Wave Glider Sensor User Guide

For WET Labs ECOPuck

Change Record

Date	Description of change	By
7/19/2011	Initial version	Robert Olson
8/03/2011	1st review	Viet Tran
8/24/2011	Added N seconds in --start and -v | --verbose in EXPORT command	Viet Tran
8/24/2011	Minor textual changes	Robert Olson
8/30/2011	Added --setconfig, fixed --verbose, and modified the samples	Viet Tran
10/4/2011	Review	Robert Olson

Table of Contents

1. System Overview

1.1 Product Overview

The Liquid Robotics integration of the WET Labs ECOPuck provides a mobile, long endurance platform for collecting fluorescence and optical scattering data using the ECO Puck family of sensor channels, including chlorophyll, CDOM and others. All channels offered by WET Labs are supported by the LRI integration.

The ECO Puck sensor is managed by an application program which runs on the LRI Sensor Management Computer (SMC). The name of the program is configurable, to allow you to run multiple copies on the SMC if desired. In this document it will be referred to as ECOSensor. ECOSensor manages both the serial connection to the ECOPuck and the export of data to other applications and shore-side users.

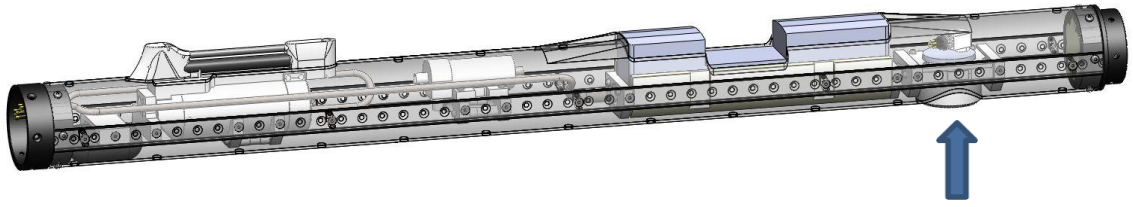
The product supports the full command set of the ECOPuck sensor and is powered by the Wave Glider's solar energy based electrical system. Samples are stored persistently on-board the Wave Glider. Custom post-processing applications written in C, C++, Java and a variety of scripting languages can run on-board in the Linux-based SMC. Data can be sent to shore using standard Wave Glider communications channels as well as payload-resident modems. The ECOSensor application software may be reconfigured dynamically to reflect changing mission objectives and constraints.

Samples on-board the SMC are stored in WET Labs-native formats, extended with tags for latitude, longitude and Unix timestamp for each sample. The reader is referred to WET Lab's documentation and web site for details of the WET Labs data formats.

This manual primarily focuses on the LRI integration specific aspects of the product, such as mechanical integration and the command set available for controlling ECOSensor program. It assumes a working familiarity with the features and operation of the ECOPuck sensor.

1.2 Mechanical

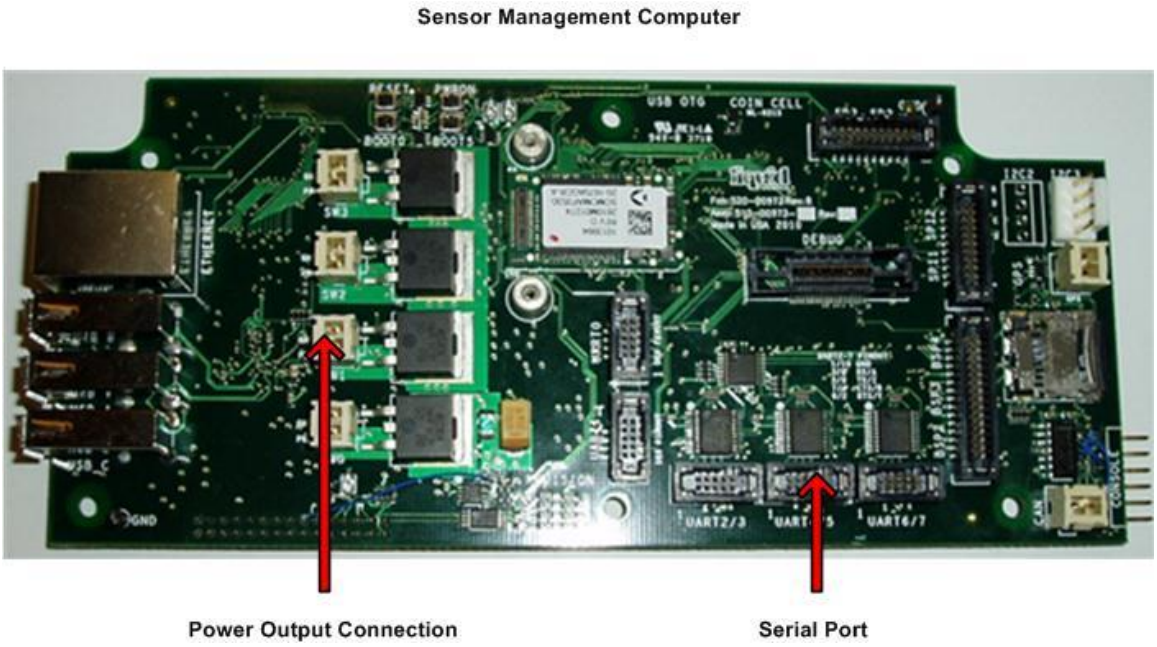
The ECOPuck sensor may be mounted in a variety of locations on the Wave Glider. The most common locations are through the hull of the float or in the Liquid Robotics configurable tow body. The tow body integration is shown. Float integration is dependent on considerations beyond the scope of this document, such as other sensors mounted on the Wave Glider's float.



In the illustration, the ECOPuck sensor is mounted in the tow body along with a WET Labs C-Star Transmissometer and a Sea Bird Electronics Glider Payload CTD. All would be cabled to a Liquid Robotics Sensor Management Computer that also would reside in the tow body and communicate up the umbilical to the Command and Control (C &C) module and other electronics in the float.

1.3 Electronics

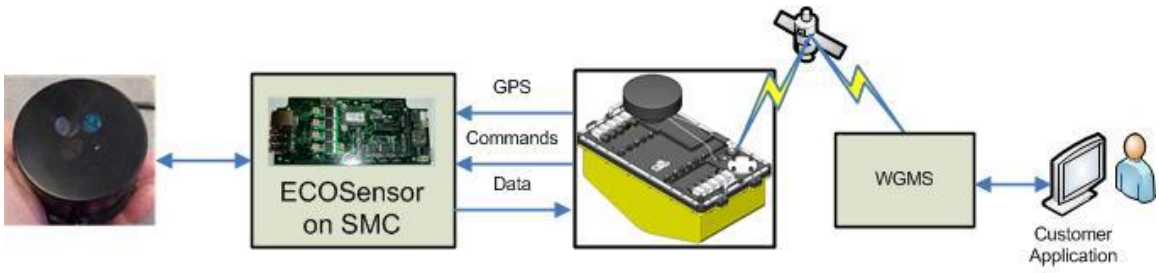
The ECOPuck sensor is controlled by software running on the Sensor Management Computer. Power is provided by one of the power output ports on the SMC, and the RS-232 connection to the sensor is provided by one of the serial ports on the SMC. The power port number and the serial port number are configurable in the XML configuration file.



Cables are included with the product.

1.4 Software

Broadly, data flows through the paths as shown in the image ECOPuck Control and Data Flow.



ECO Puck Control and Data Flow

ECOSensor supports three types of commands. The Select and Export commands are similar to those provided for other LRI-integrated sensors. The ECOPuck command provides for direct control of the ECOPuck sensor itself.

Starting from the right, the user plans his use of the ECOPuck using appropriate resources, such as WET Labs documentation. He then uses WGMS to send commands to the ECOSensor application program running on the SMC. These can be individual commands or a file of commands (a script). These commands set parameters such as the number of measurements to average per sample and provide control functions such

as turning the sensor on. Commands also can be issued from a shell script running on the SMC, using the LRI ServerExec command.

When commanded by the user, the ECOPuck will autonomously send samples to ECOSensor. ECOSensor will store the samples in the SMC file system in a raw format and can select a subset of the samples to be made available to other SMC-resident applications via a SMC-resident event manager.

The SMC event manager delivers the data to other applications that request it. In the most common case, the data is delivered to the SMC-resident payload manager. The payload manager will package the data into an LRI payload telemetry message (type 0x91) and send through the C&C communications infrastructure for delivery to shore. In the typical case, WGMS will hold the data on behalf of the user. The format of the 0x91 message is described later in this document.

Data (in the form of events) also can be delivered to SMC-resident applications other than the payload manager. For example, a custom program could filter and compress samples and then deliver them to the payload manager to be forwarded to WGMS. This could reduce data transmission with consequent savings in power and bandwidth. Another possible application would be to study chemical gradients by controlling the Wave Glider navigation directly from a SMC application that monitors the ECOPuck.

1.5 Related Documents

- WET Labs ECO Three-measurement Sensor User's Guide, Revision F, 7 May 2009
- Interface Control Description, version 2.00, Liquid Robotics P/N 030-00026-02

1.6 Glossary

ECOPuck	The WET Labs fluorometry sensor of the same name.
WetLabsSensor	The Liquid Robotics application program that runs on the Sensor Management Computer and directly manages the ECOSensor program.
ECOSensor	The configurable process name of an instance of the WetLabsSensor program which is configured for interfacing to an ECOPuck sensor.
ECO	The prefix for issuing commands to, and displaying responses from, the ECOPuck sensor.
ECOPuck Script	A sequence of ECOPuck commands collected in a file intended to be executed in sequential order by the ECOSensor process.
C&C	The Wave Glider's Command and Control module.
LRI	Liquid Robotics, Incorporated
SMC	Sensor Management Computer – an embedded Linux application processor that is resident in a payload bay.
WGMS	Wave Glider Management System – the LRI shore-side control application.

2. Collecting Data

2.1 Data Collection Process Overview

Collecting data with the ECOPuck involves four operations.

1) Plan the frequency of data collection and reporting.
2) Send ECOSensor commands to configure the collection and reporting process.
3) Start collecting the data
4) Retrieve the data from the SMC and/or WGMS.

Step 1: Plan data collection and reporting

The ECO Puck can be configured to report averages of multiple measurements. ECOSensor receives samples at whatever rate the sensor sends them and stores them in a raw data file. In addition, ECOSensor can be directed to select a subset of those samples for further attention. The typical use of this feature is to send a subset of the samples to shore. These choices affect power consumption, storage requirements and communications bandwidth consumption.

Step 2: Configure the collection and reporting process

The four functions performed by ECOSensor are to forward commands to the ECO Puck, collect data sent back by the ECO Puck, store that data on the SMC and forward samples of interest to shore or to other processes running on the SMC. These functions are implemented through the commands described in chapter ???.

ECOSensor can execute commands singly from WGMS, from a Linux shell script on the SMC or from a file stored on the SMC. A typical sequence of operations for an ECOSensor command script would be

- Set the number of measurements to average into a sample.
- Set the name of the file to hold raw samples
- Set the frequency of data reports to shore
- Start sampling (and reporting)
- Stop sampling

Typically you will have multiple ECOSensor scripts stored on the SMC to cover each scenario you anticipate. These scripts can be downloaded using the mechanism described in Appendix ???.

Step 3: Start collecting data

The ECO Puck will start streaming data to ECOSensor when the “$run” command is sent to the sensor. ECOSensor will automatically process the stream based on the instructions it has received.

Step 4: Retrieve the data for analysis

All raw data is stored in a file in the SMC file system, using a file name that you control. That file can be retrieved from the SMC through a communications transport such as RUDICS. The use of RUDICS is described in Appendix ???. The format of data in the file is described in section ???

Samples selected through the SELECT and EXPORT commands will be sent to shore using the LRI payload telemetry message and be stored in WGMS. Typically they will be retrieved from WGMS by using the WGMS Export function. The format of data in WGMS is described in section ???

2.2 Planning Data Collection

The combination of the Wave Glider sensor management system with the WET Labs ECOPuck supports a large number of possible applications. The planner must balance collection objectives and environmental constraints when developing the detailed collection plan.

One of the key advantages of the LRI design is that it is easy to modify a collection program at any time, including after launch. The core mechanism for controlling the ECOPuck is to send ECOSensor scripts to execute. These scripts can be executed from the SMC file system or be "real-time", sent from shore. We recommend preparing for likely alternatives prior to launch so that you can test and pre-load the scripts on-shore, which is much cheaper and faster than downloading them via Iridium. Of course, it is always possible to load new scripts onto a Wave Glider at sea.

Key considerations include:

- Reporting frequency – On average, the Wave Glider moves about 50 meters per minute. The ECOPuck provides a mechanism for averaging readings. Taking longer averages reduces data storage requirements and bandwidth consumption. In addition, it is possible to sub-sample the data collected by the sensor based on distance traveled, time and number of samples. This also can reduce resource consumption.
- Power budget – The ECOPuck subsystem is a relatively low power sensor. However, its consumption must be included when assessing the total power budget for the Wave Glider.
- Communications bandwidth – It is possible to configure the ECOPuck system to generate more raw data than the Iridium communications channel can transport. Once a session is established, the underlying maximum data rate of the Iridium network is 2400 bps. ECOSensor provides several options for collecting raw data at full rate and then returning just a subset of that data to shore. If those options are not appropriate for your application, it is possible to write custom code for the SMC to reparse the data into a form more useful to you.
- Onboard storage – In general, ECOSensor stores the raw samples in the SMC file system. The SMC generally ships with a 16GB Secure Digital flash file system, of which approximately 12GB is available for user data. You should estimate how much data your application will generate and configure the system accordingly. Managing file system space and deleting unneeded files is a user responsibility.

2.3 Configuring ECOSensor

The WetLabsSensor program is configured for interfacing to an ECO Puck sensor, using the WetLabsSensor-config.xml file located on the Sensor Management Computer in /home/smc/roots/200_ECO/, and assigned the default process name **ECOSensor**. You can manage multiple ECO Puck sensors by creating multiple instances of ECOSensor. Each instance must have a distinct process name, such as ECOSensor2.

The ECOPuck is commanded by means of an "*ECOSensor script*". An ECOSensor script is a sequence of commands that are sent to ECOSensor. These commands generally include commands to the sensor itself to configure measurement parameters such as averaging, as well as commands that control the behavior of ECOSensor.

In most cases, the ECOPuck script is read from a file in the SMC file system. A command to the ECOPuck to start returning data results in samples being sent to ECOSensor automatically. The samples being returned conform to WET Labs data format rules. These samples are appended to the currently open ECOSensor output file and processed according to the SELECT and EXPORT command parameters.

Scripts can be downloaded to the SMC by following the instructions in Appendix ???

If possible, you should create and test a separate ECOPuck script for each configuration you anticipate using. For example, if you anticipate running the system with two different averaging periods, you should test both scripts and store them on the SMC with a descriptive file name. It is possible to send scripts during a mission, but testing them prior to deployment reduces the risk of errors.

Commands also can be sent to the ECOSensor process from other applications running on the SMC, such as a Linux shell script, by using the ServerExec shell command. This can be very convenient, as it allows you to integrate multiple sensors in your decision making. For example, you might change the sampling rate based on a trigger from a different ECOPuck on the same Wave Glider. You can create arbitrarily complex ECOSensor scripts.

Example

<need example script here>

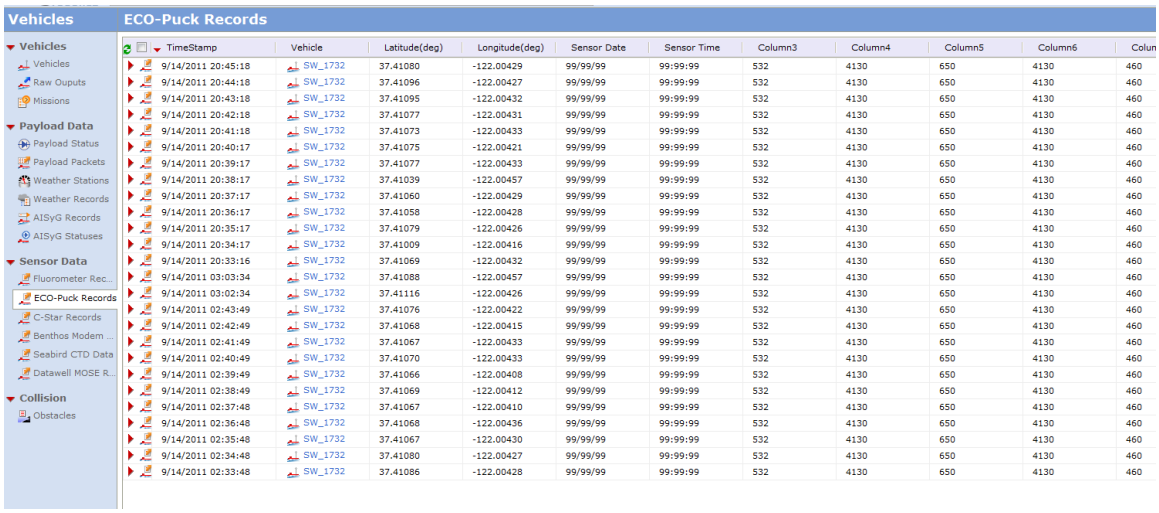
2.4 Collecting Data

The ECOPuck --start command issues a $run command to the ECO Puck. The sensor responds by automatically returning samples. ECOSensor will store the returned samples in the raw data file and subsample the data stream according to the SELECT and EXPORT commands.

2.5 Retrieving ECOPuck data

Data can be sent to shore as generated or can be stored on board the Sensor Management Computer for later retrieval.

If you are using WGMS, samples returned as generated are stored in the WGMS database, from which they can be retrieved using standard WGMS mechanisms. You also can view the data directly in WGMS. Below is an example of how the data looks in WGMS.



Files containing samples can be collected from the SMC using standard file transfer mechanisms. Those mechanisms are described in ???

2.6 Maintenance and Service

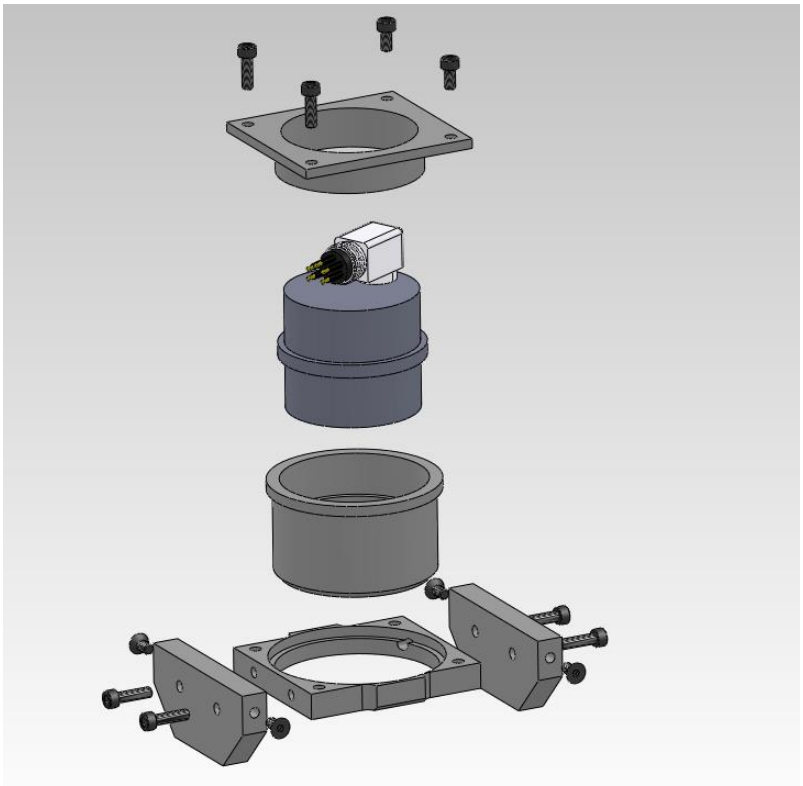
WET Labs software and documentation, included in the shipment, describe WET Labs-recommended testing and maintenance procedures. Maintenance and service for the ECOPuck itself are provided by WET Labs. There are no maintenance requirements specific to the Wave Glider.

3. Mechanical and Electrical Integration

3.1 Mechanical and Electrical Setup

3.1.1 Mechanical Setup

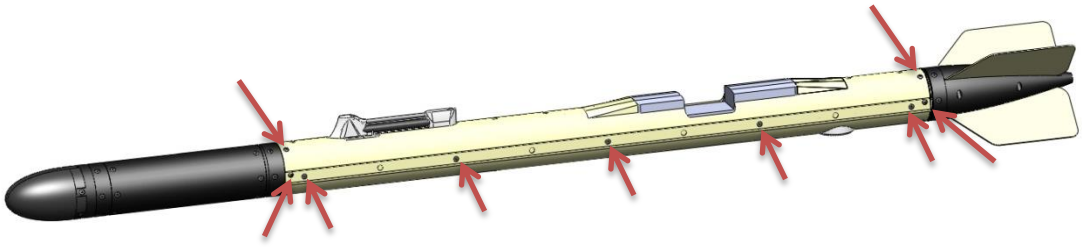
The Eco Puck is secured in the METOC Towfish using five UHMW clamps and brackets. This approach allows for convenient removal and re-installation, while remaining mechanically robust. The Eco Puck, housed in a sleeve, is clamped between two of the brackets, which are attached to the bottom shell of the METOC Towfish.



3.2 Eco Puck Removal

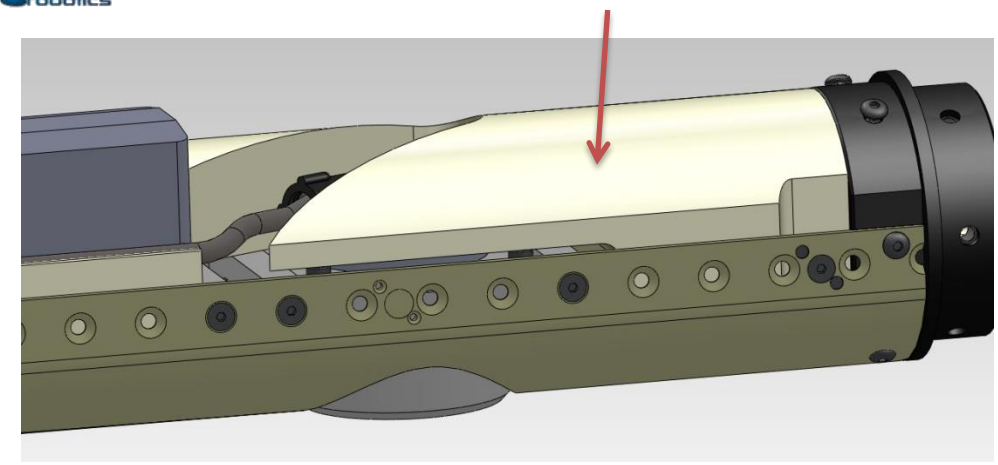
3.2.1 Remove Top Shell of METOC Towfish

Unscrew flat-head and button head screws that attach the top shell to the bottom shell. There should be 18 screws in total to remove.



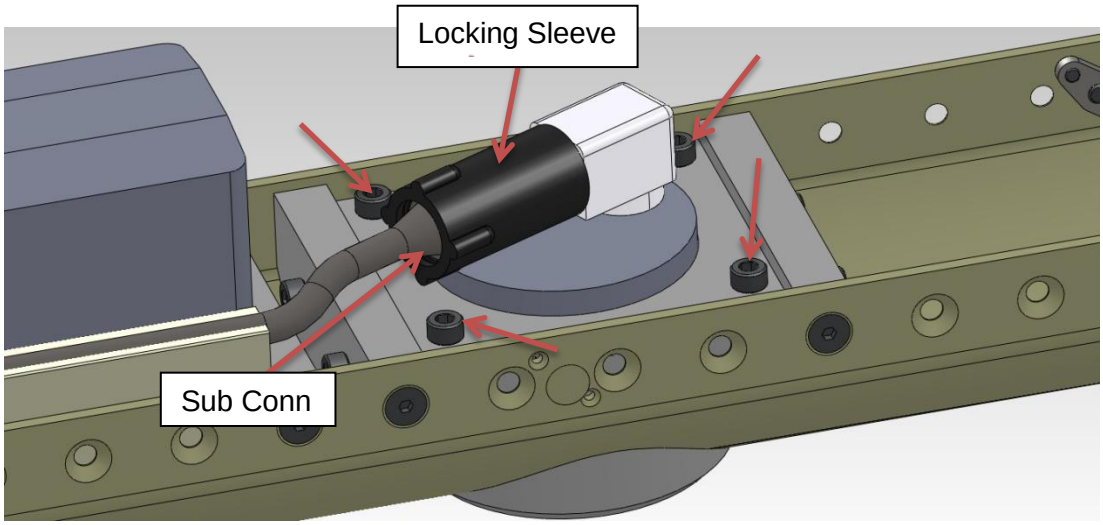
3.2.2 Remove Foam

Pull foam piece on top of eco puck vertically out of shell.

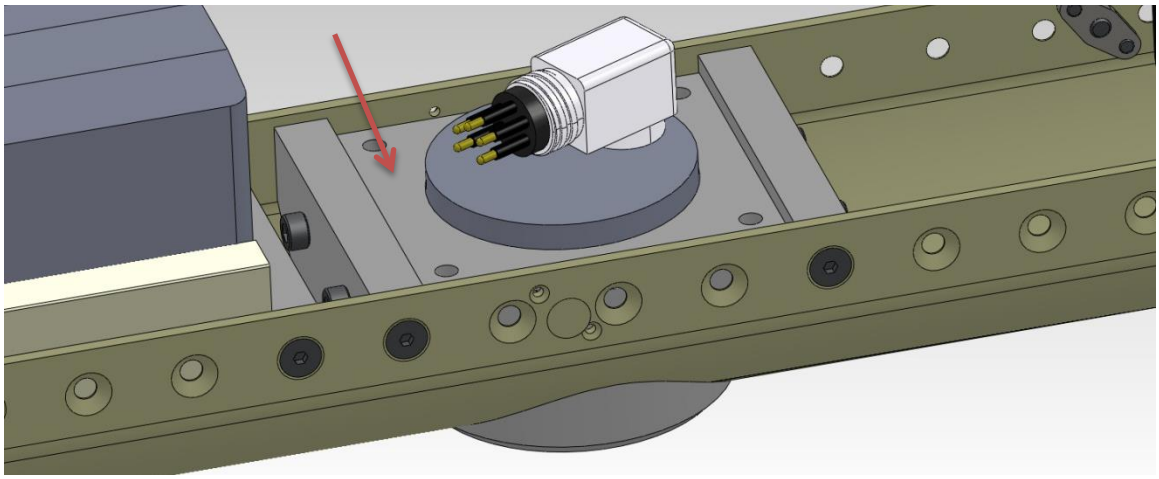


3.2.3 Remove Screws from top clamp, detach Sub Conn

Unscrew four SHCS from top clamp. Unscrew locking sleeve. Unplug Sub Conn.



3.2.4 Remove Top Clamp



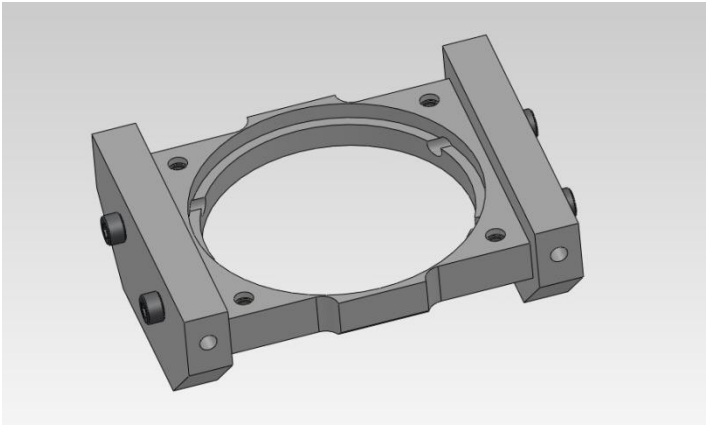
3.2.5 Remove Eco Puck and Sleeve

3.2.6 Remove Sleeve

3.3 Eco Puck Installation

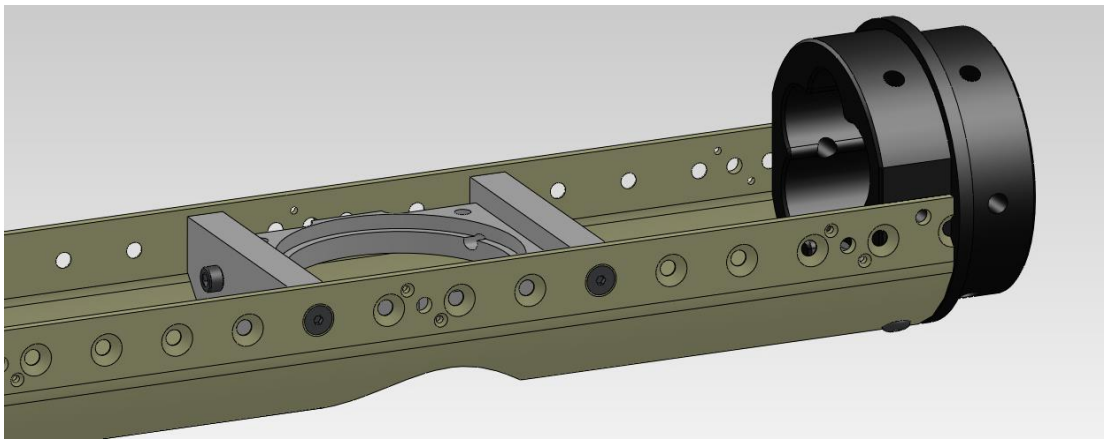
3.3.1 Assemble Chassis

Assemble using four M5x0.8 18MM SHCS.

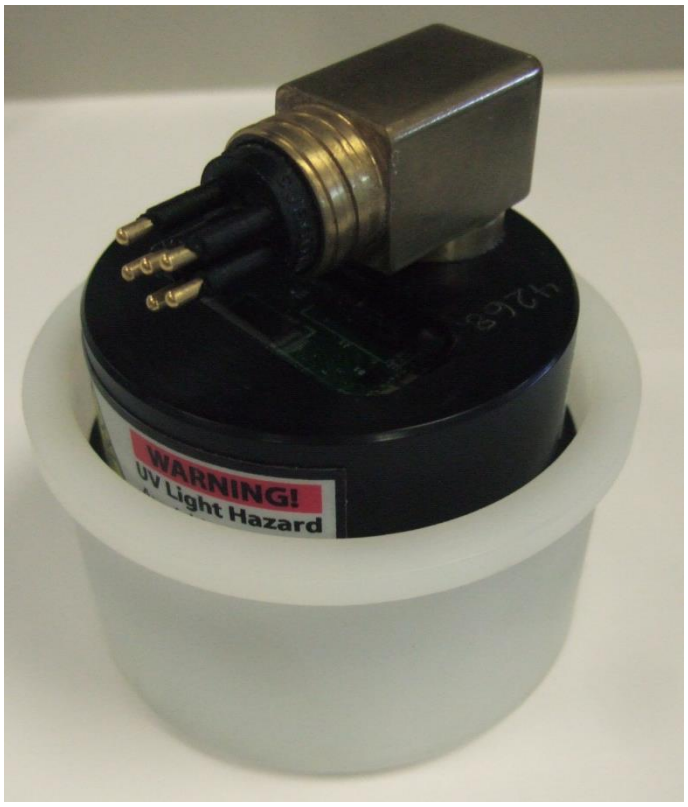


3.3.2 Install Chassis in Bottom Shell

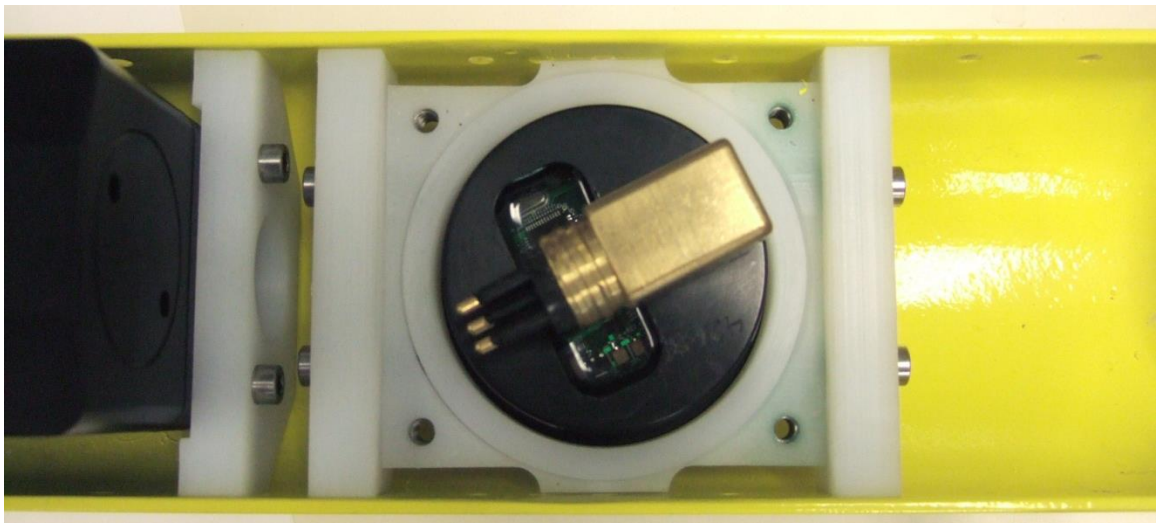
Install using four M5x0.8 12MM FHCS. Position according to picture below.



3.3.3 Put Eco Puck in Sleeve.

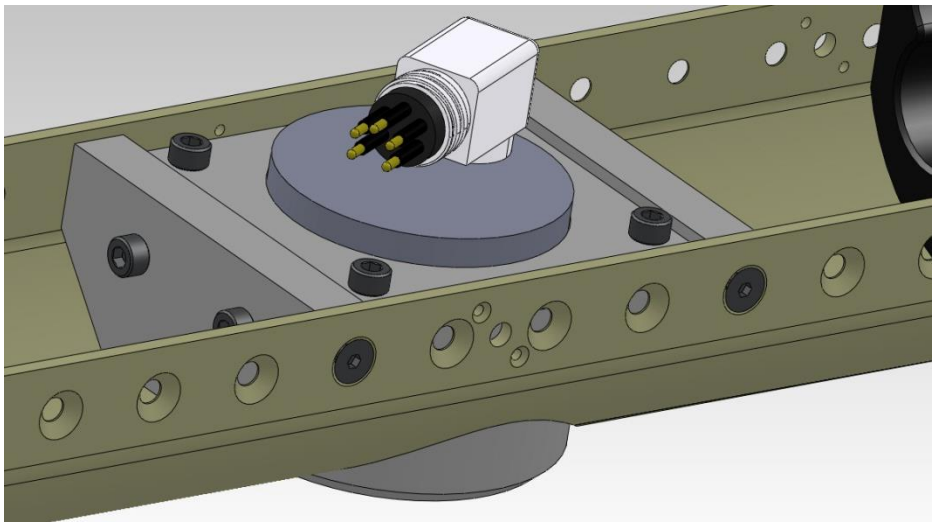


3.3.4 Place Sleeve and Puck in Chassis. Align with Foam



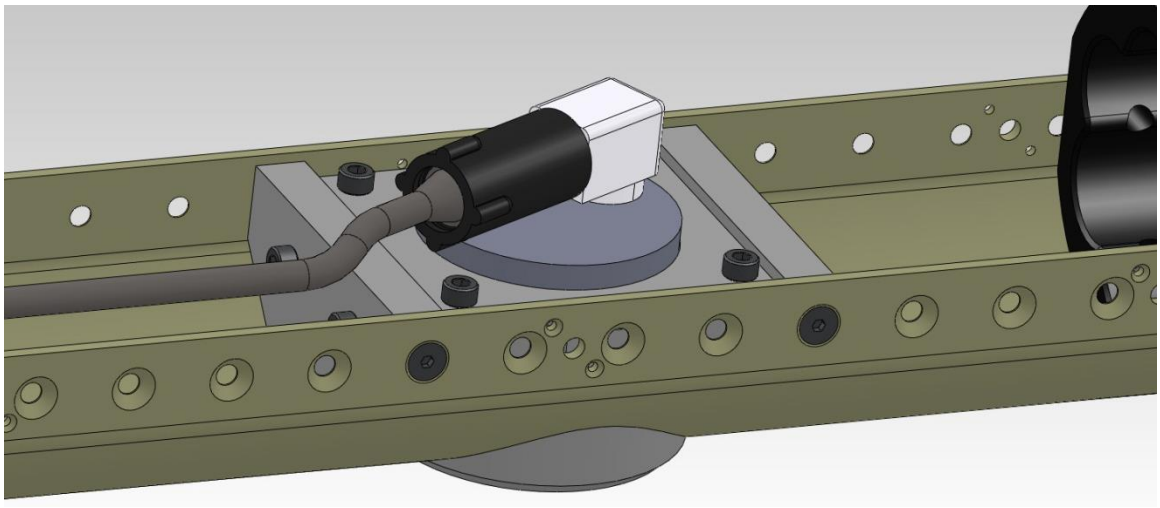
3.3.5 Secure Eco Puck with Top Clamp

Insert clamp and secure with four M5x0.8 18MM SHCS. Use Loctite 243.

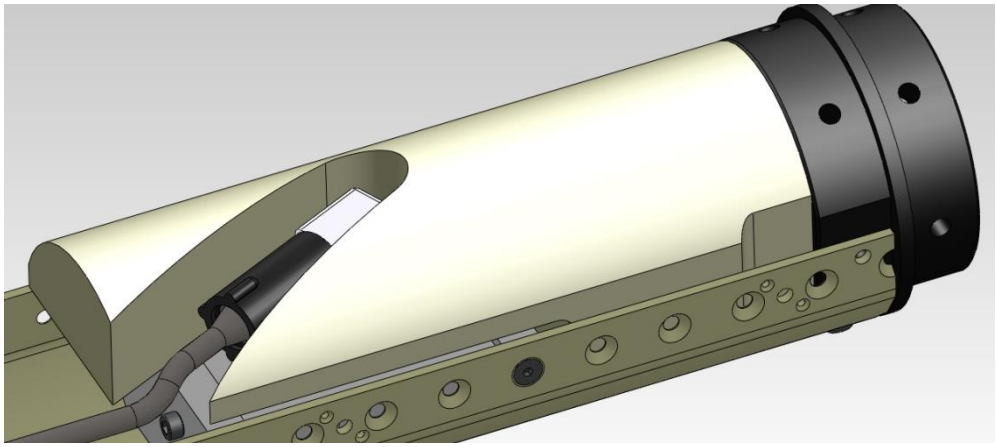


3.3.6 Plug in Cable

Plug in Eco Puck cable, spraying mating surfaces with 3M Silicone Lubricant. Secure sub conn with locking sleeve and finger tighten.



3.3.7 Position Foam above Eco Puck



4. ECOSensor Commands

Like other sensors, ECOPUCK sensor has 3 basic commands - ECOPUCK, SELECT, and EXPORT. They can be sent individually or as part of a script file.

At power on, the ECOPUCK sensor does not start to collect data samples right away. The firmware power-cycles the ECOPUCK sensor and then puts it in the STOP state. Users can send commands to the sensor at this point.

4.1 ECOPuck Command

The ECOPuck command invokes operations that directly control the ECOPuck sensor. It starts with the key word “ECOPuck” followed by a required command parameter.

Synopsis

ECOPuck –r | --run [*pathname*]

ECOPuck --stop

ECOPuck --start [N]

ECOPuck -o | --output *pathname*

ECOPuck --sleep

ECOPuck --wakeup

ECOPuck --setavg

ECOPuck --setpkt

ECOPuck --getconfig

ECOPuck --setconfig

4.1.1 ECOPuck RUN [pathname]

Syntax

ECOPuck -r | --run [*pathname*]

Summary

Run the script located at *pathname*. Run the default script specified in the xml configuration file if the command does not specify a file.

Semantics

Run a sequence of commands in the script where each line is a command.

Although each command in the script gets its own NACK/ACK, that result is only used as a condition to continue to the next command. A NACK in the middle of the script will terminate the script, with the run command getting a NACK as the final result. If all commands execute successfully, the run command should get a single ACK. In addition to NACK/ACK, the event manager also receives a full string of responses combined from each of the successful command in the script.

Notes

Pathname may be relative or fully qualified within the SMC file system. *Pathname* is case sensitive. *Pathname* is an optional parameter.

Samples

Command:

ServerExec EcoSensor "ECOPuck --run"

Response:

ACK ecopuck --run ecoscript.txt, 2011/08/29, 23:03:23.980717, 37.4074085, -122.0043925
ECOPUCK --output ecosamples.txt
EXPORT --output ecoexports.txt
SELECT -t 30
EXPORT --count
20000
ECOPUCK --start
ACK, ECOPUCK --output ecosamples.txt, 2011/08/29, 23:03:23.992802, 37.4074085,
-122.0043925
ACK, EXPORT --output ecoexports.txt, 2011/08/29, 23:03:23.993626, 37.4074085,

-122.0043925
ACK, SELECT -t 30, 2011/08/29, 23:03:23.994205, 37.4074085, -122.0043925
ACK, EXPORT --count 20000, 2011/08/29, 23:03:23.994449, 37.4074085, -122.0043925
ACK, ECOPUCK --start 99/99/99 99:99:99 532 887 650 434 460 79 524, 2011/08/29, 23:03:39.068380, 37.4074085, -122.0043925, 2011/08/29, 23:03:39.152604, 37.4074085, -122.0043925

4.1.2 ECOPuck STOP

Syntax

ECOPuck --stop

Summary

Puts ECO Puck into a standby mode

Semantics

Send "!!!!!" to the sensor to stop receiving samples. This command puts the sensor into a standby mode (STOP STATE) as it changes the state from RUN to STOP and leaves the power on. It can be sent individually or as part of a script.

Notes

This command in effect pauses the ECO Puck processing and does not change the settings. Users can change the settings at this point and/or resume with the "ECOPuck --start" command.

Sending a stop command at SLEEP state will go through the wake up sequence while sending a stop command at STOP state results in an ACK.

Samples

Command:

ServerExec EcoSensor "ECOPuck --stop"

Response:

ACK EcoSensor ECOPuck --stop Ser BB2FLLIQ-802 Ver Triplet 4.07 Ave 50 Pkt 0, 2011/06/28, 17:05:51.245025, 37.410807999999994, -122.0042895

4.1.3 ECOPuck START

Syntax

ECOPuck --start <N>

Summary

Start receiving data samples from the sensor

Semantics

Send a "$run" command to the sensor and wait N seconds for the samples from it. Command parameter N is optional. If N is not specified, N uses the default value of 30 seconds (reconfigurable in the WetLabsSensor xml) or the previous value set by the user. If N equals to 0, it will wait forever. This command alters the state of the sensor from STOP to RUN.

The response of the ECOPuck --start contains the first sample received from the sensor.

Notes

This command does not change the settings as it just resumes from the previous stop.

Sending a start command at RUN state does not change the STATE or the settings of the sensor. Sending a start command at SLEEP state will go through the wake up sequence.

Samples

Command:

ServerExec EcoSensor "ECOPuck --start"

Response:

ACK, ECOPUCK --start 99/99/99 99:99:99 532 887 650 434 460 79 524, 2011/08/29, 23:03:39.068380, 37.4074085, -122.0043925

Input Data Samples:

Raw date	Raw time	N/U	650mm	N/U	Chl	N/U	CDOM	Therm	Date
99/99/99	*99:99:99*	*532*	*4130*	*650*	*4130*	*460*	*4130*	*525,*	*2011/06/28*

Time	Latitude	Longitude
17:05:45.430877	*37.410927833333325*	*-122.00422849999999*

4.1.4 ECOPuck OUTPUT pathname

Syntax

ECOPuck -o | --output pathname

Summary

Directs samples to the file located at *pathname*.

Semantics

Save all data samples to a file in the SMC file system. If *pathname* already exists, samples will be appended to the existing file. If *pathname* does not exist, the file is created. If EcoSensor is unable to open *pathname*, the command will fail, the process will return a NACK and the previously active output file will continue to be used.

Notes

Pathname may be relative or fully qualified. *Pathname* is case sensitive. *Pathname* is a required parameter.

This command can be sent at any state.

Samples

Command:

ServerExec EcoSensor "ECOPuck --output ecosamples.txt"

Response:

ACK, ECOPUCK --output ecosamples.txt, 2011/08/29, 23:03:23.992802, 37.4074085, -122.0043925

4.1.5 ECOPuck SLEEP

Syntax

ECOPuck --sleep

Summary

Turn power off

Semantics

Turn the SMC resident power switch off to cut power to the sensor regardless of what state the sensor is in.

Notes

Use “ECOPuck --wakeup” to turn it back on.

Samples

Command:

ServerExec EcoSensor "ECOPuck --sleep"

Response:

SetPowerSwitch: 0x00

ACK ECOPuck --sleep, 2011/06/28, 22:39:41.082305, 37.410954500000006, -122.00444483333334

4.1.6 ECOPuck WAKE UP

Syntax

ECOPuck --wakeup

Summary

Power cycle the sensor and leave it in STOP state

Semantics

This wake up command is a sequence of 3 different commands. Turn off the power, wait 1 minute as recommended by WET Labs, turn the power back on and wait 30 seconds that are set in the XML configuration file, and finally send a stop command to pause it.

Notes

This command does not change the previous settings.

Samples

Command:

ServerExec EcoSensor "ECOPuck --wakeup”

Response:

SetPowerSwitch: 0x00

Sleep 1 minute after power off

SetPowerSwitch: 0x44

ACK ECOPuck --wakeup Ser BB2FLLIQ-802Ver Triplet 4.07 Ave 50 Pkt 0, 2011/06/28, 17:17:09.932830, 37.410792666666675, -122.00432366666669

4.1.7 ECOPuck SETAVG

Syntax

ECOPuck --setavg N

Summary

Set the average number of measurements for each sample

Semantics

Set the average value by sending “$ave” to the sensor. Each received sample is the average value of the number of measurements.

N = 1 - 1000. Default is the manufacturing value which is set to 1Hz.

Samples

Command:

ServerExec EcoSensor "ECOPuck --setavg 50"

Response:

ACK ECOPuck --setavg 50 Ser BB2FLLIQ-802Ver Triplet 4.07 Ave 50 Pkt 0, 2011/06/28, 17:25:16.262023, 37.410737833333337, -122.00434933333331

Command:

ServerExec EcoSensor "ECOPuck --setavg 20000"

Response:

NACK ECOPuck --setavg 20000 average value should be >=1 and <=1000

4.1.8 ECOPuck SETPKT

Syntax

ECOPuck --setpkt N

Summary

Set the total number of samples to be received

Semantics

Set the total number of samples to be collected in one run by sending “$pkt” to the sensor.

Notes

N = 0 - 65535. Default is 0 = forever.

If you configure the number of samples to a low number (less than 5), the meter will sample the specified number of times, then it may go into a sleep state. You will be unable to communicate with it at this state. Use command “ECOPuck --wakeup” to wake it up.

LRI recommends using a combination of start, stop, sleep, and wake-up commands to more finely control the sensor as an alternative to using this command.

Samples

Command:

ServerExec EcoSensor "ECOPuck --setpkt 20000"

Response:

ACK ECOPuck --setpkt 20000 Ser BB2FLLIQ-802 Ver Triplet 4.07 Ave 50 Pkt 20000, 2011/06/28, 17:45:06.386840, 37.410999666666663, 122.00412633333331

4.1.9 ECOPuck GET CONFIGURATION

Syntax

ECOPuck --getconfig

Summary

Get the settings from the internal memory of the sensor.

Semantics

Retrieve the settings from the internal memory by sending “$mnu” to the sensor. The result is the full contents of the menu returned from the sensor.

Samples

Command:

ServerExec EcoSensor "ECOPuck --getconfig"

Response:

ACK ECOPuck --getconfig Ser BB2FLLIQ-802 Ver Triplet 4.07 Ave 50 Pkt 20000, 2011/06/28, 17:50:09.551208, 37.411158666666675, -122.004228

4.1.10 ECOPuck SET CONFIGURATION

Syntax

ECOPuck --setconfig

Summary

Save the settings to the internal memory of the sensor.

Semantics

Save the settings to the internal non-volatile memory by sending a “$sto” to the sensor. The result is a word “done” returned from the sensor.

Samples

Command:

ServerExec EcoSensor "ECOPuck --setconfig"

Response:

ACK ECOPuck --setconfig done

4.2 SELECT command

Syntax

Select [-t | --time | -s | --samples | -d | --distance] n

Summary

Direct ECOSensor to select samples from the ECO Puck's data stream for further processing.

Semantics

ECOSensor, like other sensor managers, tags samples sent by the ECOPuck with GPS-derived latitude, longitude and timestamp. The SELECT command directs ECOSensor to select samples from that data stream for further processing.

This command can be entered at any time and at any state. It will change the N value instantly in the next input sample.

Time, sample counts, and distances are initialized to the first sample received after the Export command is executed.

n is an integer which is interpreted in the context of the type option. The default type option is -s (--samples). Consequently, the default value of the select variable is to select every sample.

-t, --time – The parameter, n, sets a "tick" rate for a clock measured in seconds. The clock is initialized to zero by ECOPuck on receipt of the Export command. The selected sample is the first one after the tick. For example, if n is 5, there will be ticks at zero, 5, 10, 15, 20 and so forth. If samples are generated every second and n is 5, every fifth sample will be selected. If samples are generated every three seconds and N is 5, the sample selected will be the one at the 6th second, the one at the 12th second, the one at the 15th second, the one at the 21st second and so forth. If one sample is generated every sixty seconds and n is 1, every sample will be selected. In that case, only one sample will be selected even though sixty seconds passed. No sample is selected more than once even if multiple ticks intervene.

-s, --samples – This parameter specifies that the selected sample is the one that is n samples after the previously selected sample. For example, if n is 2, every second sample from the ECOPuck will be selected for further processing.

-d, --distance – Analogous to the time parameter, the parameter n sets a "tick" rate for an odometer measured in meters. The odometer is initialized to zero by ECOPuck on receipt of the Export command. The selected sample is the first one after the odometer passes a tick. For example, if n is 100, the selected sample will be the first one after the odometer passes 100, 200, 300 and so forth. The distance is calculated using GPS data.

Examples

SELECT --samples | -s 3: selects the next sample after the EXPORT command is processed and then every third sample.

Command:

ServerExec EcoSensor "SELECT --samples 10"

Response:

ACK SELECT --samples 3, 2011/06/28, 21:22:51.348358, 37.410914333333337, -122.00431616666669

SELECT --time | -t 10: selects the next sample after the EXPORT command is processed and uses its timestamp as the zero reference. It then selects the next sample after t=10, 20, 30, 40, 50, 60, 70, 80, 90.

Command:

ServerExec EcoSensor "SELECT --time 10"

Response:

ACK SELECT --time 10, 2011/06/28, 21:21:33.371307, 37.411145666666675, -122.00473733333336

SELECT --distance | -d 50: selects the next sample after the EXPORT command is processed and uses its lat/lon as the zero reference point. It then selects the next sample after d=50m, 100m, 150m, 200m.

Command:

ServerExec EcoSensor "SELECT --distance 50"

Response:

ACK ECOPuck --getconfig Ser BB2FLLIQ-802 Ver Triplet 4.07 Ave 40 Pkt 20000, 2011/06/28, 21:10:24.730865, 37.410949666666657, -122.00432950000001

4.3 EXPORT command

Syntax

Export --count | -o | --output *pathname* | --stop | -c | --column | -v | --verbose

Summary

Specify the total exported samples and reset the counter.

Direct the exported samples to the export file located at *pathname*.

Stop exporting the samples.

Indicate a combination of data columns of the selecting exported samples.

Semantics

--count – Set the total count of the exported samples.

-o | --output – Direct the exported samples to the exported file located at *pathname.*

--stop – Disables the export of samples. Export can only be restarted by issuing a new EXPORT --count command, which reinitializes the counters.

-c | --column – Specify which sample columns to export. Users can collect all 9 columns of a raw sample by default or use this command to select any combination of columns and in any order, as long as that particular combination has been defined in the PayloadInterface-Telemetry-config XML configuration file.

-v | --verbose – Select the verbosity level of the response messages.

SWSMAC version 1.1.0 has only 2 levels basically: on and off. <= 0 is off while >0 is on. The initial verbosity level is set to off (0) in the XML configuration file. If the level is not specified in the command, the level is set to on (1).

When the verbosity level is off, the response has a plain ACK instead of the response contents and the GPS info. All NACK responses are always set to on regardless of what verbosity level was set before.

Notes

The export types are mutually exclusive – only one can be active at any time.

This command can be entered at any time and at any state.

Samples

--count – Specify the total exported samples.

Command:

ServerExec EcoSensor "EXPORT --count 5000"

Response:

ACK EXPORT --count 5000, 2011/06/28, 21:29:25.304840, 37.411416666666675, -122.00394566666667

-o | --output – Direct the exported samples into the file located at *pathname.*

Command:

ServerExec EcoSensor "EXPORT --output /home/smc/roots/200_EcoSensor/exports.txt"

Response:

ACK EXPORT --output /home/smc/roots/200_EcoSensor/exports.txt, 2011/06/28, 21:33:23.884094, 37.410835666666663, 122.00423966666667

--stop – Disable the export of samples. Export can only be restarted by issuing a new EXPORT --count command, which reinitializes the counters.

Command:

ServerExec EcoSensor "EXPORT --stop"

Response:

ACK EXPORT --stop, 2011/06/28, 21:36:19.435089, 37.412057833333343, -122.00435999999999

-c | --column – Specify which sample columns to be exported in any combination and any order

Command:

ServerExec EcoSensor "EXPORT --count 2 3 4 5 6 7"

Response:

ACK EXPORT -c 3 4 5 6 7, 2011/07/05, 22:30:48.950990, 37.410903333333337, -122.004243

-v | --verbose – Select the verbosity level of the response messages.

Command:

ServerExec EcoSensor "EXPORT --verbose"

Response:

ACK EXPORT --verbose, 2011/06/28, 21:36:19.435089, 37.412057833333343, -122.00435999999999

Command:

ServerExec EcoSensor "EXPORT --verbose 0"

Response:

ACK

4.4 Comment command

Lines that start with a semi colon are comments and are ignored.

4.5 Example scripts

Assume the application has a local working directory. Pathnames will be relative to that working directory.

4.5.1 Example 1 – export 2000 samples at 30 seconds interval

```
ECOPUCK --output ecosamples.txt          ; all samples file
EXPORT --output ecoexports.txt           ; exported samples file
SELECT -t 30                             ; 30 seconds interval
EXPORT --count 20000                     ; export 20000 samples
ECOPUCK --start                          ; start to collect samples
```

Samples

Command:

ServerExec EcoSensor "ECOPuck --run"

Response:

ACK ecopuck --run ecoscript.txt, 2011/08/29, 23:03:23.980717, 37.4074085, -122.0043925
ECOPUCK --output ecosamples.txt
EXPORT --output ecoexports.txt
SELECT -t 30
EXPORT --count
20000
ECOPUCK --start
ACK, ECOPUCK --output ecosamples.txt, 2011/08/29, 23:03:23.992802, 37.4074085,
-122.0043925
ACK, EXPORT --output ecoexports.txt, 2011/08/29, 23:03:23.993626, 37.4074085,
-122.0043925
ACK, SELECT -t 30, 2011/08/29, 23:03:23.994205, 37.4074085, -122.0043925
ACK, EXPORT --count 20000, 2011/08/29, 23:03:23.994449, 37.4074085, -122.0043925
ACK, ECOPUCK --start 99/99/99 99:99:99 532 887 650 434 460 79 524,
2011/08/29, 23:03:39.068380, 37.4074085, -122.0043925, 2011/08/29, 23:03:39.152604, 37.4074085, -122.0043925

5. Data Formats

All data samples are stored in an all samples file. That file is created by default during initialization using the default pathname stored in the WetLabsSensor-config XML file. Users can create a new file in the SMC file system by using the ECOPuck output command.

The SELECT and EXPORT commands filter the sample stream for samples that meet specific criteria. The SELECTed samples can be stored in a file in the SMC file system or can be forwarded to shore using the Payload Telemetry Message, format 0x91.

As received by EcoSensor, an ECO Puck sample consists of a tab-delimited string of characters with each group of characters corresponding to a specific value, as

configured on the ECO Puck. This section describes the most common configuration, where the sensor returns nine values. EcoSensor processes that sample in different ways depending on whether the data is being returned via the 0x91 Payload Telemetry Message or being stored in a file on the SMC.

5.1 Real-time transfer

The telemetry is returned as a type 0x91 structure as follows. See WET Labs documentation for definitions of the sensor specific values. Where appropriate, the text strings are converted to binary to save space.

FORMAT CODE 0 – ECO PUCK SAMPLE

Field Name	Bytes Used	Byte #	Format
PayloadType	4	0-3	123 (WET Labs ECO Puck)
Board Id	1	4	Unsigned 8-bit integer
Task Id	1	5	Unsigned 8-bit integer
Sensor Data Format	1	6 (bits 0-3)	Unsigned 4-bit integer – 0
Default Parser	1	6 (bit 4)	0=Binary/1=ASCII – 0
Delivery Priority	1	6 (bits 5-7)	Unsigned 3-bit integer – 0
Repeat Count	1	7	Unsigned 8-bit integer

Individual samples

Latitude	4	0-3	Signed 32-bit integer
Longitude	4	4-7	Signed 32-bit integer
Time	4	8-11	Unix time_t as read on the SMC
Raw date from sensor	8	12-19	String returned by sensor
Raw time from sensor	8	20-27	String returned by sensor
Reference frequency 1	2	28-29	Unsigned 16-bit integer
Reading 1	2	30-31	Unsigned 16-bit integer
Reference frequency 2	2	32-33	Unsigned 16-bit integer
Reading 2	2	34-35	Unsigned 16-bit integer
Reference frequency 3	2	36-37	Unsigned 16-bit integer
Reading 3	2	38-39	Unsigned 16-bit integer
Therm	2	40-41	Unsigned 16-bit integer

Latitude and Longitude are 1/10,000 of a minute.

5.1.1 Exported samples file

The sensor sample selected for real-time transfer is also stored in the exported samples file. That file can be named using the EXPORT --o command.

The records in the file are a string of characters consisting of an ECO Puck raw sample as received from the sensor, with comma delimited GPS date, GPS time, Latitude and Longitude appended. A newline is appended.

Example of a full raw sample with 9 data columns and the system info:

99/99/99 99:99:99 532 4130 650 4130 460 4130 525,

2011/06/28, 17:05:45.430877, 37.410927833333325, -122.00422849999999

That sample would parse into:

Raw date	Raw time	ref	value	ref	value	ref	value	therm
99/99/99	*99:99:99*	*532*	*4130*	*650*	*4130*	*460*	*4130*	*525,*
GPS date	**GPS time**	**Latitude**			**Longitude**			
2011/06/28	*17:05:45.430877*	*37.410927833333325*			*-122.00422849999999*			

5.1.2 Full samples file

All raw samples received from the ECOPuck sensor are stored in the all samples file. The all samples file is created by default during the initialization. Users can create a new file by using command “ECOPuck --output”. The format of records in the full samples file is identical to the format in the exported samples file.

6. Configuration

ECOSensor is initialized to communicate with the ECOPuck via RS232 using the serial port and communications settings specified in the configuration XML file /home/smc/roots/200_ECO/WetLabsSensor-config.xml.

6.1 Common XML file keys to edit

6.1.1 For interfacing to a Wet Labs ECOPuck:

sensorCmdToken set to default **ECOPUCK**

sampleStart set to default “99/99/99“. This is the unique ASCII string of characters at the beginning of each data sample sent from the ECOPuck.

serverName and **payloadDescription → readableName** set to default ECOSensor.

6.1.2 For controlling the power supply to the ECOPuck:

powerOnVerb set to default power switch 3 to 1.

powerOffVerb set to default power switch 3 to 0.

6.1.3 For communicating with the ECOPuck via RS232:

serialPortSettings default serial port /dev/ttyLX4, 19200 baud, 8 data bits, 1 stop bit, no parity,

6.1.4 For non-standard ECOPuck installations, including multiple units, the following XML file keys will need to be modified:

serverName

payloadDescription → boardID

payloadDescription → taskID

payloadDescription → readableName

6.2 Sample Configuration XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="7">
<rootlevel class_id="0" tracking_level="0" version="0">
        <sensorCmdToken>ECOPUCK</sensorCmdToken>
        <sampleStart>99/99/99</sampleStart>
        <serialPortSettings class_id="1" tracking_level="0" version="0">
                <portName>/dev/ttyLX4</portName>
                <baudRate>19200</baudRate>
                <parity>NONE</parity>
                <characterSize>8</characterSize>
                <stopBits>1</stopBits>
                <flowControl>NONE</flowControl>
                <rts>1</rts>
                <dtr>1</dtr>
                <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
        </serialPortSettings>
        <maxString>1000</maxString>
        <readTimeoutMs>50</readTimeoutMs>
        <ecoSensorEventName>EcoPuckSample</ecoSensorEventName>
        <ecoSensorEventType>EcoPuckType</ecoSensorEventType>
        <ecoSensorEventDomain>LiquidR-SMC</ecoSensorEventDomain>
        <serverName>EcoSensor</serverName>
        <serverKind></serverKind>
        <dateSetDirectly>1</dateSetDirectly>
        <dateVerb>sudo date --utc {mm}{dd}{HH}{MM}{yyyy}.{SS}</dateVerb>
        <gpsEventSettings>
                <eventName>GPSSample</eventName>
                <eventType>
                        <typeName>GPSSample</typeName>
                        <domainName>LiquidR-SMC</domainName>
                </eventType>
        </gpsEventSettings>
        <eventTransceiverSettings class_id="2" tracking_level="0" version="0">
```

```
                <eventServiceName>EventManager</eventServiceName>
                <eventServiceKind></eventServiceKind>
                <receiveEnable>1</receiveEnable>
                <aliasToEvent class_id="3" tracking_level="0" version="0">
                        <count>0</count>
                        <item_version>0</item_version>
                </aliasToEvent>
                <eventFilterEnabled>0</eventFilterEnabled>
                <eventNameFilter class_id="4" tracking_level="0" version="0">
                        <count>0</count>
                        <item_version>0</item_version>
                </eventNameFilter>
                <eventTypeFilter class_id="5" tracking_level="0" version="0">
                        <count>0</count>
                        <item_version>0</item_version>
                </eventTypeFilter>
        </eventTransceiverSettings>
        <pmClientSettings class_id="11" tracking_level="0" version="0">
                <serviceName>PayloadManager</serviceName>
                <serviceKind></serviceKind>

        <deviceAddedEventName>PayloadDeviceAdded</deviceAddedEventName>

        <deviceChangedEventName>PayloadDeviceChanged</deviceChangedEventNa
me>

        <deviceRemovedEventName>PayloadDeviceRemoved</deviceRemovedEventN
ame>
                <commonEventType>PayloadRegistrationEvent</commonEventType>
                <commonEventDomain>LiquidR-SMC</commonEventDomain>
                <payloadDescription class_id="12" tracking_level="0" version="0">
                        <boardID>32</boardID>
                        <taskID>18</taskID>
                        <description>Eco Sensor and distribution</description>
                        <ior>(This will be overwritten)</ior>
                        <readableName>EcoSensor</readableName>
```

```
            </payloadDescription>
        </pmClientSettings>
        <outFileName>ecosensorout.txt</outFileName>
        <recordFileName>ecorecord.txt</recordFileName>
        <powerOnVerb>sudo SetPowerSwitch 3 1</powerOnVerb>
        <powerOffVerb>sudo SetPowerSwitch 3 0</powerOffVerb>
        <smcTarget>1</smcTarget>
        <scriptFileName>ecoscript.txt</scriptFileName>
        <commandCompleteEventSettings class_id="1" tracking_level="0" version="0">
            <eventName>CommandComplete</eventName>
            <eventType class_id="2" tracking_level="0" version="0">
                <typeName>SendCommandCompleteEvent</typeName>
                <domainName>LiquidR-SMC</domainName>
            </eventType>
        </commandCompleteEventSettings>
    <sampleEventTotalColumns>9</sampleEventTotalColumns>
    <sampleEventColumnOne>TypeString</sampleEventColumnOne>
    <sampleEventColumnTwo>TypeString</sampleEventColumnTwo>
    <sampleEventColumnThree>TypeUInt16</sampleEventColumnThree>
    <sampleEventColumnFour>TypeUInt16</sampleEventColumnFour>
    <sampleEventColumnFive>TypeUInt16</sampleEventColumnFive>
    <sampleEventColumnSix>TypeUInt16</sampleEventColumnSix>
    <sampleEventColumnSeven>TypeUInt16</sampleEventColumnSeven>
    <sampleEventColumnEight>TypeUInt16</sampleEventColumnEight>
    <sampleEventColumnNine>TypeUInt16</sampleEventColumnNine>
        <txRetries>2</txRetries>
        <rxWaitTimes>5</rxWaitTimes>
        <rxDelaySeconds>30</rxDelaySeconds>
        <upperAvgLimit>1000</upperAvgLimit>
        <shutdownEventSettings class_id="5" tracking_level="0" version="0">
            <eventName>Shutdown</eventName>
            <eventType class_id="6" tracking_level="0" version="0">
                <typeName>EventBase</typeName>
                <domainName>LiquidR-SMC</domainName>
            </eventType>
```

```
        </shutdownEventSettings>
        <telemetryMaxSamples>3</telemetryMaxSamples>
        <wakeupDelaySeconds>20</wakeupDelaySeconds>
        <runDelaySeconds>30</runDelaySeconds>
        <verboseLevel>0</verboseLevel>
</rootlevel>
</boost_serialization>
```