file list.



Configure Communications with the Terminal Program (ProComm Plus)

Many users have decided to have a mobile Dockserver and a permanent installation Dockserver. If you do not have a mobile Dockserver, the following settings in the ProComm Plus Terminal program will allow direct communications with a terminal program to the glider.

1. Connect the powered FreeWave to a serial comm port on computer with the serial cable provided.
2. Open ProComm Plus.
3. Select the proper com port:
    - Baud 115200
    - Parity N-8-1
4. Go to **Options > System Options > Modem Connection**.
5. Click on **Modem Connection Properties**.
6. If the **Use hardware flow control check box** is unchecked, check it and click **OK**.
7. Click on the **Data** tab.
8. Next to **Receiver Crash Recovery Settings**, click **Change Settings**.
9. Check **If date/time match** under **Crash Recovery Options**.
10. Check **Overwrite if incoming newer** under **Overwrite Options**.
11. Click **OK**.
12. Next to **Sender Crash Recovery Settings**, click **Change Settings**.
13. Check **Crash recovery off** under **Crash Recovery Options**.
14. Check **Always overwrite** under **Overwrite Options**.
15. Click **OK**.
16. Select **Streaming from the Transmit method** menu and uncheck **Use local EOL convention**.
17. Select **32 bit CRC** from the **Error detection** menu, and check **Original file time stamp**.

18. Click **OK**. You are now ready to begin communications with the glider and conduct ZR/ZS testing.

NOTE There are known problems with using HyperTerminal and attempting to ZR/ZS. TeraTerm is another viable terminal program (see http://www.ayera.com/teraterm/).

Surface Dialog

An example of a surface dialog is shown below. Note the available glider commands are in bold.

```
Glider bensim at surface.
Because:Hit a waypoint [behavior surface_2 start_when = 8.0]
MissionName:initial.mi MissionNum:bensim-2010-123-2-0
(0103.0000)
Vehicle Name: bensim
Curr Time: Tue May 4 13:25:20 2010 MT:     316
DR Location: 3342.801 N -11824.540 E measured 1.487 secs ago
GPS TooFar: 69696969.000 N 69696969.000 E measured 1e+308 secs
ago
GPS Invalid: 3342.832 N -11824.533 E measured 252.734 secs ago
GPS Location: 3342.801 N -11824.540 E measured 3.994 secs ago
   sensor:m_battery(volts)=13.121562938211 3.926 secs ago
   sensor:m_iridium_signal_strength(nodim)=-1 1e+308 secs ago
   sensor:m_leakdetect_voltage(volts)=2.5 3.921 secs ago
   sensor:m_vacuum(inHg)=6.50223565323565 8.214 secs ago
devices:(t/m/s) errs: 0/ 0/ 0 warn: 0/ 0/ 0 odd: 0/ 0/ 0
ABORT HISTORY: total since reset: 0
```

- Press **Control-R** to resume the mission, i.e. dive!
- Press **Control-C** to end the mission, i.e. GliderDOS.
- Press **Control-E** to extend the surface time by five minutes.
- Press **Control-W** to get device warning reports.
- Press **Control-F** to re-read yo, goto_l, sample, drift_ mafiles.
- Press **S [-f={rf}|{irid] [-num=<n>] [-t=<s>] [filespec ...]** to send log files.
- Press **! <GliderDos cmd>** to execute <GliderDos cmd>.
- Press **Control-T** to consci to science computer when comms ready:
  ...communications ***not*** ready for consci.
  ... because: sci_m_science_on = 0

```
Water Velocity Calculations COMPLETE
Waypoint: (3342.8323,-11824.5333) Range: 58m, Bearing: 11deg,
Age: -1:-1h:m
Drifting toward outer watch circle, centered on waypoint
Now 58.3 meters from middle, will dive at 100.0 meters
Time until diving is: 150 secs(estimated)
```

File Manipulation

Transferring Files to and from the Glider

Commands for transferring files are named from the perspective of the glider. The send command sends data files from the glider. The zr (receive) command allows the glider to receive files. If FreeWave and Iridium connections are both present, files are sent over FreeWave. A pilot should never use the *.* wildcard when FreeWave communication is not present.

Transferring Files from the Glider

While in GliderDOS, use the `send` command to transfer data files from the glider to the Dockserver or to a computer running a terminal emulator. The `send *.*` command sends the 30 most recent data files (of type .sbd, .mbd, .dbd, .mlg, .tbd, .nbd, .ebd, .nlg, and sys.log). The `send` command can also be used to send specific data files or files with specific extensions. For example, `send 12345678.ebd` will send the file named 12345678.ebd, and `send *.nlg` will send the 30 most recent .nlg files from the glider.

During a mission, the `s` command is used in place of the `send` command. The syntax of these two commands is identical; during a mission, `s 12345678.ebd` will send the file 12345678.ebd from the glider.

In order to send non-data files from the glider while in GliderDOS, the `zs` command is used instead of the `send` command, and during a mission the command is prepended by an exclamation point (referred to as a “bang”). For example, to send the autoexec.mi file from a glider in GliderDOS, type `zs \config\autoexec.mi`; to send the longterm.sta file from the glider during a mission, type `!zs \state\longterm.sta`.

Transferring Files to the Glider

Sending files to the glider requires the use of either the `zr` or `dockzr` command. While the glider is in GliderDOS, pilots operating via a terminal emulator will use the `zr` command and pilots operating via Dockserver will use the `dockzr` command. During a mission, an exclamation point (referred to as a “bang”) is prepended to the command. Pilots using the `zr` command from a terminal emulator are required to include the file path and filename. For example, while in a mission a pilot using a terminal emulator would type `!zr <path>\<filename>` to send files to the glider. With the glider in GliderDOS, a pilot using Dockserver would type `dockzr autoexec.mi` to send a new autoexec.mi file to the glider. When using Dockserver, the desired file or files must be in the glider's directory on the Dockserver. It is not necessary to specify the target file path when using this command because the glider will sort the files into their respective directories automatically.

Glider Dos and Don’ts

Dos

Always do these things with a glider:

- Secure it properly for shipping with the cart strap and two crate straps.
- Use fresh desiccants on each deployment.
- Monitor internal vacuum before launch (less vacuum indicates a leak; positive pressure may indicate dangerous gas accumulation).
- Simulate missions before launch.
- Test Iridium and Argos telemetry before launch.

Don’ts

Never do these things with a glider:

- Never power up a shallow glider without a vacuum.
- Never run a simulation on a glider other than “on_bench.”
- Never deploy a glider in simulation.
- Never deploy a glider in “boot pico.”
- Never exit to pico during a deployment.
- Never power on a glider with more than 15 vDC from an external power supply.
- Never deploy a glider in `lab_mode`.
- Never perform the top of a yo below 30 meters with a shallow glider.
- Never secure the glider to the glider cart while over railing or in the water.

Mission Files (.mi and .ma)

For the default Webb Ashumet missions below, insert text of actual missions and .ma files here if desired. Sensors and arguments commonly changed by users are indicated in blue text.

stock.mi

```
## stock.mi
#
#  Retrieves waypoints from mafiles/goto_l10.ma
#  Retrieves yo envelope  from mafiles/yo10.ma
#  Retrieves climb to surface controls from mafiles/surfac01 through 06.ma
#  Surfaces:
#     if haven't had comms as controlled by surfac10.ma
#     mission done (finished all the waypoints) surfac11.ma
#     Bad altimeter or half yo's finish surfac.12.ma
#     Every waypoint surfac13.ma
#     If requested by science surfac14.ma
#     Every x minutes Surfac15.mi
# All science sensors sample on only downcast from sample10.ma
#
# 10-July-2010 ballsup@webbresearch.com Initial (based on glmpc.mi)
# 30-Nov-2010  ballsup@webbresearch.comchanged abort for cop tickle 13.5 hours and disabled
percentage method
#
################################################################################
sensor: c_science_all_on_enabled(bool)   1  # in, non-zero enables c_science_all_on
    # disable this sensor to allow for individually
    # sampled science sensors ie sample11.ma and greater

sensor: u_use_ctd_depth_for_flying(bool) 0  # true=> use ctd measurement for m_depth
                                            # implemented as emergency workaround for
                                            # broken ocean pressure

sensor: u_use_current_correction(nodim)  1  #  0 calculate, but do not use m_water_vx/y
                                            #  1 use m_water_vx/y to navigate AND aim


################################################################################

behavior: abend
    b_arg: overdepth_sample_time(s)       15.0 # how often to check
                                               # MS_ABORT_OVERTIME
    b_arg: overtime(s)                    -1.0 # < 0 disables
                                               # MS_ABORT_WPT_TOOFAR
```

```
   b_arg: max_wpt_distance(m)         -1   # Maximum allowable distance to a waypoint
    b_arg: samedepth_for_sample_time(s)   30.0 # how often to check
    b_arg: undervolts(volts)              10.0 # < 0 disables Decrease to 9
                                               # for Lithium primary batteries

    b_arg: no_cop_tickle_for(sec)      48600.0 # secs, abort mission if watchdog
                                               # not tickled this often, <0 disables
    b_arg: no_cop_tickle_percent(%)         -1 # 0-100, <0 disables

################################################################################

# Come up if haven't had comms for a while
behavior: surface
    b_arg: args_from_file(enum) 01 # read from mafiles/surfac01.ma

################################################################################

    # Come up when mission done
    # This is determined by no one steering in x-y plane (no waypoints)
behavior: surface
    b_arg: args_from_file(enum) 02 # read from mafiles/surfac02.ma

################################################################################

    # Come up briefly if "yo" finishes
    # This happens if a bad altimeter hit causes a dive and climb to
    # complete in same cycle.  We surface and hopefully yo restarts
    # or change keystroke_wait_time if surfacing for num_half_cycles_to_do
behavior: surface
    b_arg: args_from_file(enum) 03 # read from mafiles/surfac03.ma

################################################################################

    # Come up every way point
behavior: surface
    b_arg: args_from_file(enum) 04 # read from mafiles/surfac04.ma

################################################################################

    # Come up when requested by science
behavior: surface
    b_arg: args_from_file(enum) 05 # read from mafiles/surfac05.ma


################################################################################

    # Come up every x minutes
```

```
behavior: surface
    b_arg: args_from_file(enum) 06 # read from mafiles/surfac06.ma


################################################################################

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

################################################################################

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

################################################################################

   # Sample all science sensors only on downcast
behavior: sample
b_arg: args_from_file(enum)            10  # >= 0 enables reading from mafiles/sample10.ma

################################################################################

   # Sample ctd only on downcast
   # sensor c_science_all_on_enabled must be set to 0 to uncouple science sensor union
#behavior: sample
#b_arg: args_from_file(enum)            11  # >= 0 enables reading from mafiles/sample11.ma

################################################################################

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

################################################################################

behavior: sensors_in       # Turn most input sensors off
```

surfac01.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma

# Come up if haven't had comms for a while, 20 minutes

<start:b_arg>

    b_arg: start_when(enum)         12            # BAW_NOCOMM_SECS 12, when have not had
comms for WHEN_SECS secs

    b_arg: when_secs(sec)           1200          # Surface every 1hr for no comms

    b_arg: end_action(enum)         1             # 0-quit, 1 wait for ^C quit/resume, 2
resume, 3 drift til "end_wpt_dist"
    b_arg: gps_wait_time(s)         300           # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 300           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg
    b_arg: printout_cycle_time(sec) 60.0          # How often to print dialog

<end:b_arg>
```

surfac02.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma

#  Come up when mission done
#  This is determined by no one steering in x-y plane (no waypoints)

<start:b_arg>

   b_arg: start_when(enum)         3            # 0-immediately, 1-stack idle 2-pitch idle
3-heading idle
    b_arg: end_action(enum)         0             # 0-quit, 1 wait for ^C quit/resume, 2
resume, 3 drift til "end_wpt_dist"
    b_arg: gps_wait_time(s)         300           # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 180           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg

<end:b_arg>
```

surfac03.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma

# Come up briefly if "yo" finishes
# This happens if a bad altimeter hit causes a dive and climb to
# complete in same cycle.  We surface and hopefully yo restarts

# or change keystroke_wait_time if surfacing for num_half_cycles_to_do

<start:b_arg>

    b_arg: start_when(enum)        2             # 0-immediately, 1-stack idle 2-pitch idle
3-heading idle
    b_arg: end_action(enum)         1             # 0-quit, 1 wait for ^C quit/resume, 2
resume, 3 drift til "end_wpt_dist"
    b_arg: gps_wait_time(s)         15            # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 180           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg

<end:b_arg>
```

surfac04.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma
# Come up every way point

<start:b_arg>

    b_arg: start_when(enum)         8             # 8-when hit waypoint
    b_arg: end_action(enum)         1             # 0-quit, 1 wait for ^C quit/resume, 2
resume, 3 drift til "end_wpt_dist"
    b_arg: gps_wait_time(s)         300           # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 300           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg
    b_arg: printout_cycle_time(sec) 60.0          # How often to print dialog

<end:b_arg>
```

surfac05.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma

# Come up when requested by science

<start:b_arg>

    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, 3 drift til "end_wpt_dist"
    b_arg: gps_wait_time(s)         300           # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 300           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg

<end:b_arg>
```

surfac06.ma

```
behavior_name=surface
# climb to surface with ballast pump full out
# pitch servo'ed to 26 degrees
# Hand Written
# 10 July 2010 ballsup@webbresearch.com based on legacy surfac10.ma

# Come up every three hours

<start:b_arg>

    b_arg: start_when(enum)         9             # 9-every when_secs
    b_arg: end_action(enum)         1             # 0-quit, 1 wait for ^C quit/resume, 2
resume, 3 drift til "end_wpt_dist"

    b_arg: when_secs(s)             10800           # How long between surfacing, only if
start_when==6 or 9

    b_arg: gps_wait_time(s)         300           # how long to wait for gps
    b_arg: keystroke_wait_time(sec) 300           # how long to wait for control-C
    b_arg: when_wpt_dist(m)         10            # how close to waypoint before surface,
only if start_when==7
    b_arg: c_use_pitch(enum)        3             # 3:servo
    b_arg: c_pitch_value(X)         0.4528        # 26 deg
    b_arg: printout_cycle_time(sec) 60.0          # How often to print dialog

<end:b_arg>
```

goto10.ma

goto10.ma

```
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
#  ??-Apr-03 kniewiad@webbresearch.com changed to ashument
#  17-Apr-03 tc@DinkumSoftware.com fixed comments
# goto_l10.ma
# Flies the box in ashumet
# Each leg about 200m<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) 4
<end:b_arg>
<start:waypoints>
-7032.0640   4138.1060
-7031.9200   4138.1090
-7031.9170   4138.0000
-7032.0610   4137.9980
<end:waypoints>
```

yo10ma

yo10ma.

```
behavior_name=yo
# yo10.ma
# climb 3m   dive 12m alt 9m pitch 26 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)
# 09-Apr-03 kniewiad@webbresearch.com Adjusted for Ashumet
<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)     12
    b_arg: d_target_altitude(m)   3
    b_arg: d_use_pitch(enum)      3   # 1:battpos  2:setonce  3:servo
                                      #   in         rad        rad, <0 dive
    b_arg: d_pitch_value(X)   -0.4528     # -26 deg

    # arguments for climb_to
    b_arg: c_target_depth(m)      3
    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)     0.4538     # 26 deg
    b_arg: end_action(enum) 2     # 0-quit, 2 resume
<end:b_arg>

# NOTE: These are symbolically defined beh_args.h
# b_arg: START_WHEN     When the behavior should start, i.e. go from UNITIALIZED to ACTIVE
#    BAW_IMMEDIATELY    0   // immediately
#    BAW_STK_IDLE       1   // When stack is idle (nothing is being commanded)
#    BAW_PITCH_IDLE     2   // When pitch is idle(nothing is being commanded)
#    BAW_HEADING_IDLE   3   // When heading is idle(nothing is being commanded)
#    BAW_UPDWN_IDLE     4   // When bpump/threng is idle(nothing is being commanded)
#    BAW_NEVER          5   // Never stop
#    BAW_WHEN_SECS      6   // After behavior arg "when_secs", from prior END if cycling
#    BAW_WHEN_WPT_DIST  7   // When sensor(m_dist_to_wpt) < behavior arg "when_wpt_dist"
#    BAW_WHEN_HIT_WAYPOINT 8 // When X_HIT_A_WAYPOINT is set by goto_wpt behavior
#    BAW_EVERY_SECS     9   // After behavior arg "when_secs", from prior START if cycling
#    BAW_EVERY_SECS_UPDWN_IDLE 10   // After behavior arg "when_secs", from prior START AND
#                                 //        updown is idle, no one commanding vertical motion
#    BAW_SCI_SURFACE    11  // SCI_WANTS_SURFACE is non-zero
#    BAW_NOCOMM_SECS    12  // when have not had comms for WHEN_SECS secs
# b_arg: STOP_WHEN
#   0   complete
#   1-N same as "start_when"
```

sample10.ma

```
behavior_name=sample
# sample all science sensors on down cast only
# 10-July-2010 ballsup@webbresearch.com handwritten for stock.mi

<start:b_arg>
   b_arg: sensor_type(enum)                 0 # ALL    0  C_SCIENCE_ALL_ON
                                              # PROFILE     1  C_PROFILE_ON
                                              # HS2    2  C_HS2_ON
                                              # BB2F    3  C_BB2F_ON
                                              # BB2C    4  C_BB2C_ON
                                              # BB2LSS      5  C_BB2LSS_ON
                                              # SAM    6  C_SAM_ON
                                              # WHPAR       7  C_WHPAR_ON
                                              # WHGPBM      8  C_WHGPBM_ON
                                              # MOTEBB      9  C_MOTEBB_ON
                                              # BBFL2S     10  C_BBFL2S_ON
                                              # FL3SLO     11  C_FL3SLO_ON
                                              # BB3SLO     12  C_BB3SLO_ON
                                              # OXY3835    13  C_OXY3835_ON
                                              # WHFCTD     14  C_WHFCTD_ON
                                              # BAM        15  C_BAM_ON
                                              # OCR504R    16  C_OCR504R_ON
                                              # OCR504I    17  C_OCR504I_ON
                                              # BADD       18  C_BADD_ON
                                              # FLNTU      19  C_FLNTU_ON
                                              # FL3SLOV2   20  C_FL3SLOV2_ON
                                              # BB3SLOV2   21  C_BB3SLOV2_ON
                                              # OCR507R    22  C_OCR507R_ON
                                              # OCR507I    23  C_OCR507I_ON
                                              # BB3SLOV3   24  C_BB3SLOV3_ON
                                              # BB2FLS     25  C_BB2FLS_ON
                                              # BB2FLSV2   26  C_BB2FLSV2_ON
                                              # OXY3835_WPHASE 27 C_OXY3835_WPHASE_ON
                                              # AUVB       28  C_AUVB_ON
                                              # BB2FV2     29  C_BB2FV2_ON
                                              # TARR       30  C_TARR_ON
                                              # BBFL2SV2   31  C_BBFL2SV2_ON
                                              # GLBPS      32  C_GLBPS_ON
                                              # SSCSD      33  C_SSCSD_ON
                                              # BB2FLSV3   34  C_BB2FLSV3_ON
                                              # FIRE       35  C_FIRE_ON
                                              # OHF        36  C_OHF_ON
                                              # BB2FLSV4   37  C_BB2FLSV4_ON
                                              # BB2FLSV5   38  C_BB2FLSV5_ON
                                              # LOGGER     39  C_LOGGER_ON
                                              # BBAM       40  C_BBAM_ON
                                              # UMODEM     41  C_UMODEM_ON
                                              # RINKOII    42  C_RINKOII_ON
                                              # DVL        43  C_DVL_ON
                                              # BB2FLSV6   44  C_BB2FLSV6_ON

                                                # This is a bit-field, combine:
                                            # 8 on_surface, 4 climbing, 2 hovering, 1 diving
    b_arg: state_to_sample(enum)             1  # 0  none
                                                # 1  diving
                                                # 2  hovering
                                                # 3  diving|hovering
                                                # 4  climbing
                                                # 5  diving|climbing
                                                # 6  hovering|climbing
                                                # 7  diving|hovering|climbing
                                                # 8  on_surface
                                                # 9  diving|on_surface
                                                # 10 hovering|on_surface
                                                # 11 diving|hovering|on_surface
                                                # 12 climbing|on_surface
                                                # 13 diving|climbing|on_surface
                                                # 14 hovering|climbing|on_surface
                                                # 15 diving|hovering|climbing|on_surface

    b_arg: intersample_time(s)                0  # if < 0 then off, if = 0 then
                                                 # as fast as possible, and if
                                                 # > 0 then that many seconds
                                                 # between measurements

    b_arg: nth_yo_to_sample(nodim)            1  # After the first yo, sample only
                                                 # on every nth yo. If argument is
                                                 # negative then exclude first yo.

    b_arg: intersample_depth(m)              -1  # supersedes intersample_time
                                                 # by dynamically estimating
                                                 # and setting intersample_time
                                                 # to sample at the specified
                                                 # depth interval. If <=0 then
                                                 # then sample uses
                                                 # intersample_time, if > 0 then
                                                 # that many meters between
                                                 # measurements

    b_arg: min_depth(m)                      -5  # minimum depth to collect data, default
                                                 # is negative to leave on at surface in
                                                 # spite of noise in depth reading
    b_arg: max_depth(m)                    2000  # maximum depth to collect data

<end:b_ar
```

Appendix A: Commonly Used Glider Commands

To list all commands that are available, type `help` from a GliderDOS prompt. These commands are also listed in the table below. For examples of how commands are used, see the “Sample Mission and Comments” section in Appendix A of the *Slocum G2 Glider Operators Manual*.

Command Name	Syntax and/or Description
`attrib`	ATTRIB [+ - RASH] [d:][p][name]
`ballast`	BALLAST ?; for help
`boot`	boot [PICO][PBM][APP]
`callback`	callback <minutes til callback>
`capture`	capture [d:][p]fn [/Dx/B/N/E]
`cd`	Change directory
`chkdsk`	CHKDSK [d:][p][fn] [/F][/I] *
`clrdeverrs`	Zero device errors
`consci`	consci [-f rf|irid]; console to science
`copy`	copy source dest [/V]
`cp`	CP <src_path> <dest_path>; copies a file system branch
`crc`	Computes CRC on memory
`date`	DATE [mdy[hms[a|p]]] /IEUMCP]
`dellog`	DELLOG ALL MLG DBD SBD
`del`	DEL [drv:][pth][name] [/P]
`devices?`	Prints device driver information
`df`	Prints disk space used and disk space free
`dir`	DIR [d:][p][fn] [/PWBLV4A:a]
`dump`	DUMP file[start[,end]] *
`erase`	ERASE [drv:][pth][name] [/P] *

Command Name	Syntax and/or Description
`exit`	exit [-nofin] [poweroff|reset|pico|pbm]
`get`	GET <sensor name>
`hardware?`	HARDWARE? [-v]; hardware configuration
`heap`	Reports free memory
`help`	Prints help for commands
`highdensity`	HIGHDENSITY ?; for help
`lab_mode`	LAB_MODE [on|off]
`list`	Displays all sensor names
`loadmission`	Loads mission file
`logging`	logging on|off; during GliderDOS
`longterm_put`	LONGTERM_PUT <sensor name> <new value>
`longterm`	LONGTERM ?; for help
`ls`	LS [path] ; list a file system branch
`mbd`	MBD ?; for help
`mkdir`	MKDIR [drive:][path]
`mv`	MV <src_path> <dest_path>; copy a file system branch
`path`	PATH Show search path * PATH [[d:]path[;...]] [/P] *
`prompt`	prompt [text] [/P] *
`prunedisk`	Prunes expendable files to free space on disk
`purgelogs`	Deletes sent log files
`put`	PUT <sensor name> <value>
`rename`	RENAME [d:][p]oldname newname
`report`	REPORT ?; for help
`rmdir`	RMDIR [drive:][path]
`rm`	RM <path>; deletes a file system branch *
`run`	run [mission_file]; runs the mission file
`sbd`	SBD ?; ? for help

Command Name	Syntax and/or Description
`send`	SEND [-f={rf}|{irid] [-num=<n>] [-t=<s>] [filespec ...]
`sequence`	SEQUENCE ?; do this for help
`setdevlimit`	SETDEVLIMIT devicename os w/s w/m
`setnumwarn`	SETNUMWARN [X]; sets max dev warnings to X
`set`	SET [var=[str]] [/SLFE?] *
`simul?`	Displays a print description of what is simulated
`srf_display`	SRF_DISPLAY ?; for help
`sync_time`	sync_time [offset]; syncs system time with gps time
`tcm3`	TCM3 ?; for help
`time`	TIME [hh:mm:ss [a|p]] [/M/C]
`tvalve`	tvalve [up|charge|down][backward] *
`type`	TYPE [drv:][pth][name]
`use`	USE ?; do this for help
`ver`	Displays firmware versions
`where`	Prints latitude/longitude
`whoru`	whoru Vehicle Name:; displays vehicle name
`why?`	WHY? [abort#]; displays the reason for an abort
`wiggle`	wiggle [on|off] [fraction]; moves motor
`zero_ocean_pressure`	Recalibrates zero ocean pressure
`zr`	Zmodem Rec: zr ? for help
`zs`	Zmodem Send: zs ? for help

** not often used by average user*

Appendix B: Worksheets

This appendix contains the worksheets you will use frequently in glider operations.

Glider Operations Mission Planning Overview Worksheet

Glider number		**Prepared by**	
Payload instruments			
Deployment location	**Surf temp**	**Surf sal**	**Density**
Deployment date			
Deployment notes			
Science collection notes			
	Date	**Tech**	**Notes**
Ballast completed			
Software checklist completed			
Missions simulated			
Dockserver tested and updated			
Dockserver IP			
Pre-seal checklist completed			
Post-seal checklist completed			
All supplies packed			
Deployment details			
Cruise leaves			
Arrive on station			
Recovery details			
Cruise leaves			
Emergency recovery plans			

Pilot’s contact information	When	Phone	Email
Mission notes			

Pre-mission Seal Checklist (Final Seal)

All ballasting complete and weights adjusted; see the “Ballasting and H-moment” worksheet on page 47.

	Date	Tech	Notes
Fore			
Pump lead screw clean and greased			
Pitch lead screw clean and greased			
Leak detect in place; batteries secured			
Ballast bottles secured			
O-ring inspected and lubed			
Exterior nose/bellow clean of debris			
Interior clean of debris			
Reconstituted or fresh dessiccant installed			
Payload			
Science serial numbers			
1			4
2			5
3			6
Wiring dressed			
O-ring inspected and lubed			
Payload weights properly secured			
CF card fully seated and loaded			See software checklist (lab section)
Persistor button batteries checked			voltage
Interior clean of debris			
Aft			
Iridium SIM card installed			
SIM number			
Aft tray wiring dressed			
CF card seated and loaded			See software checklist (lab section)
Persistor button batteries checked			voltage
Ballast bottle secured			
O-ring inspected and lubed			
Battery voltages @ J13			
Fore			voltage
Pitch			voltage
Aft			voltage
All			voltage
Battery voltage @ J31 (emergency)			
Anode to main tray continuity			
Threaded rod clean and greased			
Seal			
O-rings clean of debris			
15 in/lb torque			
All sections snug together			
Vacuum pulled			

Post-seal Checklist

	Date	Tech	Notes
General			
Pick point installed			
Wing rails installed			
Wings and spared packed			
Hardware			
Exterior connectors secure and fastened			
Altimeter			
Aanderaa (if present)			
Burn wire			
MS plug seated			
Ejection weight assembly not seized			
Pressure sensors clear and clean			
Aft (flight)			
Payload (science)			
Continuity—Aft anode to tail boom			
Bladder visual inspection			
Cowling installed			
Powered by battery **inside lab**			
`Lab_mode_on`			in/hg
`Report ++m_vacuum` (6 in/Hg 7 for 1000 m)			volt
`Lab_mode_on  Wiggle on`			no errors for +5 minute
Verify time			
Verify science			
`Put c_science_all_on 0 (off = -1)`			
`Put c_science_on 3 (off = 1)`			
`Put c_science_send_all 1 (off = 0)`			
Powered by battery **outside lab tests**			
3 hrs Argos `put c_argos_on 3 (off = 1)`			
Confirm receipt of messages at Argos			
Confirm GPS			
Confirm compass			
Dockserver communications—send and receive files			
Run status.mi			
Notes:			

Shipping Checklist

	Date	Tech	Notes
Glider packed and secured with 3 straps			
Mobile computer packed			
FreeWave and FreeWave antenna			
Buoy with rope			
Glider evacuation tools			
Glider tools			
Red and green shorting plugs			
Wings packed			
Shipping address and details arranged			
Address	**Contacts**		**Details**
Flights	**Contacts**		**Details**

Ballasting and H-moment Checklist

	Date	Tech	Notes
Glider under vacuum			
Pick point installed			
Wing rails installed			
Exterior connectors secure and fastened			
Altimeter			
Aanderaa (if present)			
Burn wire			
MS plug seated			
Ejection weight assembly not seized			
Pressure sensors clear and clean			
Aft (flight)			
Payload (science)			
Bladder visual inspection			
Powered			
`Report ++m_vacuum` (6 in/Hg 7 for 1000 m)			
`Report ++m_battery`			
`Lab_mode_on  Wiggle on`			
`ballast`			
Cowling installed			
While in ballast tank			
Ensure no air in front or aft sections			
Note roll for potential adjustment			
Record weight adjustments necessary			
Rinse and dry after wettings			
`Exit` and power down glider when done			

Glider Ballast Worksheet

Glider Name: ______________ Date: ______________

Glider Displacement Disp (liters): ______________ Technician: ______________

Tank Water

Temperature (degrees C): ______________

Conductivity (S/M): ______________

Salinity (PSU): ______________

Density (kg/cu m): ______________

Target Water

Temperature (degrees C): ______________

Conductivity (S/M): ______________

Salinity (PSU): ______________

Density (kg/cu m): ______________

Weight Conversion Constants

Stainless Steel = .875 * (weight added external)
Lead = .912 * (weight added external)

First Run

	Forward	Payload	Aft
Weight Removed			
Weight Added			

Roll:

Second Run

	Forward	Payload	Aft
Weight Removed			
Weight Added			

Roll:

Third Run

	Forward	Payload	Aft
Weight Removed			
Weight Added			

Roll:

Final Weight Configuration as Shipped

Forward	Weight	Payload	Weight	Aft	Weight
Port Bottle		Top FWD		Aft Bottle	
STBD Bottle		Bottom FWD			
Bottom Bottle		Top AFT			

Roll:

H-moment:

Software Checklist

	Date	Tech	Notes
Flight CF Card contents archived			
Version updated			Version
Logs archived/deleted			
If new version			
Boot pico			
Load new app			
Install autoexec.mi in config directory			
Burnapp			
Confirm app			
Boot app			
Flight CF Card contents archived			
Version updated			Version
Logs archived/deleted			
If new version			
Boot pico			
Load new app			
Install proglets.dat in config directory			
Burnapp			
Confirm app			
Boot app			
Directory’s flight Persistor			
/Config			
Simul.sim detected			
Configure sbdlist.dat and mbdlist.dat			
Autoexec.mi			
sensor: c_iridium_phone_num			Number
sensor: F_MAX_WORKING_DEPTH(m)			Depth (m)
Confirm installations			
Confirm calibration coefficients			Only necc, if new hardware
/ma / missions			
Load custom .mi and .ma files			Files loaded
Sci>/proglets.dat			
Confirm desired sensors are installed			
Archive of all files locally			

TELEDYNE
WEBB RESEARCH
Everywhereyoulook™

49 Edgerton Drive
North Falmouth, MA 02556 USA
www.webbresearch.com