Ventana Balefire Workspace

Operations Manual

Subject: *Greensea Systems, Inc. delivered a system upgrade for the* Ventana *ROV, owned and operated by the Monterey Bay Aquarium Research Institute, in May of 2015. This upgrade included software components built on Greensea's core technology, openSEA, and Greensea's commercial ROV control system product, Balefire. Greensea developed specific configurations of the Balefire workspace for the* Ventana *system. This document outlines the basic operation of the* Ventana *software control system and the* Ventana *configurations of the Balefire Workspace.*

Revision History

Revision 012

Revision	*Initials*	*Date*	*Comments*
001	Ljd, Meb	06/03/15	Document created.
002	Ljd, Meb	05/07/15	Updated sections, made edits.
003	Ljd, Meb	06/03/15	Updated sections, made edits.
004	Ljd, Meb, Sno	06/03/15	Updated sections, made edits.
005	Ljd, Meb	05/30/15	Added signal mapper content, made edits.
006	Ljd, Meb	06/03/15	Added intro content, screenshots, edits.
007	Ljd, Meb, Sno	06/04/15	Updated sections, updated screenshots, made edits.
008	Mg, Meb, Sno	06/04/15	Edited content.
009	Meb	06/05/15	Proofread, edited.
010	Clr	06/07/15	Edited content.
011	Meb	06/10/15	Proofread, edited.
012	Meb	06/30/15	Edited.

Table 1: Revision history.

Table of Contents

Index of Tables

Illustration Index

Ventana Graphical User Interface

Operations Manual

1 Introduction

This document outlines the basic operation of the *Ventana* Balefire Workspace. Each section briefly details the workspace display components and their functions.

This document outlines Greensea's software philosophy and the basic operation of the *Ventana* Balefire Workspace. Each section briefs the operator on the workspace components and their functions. It was written for general operators under the assumption that the operator is familiar with the use of basic desktop software, modern computer hardware, and *Ventana* ROV operations.

1.1 Internal Applications

Many applications run in the background to facilitate the use of the *Ventana* Balefire Workspace. Without these internal applications the Balefire workspace could not provide navigation data or control functionality to the operator. These applications communicate to each other over Ethernet using a communications library named LCM. This system allows all of the applications to communicate to each other as long as there is a network connection between the applications.

1.2 Internal Application Configuration

Several of the applications in the *Ventana* system require configuration files for execution. Many applications will run without a path to the configuration file and many will require the application configuration to run. In all cases however, the configuration files ensure proper configuration for the *Ventana* application.

Application arguments specify the path to the configuration files required for each application. These configuration files are well documented and are written in the yaml (YAML Ain't Markup Language) format to provide a human-readable and easily edited file. As with any configuration file format, yaml requires specific syntax, structure, and punctuation. Refer to the yaml website for more specific information on this format (yaml.org).

There are four applications that require specific configurations for the *Ventana* system: the openSEA modbus interface application, openMNGR, openCMD, and openINS.

1.2.1 openSEA Modbus (gss_modbus) Interface Configuration

There are several modbus devices in the *Ventana* system. The openSEA modbus application, gss_modbus, is used for all of these interfaces. This application is highly configurable. Application arguments define the basic parameters of the interface while configuration files for each interface define the data contained in the input and output of the interface.

The gss_modbus configuration files for the topside interfaces are contained in the directory of the topside computer.

Operators should typically have no reason to modify these configuration files.

1.2.2 openMNGR Configuration

openMNGR is the openSEA application used for high-level motion control and vehicle control state arbitration. openMNGR communicates with the topside system to receive joystick commands, operator interface commands,

autopilot commands, and mission plans to calculate determine the single degree of freedom commands required.

It relies on several configuration files to specify its communication channels, input scaling parameters, and application alarms. The current system waypoint file is also contained within the openMNGR configuration file set.

The openMNGR configuration files are contained in the openmngr_config directory. Operators should typically have no reason to modify these configuration files.

1.2.3 openCMD Configuration

openCMD is the openSEA application used to calculate the low-level single degree of freedom maneuvering solution based on the high-level instructions provided by openMNGR. Several configuration files are required by the openCMD application to configure communication channels, application execution parameters, and the profiles for the individual degrees of freedom.

opencmd_app.yml is the main configuration file for the application, providing openCMD with the set of configuration files needed for use. Operators should typically have no reason to modify these configuration files.

1.2.4 openINS Configuration

openINS is the navigation engine within *Ventana* that provides a full state estimate of the system based on the available navigation sensor inputs. openINS uses a multi-state Kalman filter, as well as several pre-filters, to fuse aiding sensor data with high-rate inertial data provided by the Octans IMU/Gyro. It requires several configuration files to specify application parameters, communication channels, sensor parameters, and system transforms.

The openINS configuration files are contained in the openins_config directory. Operators should typically have no reason to modify these configuration files, though it may be required to adjust the sensor transforms should physical sensor placement on the vehicle change.

2 Concept of Operations (CONOPS)

The *Ventana* software control system provides a vehicle control and navigation system for ROV operations. Visualization and the operator interface is provided by the Balefire Workspace. The Balefire Workspace simplifies interactions between the operator and the vehicle. The following sections will give the reader an understanding of how the *Ventana* control system is designed to be used and how the operator can get the most of out it.

2.1 Navigation

Vehicle navigation is an integral component of the *Ventana* control system and is implemented through openINS and various openSEA device interfaces. openINS provides a complete software implementation of an aided Inertial Navigation System (INS) suited primarily for unmanned underwater vehicles. The software package runs on Greensea's proprietary technology, openSEA, and is part of the foundation the Balefire control and navigation system is built upon.

openINS is the core computational engine, the true muscle, behind both commercially packaged and vehicle-integrated navigation systems. It offers a robust and scalable computational platform that utilizes the device library in openSEA to support hundreds of different devices, many of which are navigation sensors.

openINS provides a robust multi-state Kalman filter that fuses inertial measurements from the IMU with data from aiding sensors to create an accurate final solution of the vehicle's state. All vehicle state information is the result of fusing independent state measurements with inertial process samples with the exception of reference-state

information such as altitude.

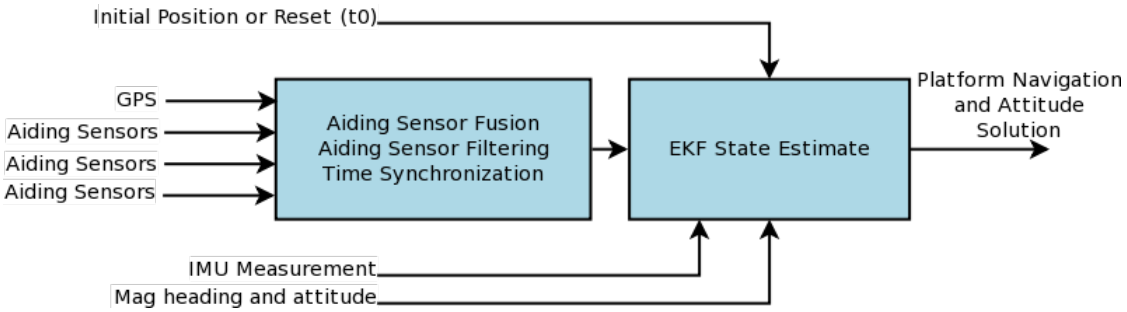


Illustration 1: General openINS navigation scheme implemented in Ventana.

The following table provides a summary of the contributions to the overall navigation estimate from each sensor.

Sensor	*DOF Data*	*Limits*	*Units*	*Orientation*
GPS	X, Y, Z	Surface only Low update rate	lat, lon	T^{B}_{gps}
USBL	X, Y, Z	Large errors Low update rate	lat, lon	T^{B}_{usbl}
Depth	Z	High-frequency noise	m	T^{B}_{dpt}
Altimeter	Zref	High-frequency noise	m	T^{B}_{dpt}
AHRS Compass	Φ, θ, Ψ $\dot{\Phi}, \dot{\theta}, \dot{\Psi}$	Magnetometer based	rad rad/sec	R^{b}_{AHRS}
IMU / Gyro	$\ddot{x}, \ddot{y}, \ddot{z}$ $\dot{\phi}, \dot{\Theta}, \dot{\psi}$	High drift Entirely dependent on initial conditions	m/sec^2 rad/sec	R^{b}_{imu}, T^{B}_{imu}
DVL	$x_{dot}, y_{dot}, z_{dot}$	Requires bottom lock Requires stable platform	m/s m/s m	T^{B}_{dvl}

Table 2: Sensor contributions to navigation system.

2.2 Coordinate Frames

There are several coordinate frames considered in a vehicle navigation frame. Greensea employs a north-east-down (NED) coordinate system with the x-axis pointing towards the front of the vehicle along the axis of the heading, the y axis pointing out the right side of the vehicle, and the z-axis pointing down. This same coordinate frame is defined for the *Ventana* instrument frame. The *Ventana* frame of reference is located at the geographic center of *Ventana* and is to where all the individual aiding sensor frames are referenced through independent transforms. The final navigation solution references the vehicle frame of reference to world coordinates and attitudes through Euler transforms and, if available, geographical reference points.

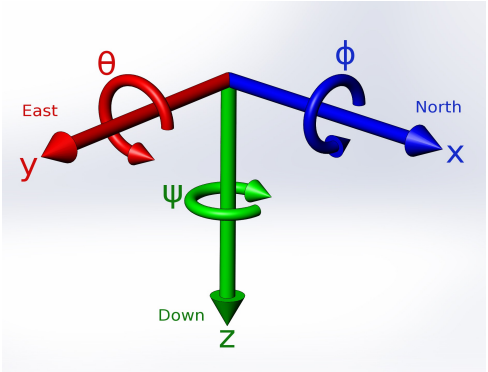


Illustration 2: NED coordinate frame.

2.3 Vehicle Control

The *Ventana* control system provides an intuitive operating environment for the vehicle with several modes of control to make precision flying easier and less fatiguing for the operator.

2.3.1 Operator Interfaces

The *Ventana* control system provides both joystick control as well as on-screen controls for flying the vehicle. Typically, operators will fly the vehicle from the joystick in normal operating modes. The Balefire Workspace provides additional functionality through the main pilot interface as well as through autopilots for programming missions.

The *Ventana* control system provides several modes of operation and several different operator interfaces for working with those modes. Operators will find different interfaces work better for different modes of operation and some operators will find personal preferences between interfaces. The system is designed to give operators options to choose the best mode of operation and the best interface for various tasks.

2.3.2 Open-Loop Control

In open-loop control, the operator directly controls the thrusters on the vehicle with joystick and vertical slider on the joybox. In open-loop mode, no sensor feedback is used to augment the motion of the vehicle. This is the most simple means to control the vehicle.

2.3.3 Autopilots

The *Ventana* system provides several autopilots to make the system easier and more effective to operate. Autopilots are single degree of freedom controllers that control a single motion of the vehicle. Autopilots can be enabled or disabled as required. All autopilots are single-input-single-output controllers, meaning they use the feedback from a single state of the system to control the thruster outputs required to achieve the desired state of that state.

The autopilot controls on *Ventana* are implemented through classic controllers and are designed carefully to provide wide bandwidth operation. This means that the controllers should provide satisfactory performance under a very wide set of operating conditions and not require tuning or reconfiguration if system characteristics such as buoyancy, payload, and manipulator position change.

Greensea has carefully designed the interface of each autopilot to provide the operator a flexible and intuitive tool for vehicle positioning. The following table lists the autopilots within *Ventana*, their controlled degree of freedom (DOF), and their interfaces.

Autopilot	*DOF*	*Feedback*	*Joystick Function*	*Other Controls*
Heading	Heading (PSI)	openINS heading	Adjusts setpoint	Incremental jog Direct input
Depth	Depth (Z)	openINS depth	Adjusts setpoint	Incremental jog Direct input
Altitude	Altitude (Z-Ref)	openINS altitude	Adjusts setpoint	Incremental jog Direct input
Station Keeping	Lateral position (X, Y)	openINS position	Disables autopilot	Incremental jog "GoTo position"
Cruise	Lateral velocity (Xdot, Ydot)	openINS velocities	Sums setpoint	Incremental jog

Table 3: Autopilot function summary.

2.3.4 Fly-by-Wire

When the Wright brothers built the first airplane, they used control rods and levers attached to each control surface to fly the airplane. This mode of control went most unchanged until airplanes became more powerful and we required more performance from them. Then engineers developed Fly-by-Wire (FBW) control for planes such that the pilot was no longer controlling the control surface, she was controlling the *motion* of the airplane. This was implemented through autopilots and allowing the pilot controls to control the autopilots instead of the actual control surfaces. The rationale was simple: The pilot has better things to do than figure out what control inputs were required to execute a precise motion, and computers were better at it anyway.

The same is true with the FBW system in *Ventana*. *Ventana* has a powerful system with tremendous capability and pilots have better things to do than worry with what thruster mix is required to execute a particular motion or keep a particular attitude. The FBW system in *Ventana* is a *concept* of operations rather than a mode of operations and it provides an extremely effective means of vehicle control.

Under most operating conditions, enabling all autopilots (heading, depth/altitude, and station keeping) and using controls to pilot the vehicle is the most effective and easiest means of flying *Ventana*. With the autopilots enabled, the pilot flies the vehicle with either the joystick or on-screen controls by controlling the set point of the autopilots. When using the joystick, there is no difference operationally than how the pilot traditionally has flown the vehicle, the vehicle just responds better. Most pilots describe the control as "stiff".

With FBW, operators can significantly shorten the distance between what they are thinking and what the vehicle is doing.

2.3.5 Mission Automation

Often, pilots are required to conduct surveys, grid searches, transects, mosaics, or repeat dives previously conducted *exactly*. The *Ventana* control system provides an easy way to automate tasks by defining a mission and allowing the vehicle to autonomously conduct the mission. Missions are defined by specifying waypoints, vehicle behavior at and between each waypoint, and arranging waypoints in order of execution.

With mission automation, operators can pre-plan dives and conduct dives using automation as a tool to ensure the dive goes as planned. Mission automation provides precision to *Ventana* operations. The Balefire Workspace allows operators to graphically plan missions, save missions, and recall missions giving operators a new set of tools for conducting dives, repeating dives, and sharing dives.

2.4 Workspace Status Feedback

2.4.1 Color Conventions

Within the Balefire Workspace, the operator interacts with several buttons, gauges, dials, and controls. The controls within the Workspace are always illuminated in either blue, green, yellow or gray. The colors indicate at a glance the current status of the system. A brief description of each color is given below:

- **Blue:** Blue is seen on soft buttons, sliders, and other user controls. Blue indicates all associated processes with that button are running properly and when the button is clicked, the associated action should, as best the software can determine, take place.
- **Green:** Green is seen on dials, buttons, sliders, LEDs and other controls. When a soft button is clicked and turns green, it means that the associated action has taken place successfully based on the feedback data.
- **Yellow:** Yellow indicates the feedback of a particular command does not correspond to the command. This is commonly seen on relay functions when the relay board is not responding to the command and

on autopilot functions when conditions are preventing the system from enabling an autopilot function though the operator has commanded it.

- **Gray:** Gray indicates a particular function is not yet communicating, or that the function is currently offline. Buttons will often be gray only as the software is launched, but in certain circumstances, communications go down and buttons will become gray and unclickable. If communications are lost, the buttons remain in the state they were in at that instant. If a relay was on before comms are lost, it will remain on while the vehicle is powered. When communications are restored, the button will re-illuminate blue and become functional once again.

2.4.2 Communication Status

The Balefire Workspace communicates with a number of different subsystems. Many of these subsystems also communicate with other systems and devices. It is important to let the operator know the status of these communications channels as status and command functions rely on the integrity of these communications channels for proper operation. The Balefire workspace uses a consistent theme and several different devices to provide status on communications channels.

- **LEDs:** On the status bar of the workspace, as well as throughout the workspace in several locations, individual LEDs provide the status of specific communications channels. A green LED indicates a valid and healthy communications link while a red LED indicates invalid communications.
- **Gray text:** Throughout the workspace, gray text indicates the data for the particular device is missing and thus suggests communications for that device or component are not available.
- **Data View information:** Data View, also called Scope View, provides detailed information on the devices and modules within the Balefire system. The Scope View also *only* shows devices that are communicating. "STAT" messages are that come from a device and "CMD" messages go to device. "STAT" messages are the best messages to use for an indication of a communicating device. Inside of each message, the "publish_count" field indicates each message sent. If the "publish_count" field is incrementing, it indicates a communicating device.

2.4.3 Alarm Notification

The Balefire Workspace provides a configurable and comprehensive alarm management and notification system. Operators can monitor any signal in the system for alarm conditions using this utility. Alarms are not static and display the current status of the particular signal they are monitoring but a history of the alarms is provided. Alarms have priority levels and color code these priority levels from "Info" level to "Fatal" level.

Another useful application of the Balefire alarm system is to simply monitor signals for specific conditions. Once a condition has been met and an alarm generated, the event will be in the alarm history.

2.4.4 LED Indicators

The Balefire Workspace uses graphical LED indicators throughout the application to display binary health and status information. The standard convention for LED indicators is that green indicates a favorable, or healthy, condition and red indicates an unfavorable, or fault, condition. This can often be confusing as positive signals can represent negative meaning, such a positive digital signal indicating a water ingress. Uniformly in the Balefire Workspace though, green means good and red means bad.

2.4.5 Numeric Representations and Resolutions

The Balefire Workspace and Balefire software system uses only two fundamental storage types for numbers:

double-precision floating point and Boolean. Double-precision values are commonly referred to as "Analog" values while Boolean values are commonly referred to as "Digital" values. The Balefire system also represents Boolean values as binary numbers with '1' representing "TRUE" and '0' representing "FALSE".

2.5 Task-specific Organization

The Balefire Workspace is designed to provide operators task-specific views so that the data and functions required for the particular task can be most prominent and accessible. The components of the workspace are all completely re-sizable. Separation bars divide the workspace components into functional groups. When the workspace component separation bars are hovered over, they change to yellow and a double-arrow cursor appears. Click-and-drag the separator to the desired size. The components can be sized to zero to make more room for components of particular operator interest.

2.6 Data Logging

The Balefire Workspace makes extensive use of data logging and the data logging system is a fundamental component of using the workspace effectively. Logs can be used to archive job data and are also an effective troubleshooting and support tool. The data logging system logs every variable in the system with a timestamp. The Balefire Workspace also allows replaying logs. When requesting support from Greensea, it is helpful to have a log of the particular event in question.

3 Balefire Workspace Overview

Upon startup, the workspace display will default to the Mission View display screen. This screen contains a split-screen with video and sonar display, as well as a grid overlaying a map.

Across the top is the Navigation Bar, which holds toolbar buttons that control switching to other high-level pages, such as Alarms, Signal Viewer, Diagnostics, Signal Mapper, and the Pilot Workspace (the “Goat” Screen). The flame icon returns to the Balefire Mission View page. Immediately to the right of the navigation toolbar buttons are the main nav and control status LEDs, followed by the display of Alarm Manger messages.

- The Nav Status LED indicates whether the *Ventana* is powered-up from the topside Power Tray.
- The Control Status LED indicates the vehicle main power is on.

These elements always appear across all pages of the workspace.

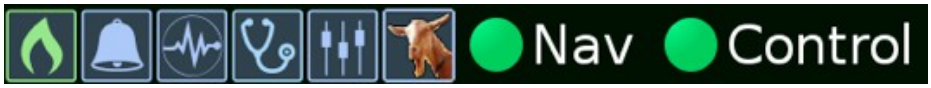


Illustration 3: The Navigation Bar and Navigation and Control Communication LEDs.

The bottom portion of the gridded map display features the Navigation Tools Menu for vehicle mission planning, marker management, maps and charts, logging, and playback.

The left side of the screen contains gauges for depth and altitude, a compass rose, and the autopilot control functions.

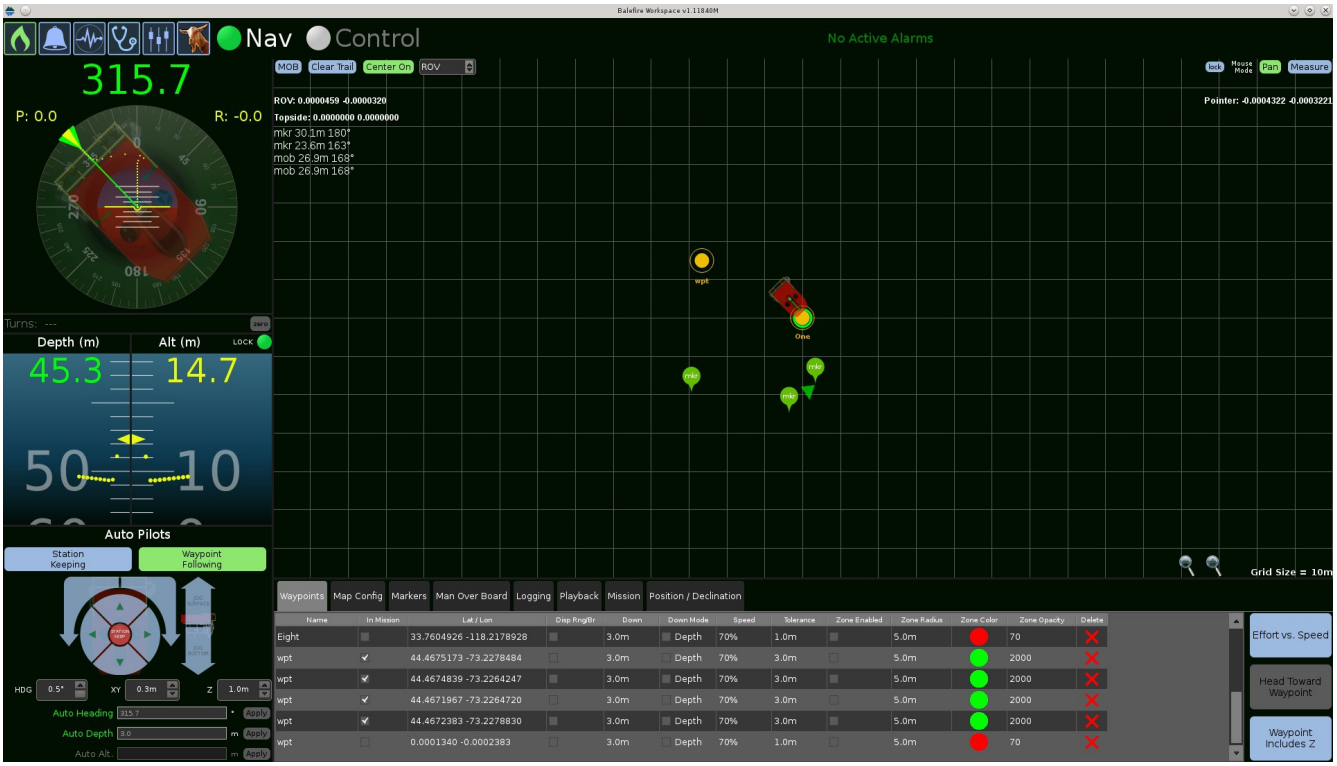


Illustration 4: Workspace in Mission View

4 Heading and Navigation Display

Many of the important tools you need for monitoring the vehicle's position sit on the left side of the display. The Navigation Bar includes the six critical shortcuts to Mission View, Alarm Manager, Data Plotter, Signal Mapper, Device Diagnostics, and the “Goat” Screen. Two circular LED lights that sit to the right of the Navigation Bar represent the Navigation and Control communication status. The altitude and depth gauges and compass rose are displayed below these lights.

4.1 Compass Rose

The compass rose displays heading, pitch, and roll. It also shows small dots that create a 'breadcrumb trail' of the vehicle's heading history. The pointer displays the current heading in yellow text while there is vehicle communication. When Auto Heading is enabled, the heading text turns green and a green pointer is displayed on the compass rose to indicate the Auto Heading set point.

There may be multiple compass rose element displayed in the workspace and each may be tied to a particular heading input (e.g. Octans /Fog or PNI compass). Right-clicking on the compass rose brings up a sub-menu that enables other items, such as displaying the topside (ship) heading as a ghost outline on the compass rose over the ROV, the option to choose between north-up and vehicle-up orientation, and options to speed up or slow down the history display.

NO COMMS displayed on the compass rose indicates a problem determining vehicle attitude. This can indicate either that openINS is not publishing a navigation solution, or that the "attitude ok" flag in the navigation solution has been set to false. A false "attitude ok" flag usually indicates that either OpenINS is not receiving data from an attitude sensor at an appropriate rate, or the data that it is receiving is invalid.

The Nav LED indicates whether or not OpenINS is publishing a navigation solution. A red Nav LED may indicate that openINS is still initializing, that it is not receiving data from any of its sensors, or that the application has crashed. The Control LED indicates whether or not the sub power button has been pressed. A red Control LED may indicate that the sub is powered off, or that the CSI WAGO application is no longer publishing data.

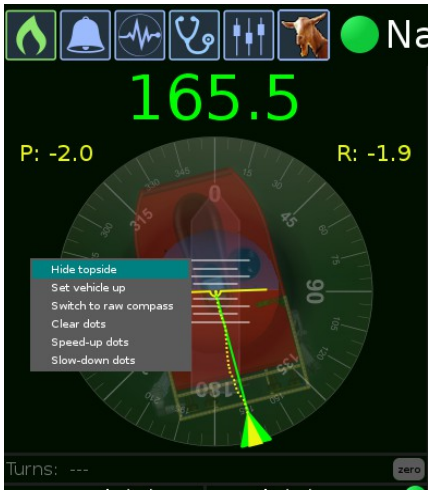


Illustration 5: The compass rose with heading, pitch, roll, open sub-menu, and turns counter.

4.2 ROV Turns Counter

A resettable turns counter, which tracks ROV turns relative to the ship, may be displayed below the compass rose as seen in Illustration 5. The **Zero** button manually resets the turns counter.

4.3 Altitude and Depth Gauges

The altitude and depth gauges sit right below the compass rose. These gauges display the vehicle's current altitude and depth in meters relative to sea floor when the vehicle has bottom lock. The green LED on the altitude gauge shows if bottom lock is valid. A yellow arrow indicates your current altitude and depth. Note the dashed 'breadcrumb trail', which displays the vehicle's depth and altitude history to allow the operator to better gauge the vehicle's vertical trajectory. This trail is also yellow with active readings. This displays the history and will remain stationary (in gray) at the last recorded altitude when communications or bottom lock is lost. Green text and a green line show the Auto Depth or Auto Altitude set point.

Right-clicking on the gauge will reveal a menu to manage the breadcrumb trail. The options include clear dots, speed-up dots and slow-down dots. This gives the operator the ability to view the vehicle's attitude graphically in a more configurable manner.

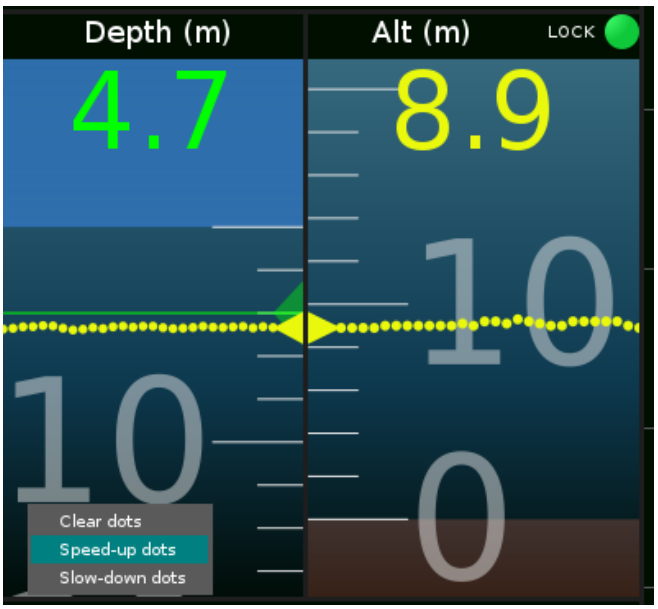


Illustration 6: Depth and altitude gauges with bottom lock LED and sub-menu displayed.

4.4 The Map Heads-Up Display

The Map Heads-Up Display (HUD) is horizontally superimposed across the four corners of the grid view. The top left HUD of the map screen shows the vehicle position as well as the topside vehicle's position.

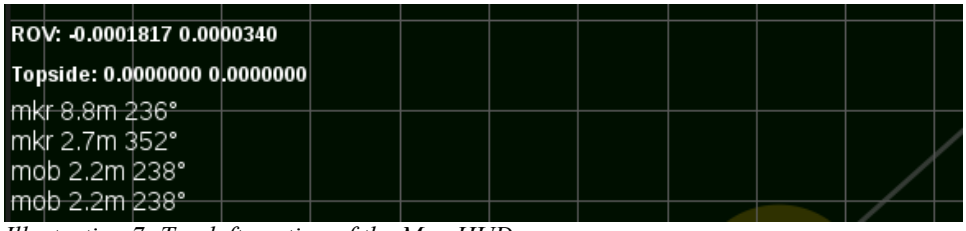


Illustration 7: Top left portion of the Map HUD.

The top right HUD displays the coordinates of the mouse cursor.

The bottom left HUD shows the coordinates of the measurement currently taken. Measurements can be taken at any given time and are outlined further in section 5.10. Once a new measurement is taken, it will be the sole measurement displayed in the bottom left Map HUD.

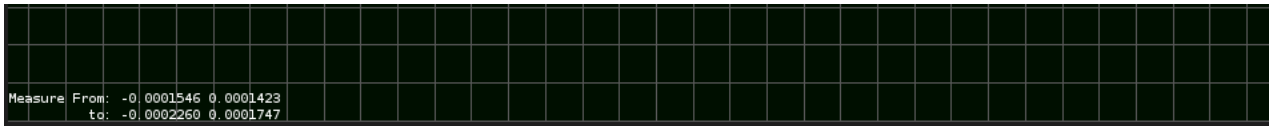


Illustration 8: Bottom left Map HUD displays measurements.

4.5 The Vehicle Graphic

The vehicle is represented on the workspace by a red 2-D *Ventana* graphic, as shown in Illustration 8.

As the vehicle travels, the path is tracked on screen, allowing you to view and record details of its travel history.

View of the vehicle can be changed with the tool buttons, which are represented by two magnifying glasses, located at the bottom of the map.

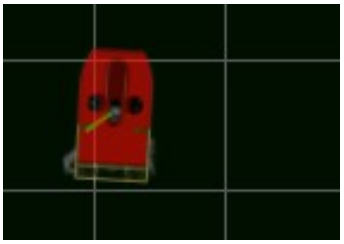


Illustration 9: The vehicle's graphical representation on the workspace.

Illustration 10: Magnifying glasses represent the zoom feature.

The center graphic for both glasses represents the level of zoom, expressed in the following table:

Magnification Level	*Command*
Zoom In	The “+” sign represents incremental inward zoom.
Zoom Out	The “-” sign represents incremental outward zoom.

Table 4: Zoom tools and their descriptions.

4.6 Man Overboard (MOB)

Man Overboard (MOB) is a special purpose marker. The **MOB** button is a shortcut to mark the vehicle's current location on the map.

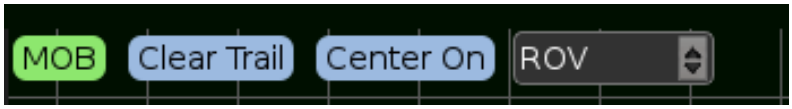


Illustration 11: MOB button creates a special purpose MOB marker.

Unlike markers which record only position, MOBs record both attitude and position of the vehicle at the point of being activated. Man Overboard markers appear as a green triangle.

The triangle points in the direction of the vehicle's heading when the MOB was created. Once created, the location data for the MOB marker is located in the MOB tabbed menu section. If enabled, the data will display on the map. The MOB tab section of the navigation menu has more details.

4.7 Clear Trail

The vehicle's traveled path is also referred to as a 'breadcrumb trail.' To reset the breadcrumb trail, select **Clear Trail** from the top center screen. This is not a map-clearing function. To clear the map, select “Clear Mission” from the function control menu.

4.8 Center On

The **Center On** button finds the vehicle and keeps it in view. If the vehicle moves off the visible screen area, clicking **Center On** will focus the map view on the vehicle. It will remain centered around the vehicle until the button is clicked again.

4.9 Pan

Selecting **Pan** from the top right of the main screen allows the vehicle operator to change map views. This feature allows the vehicle to be panned off the screen as long as **Center On** is not selected.

4.10 Measure

The **Measure** button changes the mouse mode so you can assess the distance and heading between two selected points. The GPS coordinates of the measured points will display on the bottom left of the screen. To enable this feature, select **Measure** and click the screen with the left mouse button, dragging the measurement out. The measurement will stay on screen until the screen is clicked on or another button selected.

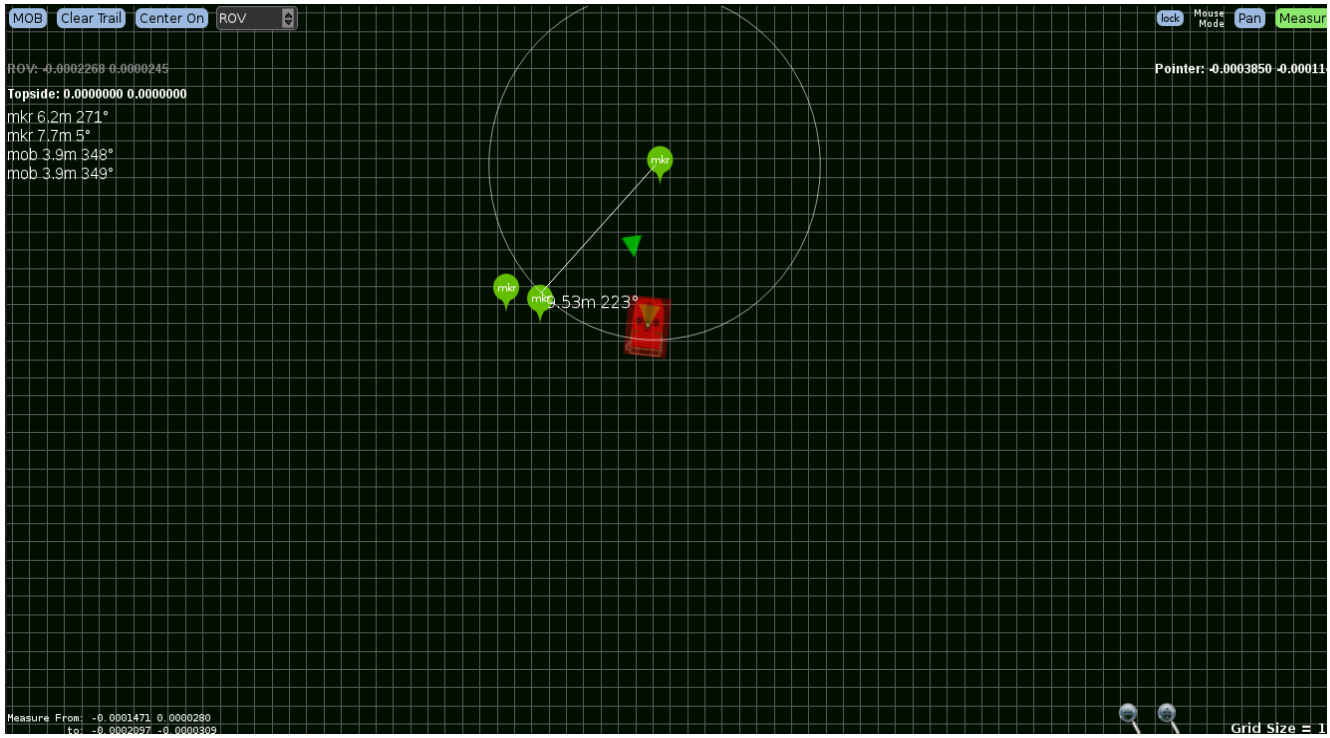


Illustration 12: A measurement is taken of two markers.

5 Autopilot Control Menu

The Balefire Workspace contains several autopilot functions that enable precise and accurate autopilot control over the vehicle. The autopilot functions can be used concurrently and can be used to fully automate vehicle motion, or to control the vehicle in fly-by-wire mode. The autopilot functions are located directly beneath the vertical velocity gauge.

Illustration 13: The Autopilot Control Menu.

Autopilot Workspace Command	*Description*
Station Keeping	Enables the vehicle to keep station at its current location. The accompanying arrow keys, overlaying the top down image of the ROV, allow incremental forward and lateral movement. These increments are set using the “XY” box.
Waypoint Following	Allows the operator to fly a mission traveling from waypoint to waypoint.
Auto Heading	Keeps the ROV at a given heading. The heading can be incrementally changed with a single mouse click.
Auto Depth	Keeps the ROV at a given depth. The “Z” box is used to set incremental changes in the positive and negative z direction. See Signal Mapping for more details.
Auto Altitude	Keeps the ROV at a given altitude. See Signal Mapping for more details.

Table 5: Autopilot commands and descriptions.

6 Navigation Tools Menu

The Navigation Tools Menu has nine tabs used to place markers, log data, and define important navigational settings. The tabs contain specific details for waypoints, map and chart configuration, markers, MOB markers, logging, missions, and vehicle position and declination.

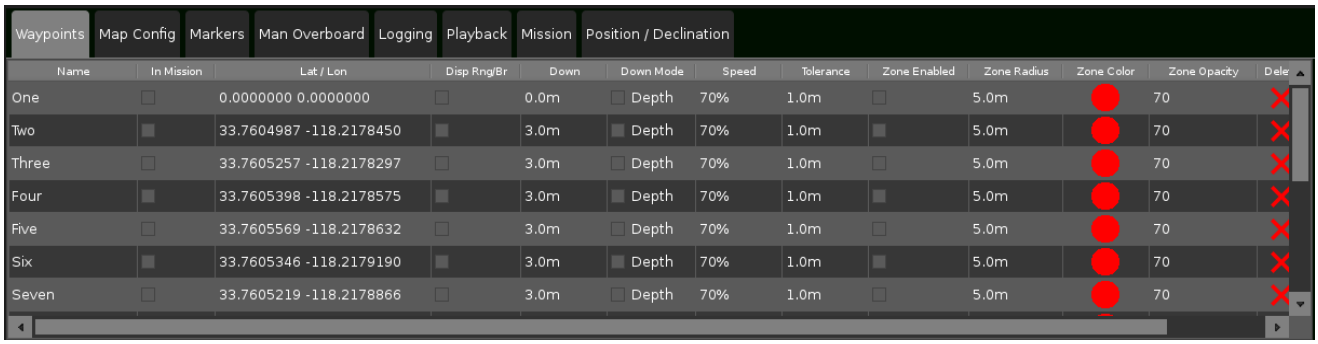


Illustration 14: The Navigation Tools Menu.

6.1 Waypoints

Waypoints are special purpose markers that designate the path of the vehicle. Waypoints are points specified by an X, Y, and Z position. The Z position of the waypoint may be either depth or altitude from the seafloor. The Waypoints menu page allows operators to set the parameters of each waypoint and configure the vehicle's behavior during a waypoint run.

Waypoint parameters are configurable during a waypoint run but configurations do not take effect until the waypoints are "pushed" to the vehicle as a mission. This process is detailed in the "Mission Planning" section.

Field	*Description*
Name	Unique identifiers can be given to a specific waypoint by editing its name field.
Latitude	The vehicle's latitude measured in degrees when placing the waypoint.
Longitude	The vehicle's longitude measured in degrees when placing the waypoint.
Display Range and Bearing	Selecting this box will show or hide the range and bearing to a waypoint in the top left HUD.
Down	The vertical position of the waypoint.
Down Mode	By clicking the box in the Down Mode menu, the status is modified from Depth (default) to Altitude. The reference is capable of holding a waypoint at a given depth from the surface or distance from the ocean floor to the ROV.
Speed	The percent of maximum speed at which the vehicle is traveling to reach the target waypoint.
Tolerance	Waypoint tolerance is the threshold (in meters) for having successfully met a waypoint in the vehicle's path. *Important Note: Tighter tolerances will require greater effort by the vehicle to achieve the waypoint. This may result in undesirable behavior in some instances.*
Zone Enabled	Selecting this box will enable or disable a hazard zone display for each waypoint.
Zone Radius	Sets the hazard zone radius around a waypoint in meters.
Zone Color	The color of the hazard zone around a waypoint. The color can be changed by double-clicking on the colored sphere and selecting a new shade from the menu that appears.
Zone Opacity	Zone opacity changes the shading of a hazard zone. A higher zone opacity will result in a more darkly shaded zone. This can be changed by double-clicking the zone opacity field, pressing the up and down arrows, or manually entering the desired value.
Delete	Click the red 'X' in the delete field to remove waypoints from the map.

Table 6: Waypoint field names and descriptions.

6.2 Map Configuration

The Map Configuration tab allows the user to configure waypoint defaults, safety zone defaults, vehicle, topside and USBL history, underlay charts on the map, and configure the HUD.

- **Defaults:** The Defaults tab allows the operator to set waypoint and safety zone default values. In the following illustration, "tolerance" is set to 1.0m, the speed at which a vehicle travels to a waypoint is 70% of maximum, "down" is set to 3.0m, and the default down mode is set to depth. The safety zones are not enabled by default, they are colored in 'port' red, have a radius of 5.0m and an opacity of 70%.

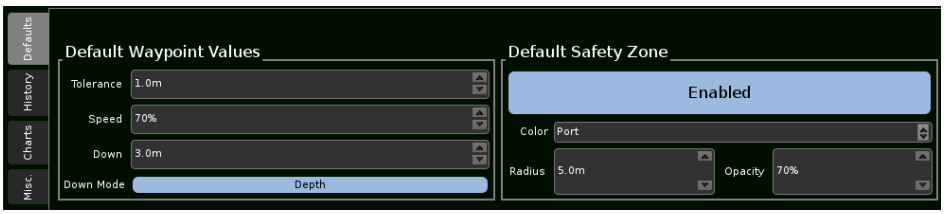


Illustration 15: Map configuration, defaults tab.

- **History:** The history tab allows the operator to set the display parameters of the travel history for the ROV, topside vessel, and USBL. This tab enables the history trails, sets the history trail color, sets the maximum length of the trail, and sets the opacity of the trails. The tab also allows the operator to set the display parameters of the USBL history, including the ping color and radius.

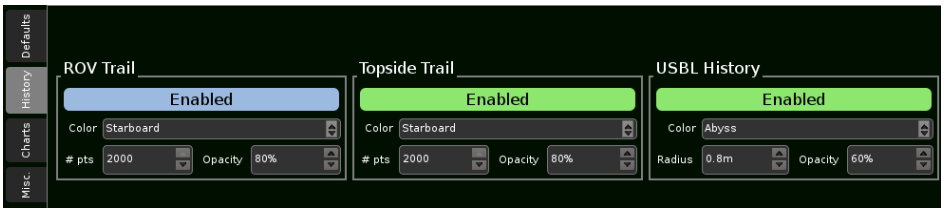


Illustration 16: Map configuration, "History" tab.

- **Charts:** The charts tab allows the operator to import files to underlay and modify the map view. Geo-referenced graphical underlay files can be imported or dragged and dropped onto the grid. To import a file, simply click **Import File**, and select the desired file from the browser.
- **Miscellaneous:** The "Misc" tab allows the operator to configure the Heads Up Display. The HUD text can be enabled or disabled and the text color can be altered. The HUD background color and opacity can be set from a menu of colors based on the operators preference for the task at hand. The units that the HUD displays for position, temperature and distance can be changed from the Displayed Units section of this tab. The operator can configure the display units for position, temperature, and distance through this tab.

6.3 Markers

Markers are reference tools to make note of specific locations on the map. They don't effect the vehicle's travel or the mission and do not clear with the mission. To add a marker right-click on the screen and select "Add Marker". Markers are normally green but turn blue when selected in the Markers menu. Operators can adjust the position of markers by moving them on the map or by editing their position parameters in the Markers menu.

The marker tab contains data for all markers on the map. Clicking the box in the Display Range/Bearing column will display the absolute heading of the marker with respect to the vehicle's position. To get rid of markers, right-click and select Delete, or highlight the marker you wish to remove in the Navigation Menu and click the red X in the "Delete" column.

Field	*Description*
Name	Unique identifiers can be given to a specific marker by editing its name field.
Latitude	The vehicle's latitude measured in degrees when placing the marker.
Longitude	The vehicle's longitude measured in degrees when placing the marker.
Display Range and Bearing	Selecting this box will show or hide the range and bearing of a marker in the top left HUD.
Down	The vertical position of the marker.
Down Mode	By clicking the box in the Down Mode menu, the status is modified from Depth (default) to Altitude. The reference is capable of holding a marker at a given depth from the surface or distance from the ocean floor to the ROV.
Zone Enabled	Selecting this box will enable or disable a hazard zone display for each marker.
Zone Radius	Sets the hazard zone radius around a waypoint in meters.

Field	*Description*
Zone Color	The color of the hazard zone around a waypoint. The color can be changed by double-clicking on the colored sphere and selecting a new shade from the menu that appears.
Zone Opacity	Zone opacity changes the shading of a hazard zone. A higher zone opacity will result in a more darkly shaded zone. This can be changed by double-clicking the zone opacity field, pressing the up and down arrows, or manually entering the desired value.
Delete	Click the red 'X' in the delete field to remove markers from the map.

Table 7: Configurable marker parameters

Illustration 17: Three markers, two green and one blue, are shown on the grid.

The "Markers" tab contains data for markers currently set on the map. Clicking the box in the "Display Range/Bearing" column will populate the absolute heading of the marker to the vehicle's position. To get rid of markers, right-click and select "Delete", or highlight in the list the marker you want to remove and click the red "X" under the "Delete" column.

6.4 Man Overboard (MOB)

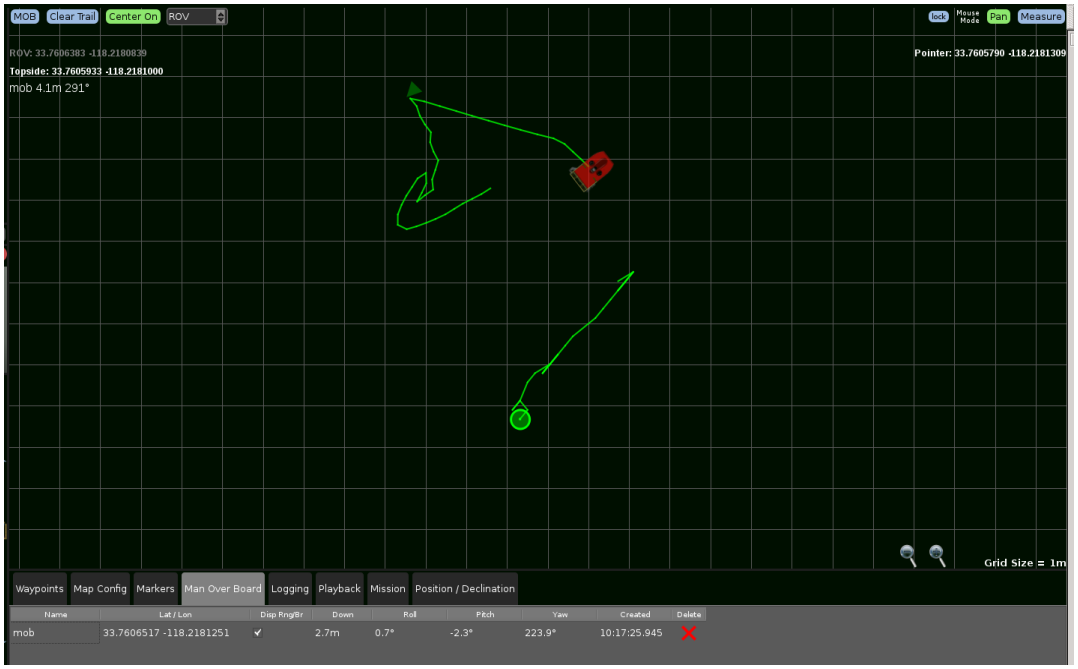


Illustration 18: The status of a MOB marker is shown on the upper left corner of the grid. The MOB marker can be seen on the map as a small green arrow pointing in the direction the vehicle was traveling when placed.

Man Overboard (MOB) is a special marker that, when activated, saves vehicle attitude and position. Select the “Man Overboard” tab to view location and configuration for MOB markers placed during a mission. Like waypoints, MOB markers can be enabled or disabled, and will display their status in the top left corner of the grid screen.

Specific Man Overboard attributes are:

Field	*Description*
Name	Unique identifiers can be given to a specific MOB by editing its name field.
Latitude	The vehicle's latitude measured in degrees when the operator keyed the MOB.
Longitude	The vehicle's longitude measured in degrees when the operator keyed the MOB.
Display Range and Bearing	Selecting this box will show or hide the range and bearing of a MOB in the top left HUD.
Down	The vertical position of the vehicle when the operator keyed the MOB.
Roll	The roll of the vehicle when the operator keyed the MOB.
Pitch	The vertical position of the vehicle when the operator keyed the MOB.
Yaw	The heading of the vehicle when the operator keyed the MOB.
Created	The system time when the operator keyed the MOB.
Delete	Click the red 'X' in the delete field to remove MOBs from the map.

Table 8: Configurable MOB parameters.

6.5 Logging

A powerful capability of the Balefire Workspace is the ability to log system data. Operators control data logging through the Logging menu tab.

Illustration 19: "Logging" is selected from the Navigation Tools Menu

There are two logging command options:

- **"Mark":** Write a note to mark a specific place in the log.
- **"Post":** Processes log files as CSV files. The recorded files are saved to the directory path shown in the browse window. To mark log data, select the file compression size for video feed and select the "Convert" tab to process the log file as a CSV file. When "Convert to CSV" is selected, a file menu will open and you'll be prompted to select a file to convert.

Illustration 20: A place in the log is marked for reference.

6.6 Playback

The Balefire Workspace supports playing back prerecorded log files. This is a powerful tool that can be used as a standalone customer deliverable, data analysis, and diagnostics. This is the primary tool Greensea uses to support customers in the field. With a recorded log file, Greensea can playback customer data and diagnose the system as if we were there in real time.

The Playback tab accesses the telemetry file by opening a browser window to the file system. Just push Play. To step through the log file, click the **Step** button. You can adjust the playback speed using the controls on this page.

6.7 Mission

Balefire manages the map view elements, including waypoints, as "Missions". The mission configuration tab provides functions for saving, recalling, and managing missions. The Mission tab has a field for naming a mission and buttons for mission planning.

Illustration 21: Buttons for mission planning are located in the "Mission" tab.

This feature is used to copy a mission back and forth between the workspace and the vehicle, save planned missions, and retrieve saved missions from a file. For a detailed description of what a mission is and how to use these functions, see the Mission Planning section.

6.8 Position/Declination

Operators can set the position of the vehicle manually through the Position/Declination tab. As well, this tab provides functions for adding a bias to the vehicle heading. This bias may be a manual input or automatically looked up from a magnetic profile based on the current location.

Note: The Declination Lookup feature requires a valid position. Unfortunately, it does not know if the current position of the vehicle is really valid. It will calculate the declination based on the value of the vehicle's current position as displayed in the workspace. If the vehicle's position is 0,0 (as if no real position has been provided) it will calculate the declination at 0,0.

This tab also provides a few useful features to operators such as copying the vehicle's position based on the ship's position or copying the vehicle's position from the system clipboard.

Illustration 22: "Position/Declination" is selected in the Navigation Tools Menu.

7 Mission Planning

A mission is a set of instructions that determines the vehicle's path and actions when in automated mode. A path is constructed with a series of waypoints for the vehicle to follow.

To access the map and plot waypoints, first select the Mission View button, which is represented by a flame icon. Mission View displays a topographical gridded map with the vehicle in its current location displayed as a red ROV graphic. Next, start adding waypoints.

Illustration 23: Mission View is selected from the Navigation Bar.

7.1 Adding Waypoints

From the Mission View screen, use your mouse to right-click on a desired waypoint location. Select “Add Waypoint”. The waypoint will appear in yellow and turn blue when selected. Waypoints can be moved with the mouse.

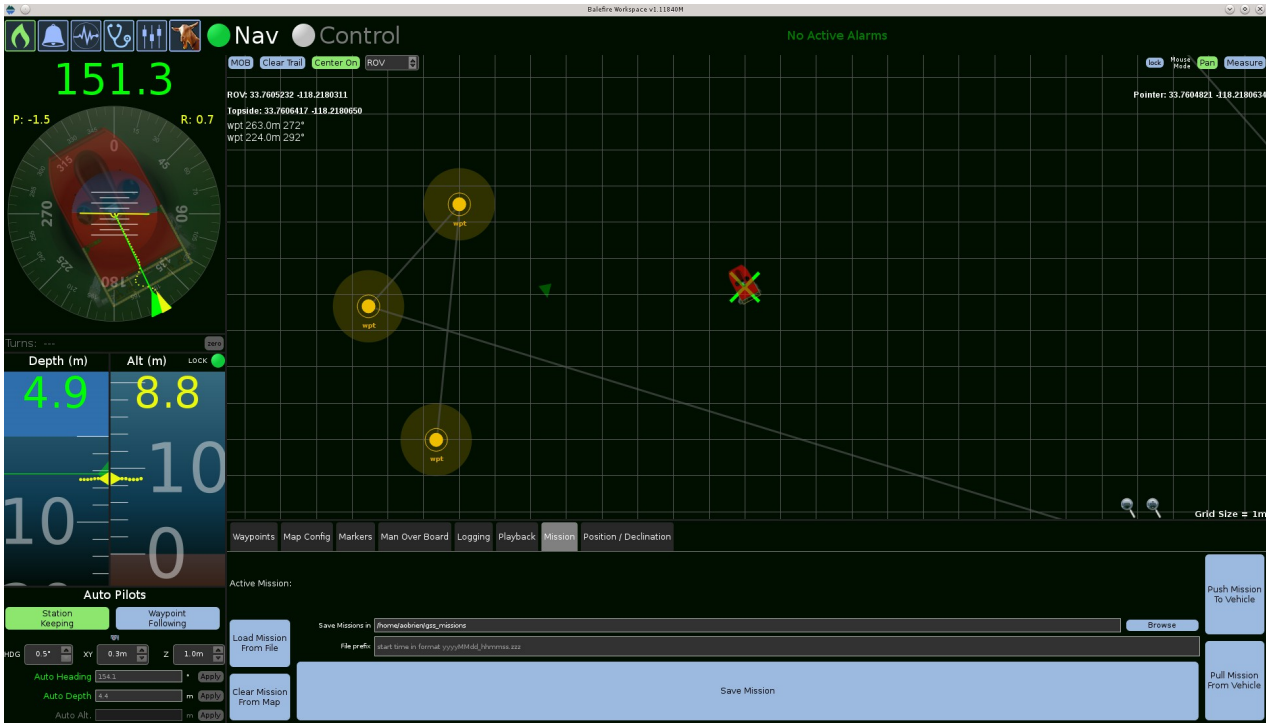


Illustration 24: Waypoints appear after being added to the mission plan.

A solid line trail will show the order the waypoints are visited as more are added. During missions, this path and the waypoint itself will change color to indicate mission progress.

Once the waypoints are added to the map, several positional editing options are available. Basic editing is covered in the next section.

7.2 Editing Waypoints

To move a waypoint with the mouse simply click-and-drag it to the desired location. Precise location and other waypoint details are edited in the “Waypoints” tab.

Waypoint data is stored in a .yml file. These files contain lots of data about waypoints. If you want to create a waypoint mission it's best to start from an existing waypoint configuration file.

Once waypoints are created and edited, the mission can be reloaded by going to the “Mission” tab in the Navigation Tools Menu, pressing **Load Mission From File**, and navigating to the modified configuration file.

```
created: 0
uuid: "{a6d08583-952f-4e56-93fd-a5b42a96aacd}"
type: WPT
name: One
note: ""
position: 73.2119 44.4758
display_relative_lbl: false
z_is_altitude: false
attitude: 0 0 0
safety_zone_enabled: false
safety_zone_meters: 5
safety_zone_color: "#ff0000"
safety_zone_opacity: 70
trail_enabled: false
trail_color: "#00ff00"
trail_max_pts: 500000
trail_opacity: 80
in_mission: true
speed: 70
tolerance: 1
task_list: ""
```

Illustration 25: The position of this waypoint and its name have been edited in the YAML file and then reloaded to the workspace.

The most important waypoint configuration options are “Down”, “Down Mode”, “Speed”, and “Tolerance”. The following table describes the basic waypoint features:

Waypoint Feature	*Description*
Down	The location of a waypoint relative to altitude or depth.
Down Mode	Changes down location from altitude to depth of a waypoint.
Speed	The maximum percentage of speed to a waypoint.
Tolerance	The distance from a waypoint's location by which the vehicle has met its target. **Note:** Tolerance must be greater than or equal to 1 meter.

Table 9: Waypoint features and descriptions.

Default values for “Depth”, “Speed”, and “Tolerance” are set when a waypoint is added. Editing the default, or the waypoint's location, is done in the “Waypoints” tab.

The coordinates to set the waypoints can be manually entered in the Latitude and Longitude field by double-clicking that field and typing a set of coordinates. Once these key waypoint attributes are set, the mission is ready to be saved and loaded onto the vehicle.

7.3 Retrieving a Mission

A mission that has been saved can be retrieved in the “Mission” tab by selecting the **Load Mission From File** button. A file browser will open and the user can navigate to the saved mission file. The stored waypoints will appear on the screen. Click **Push To Vehicle** to load the mission to the vehicle's computer. The mission is now ready to go.

7.4 Executing a Mission

Once the mission has been given to the vehicle, it is activated and deactivated by the **Waypoint Following** button within the Autopilots Control Menu. If the waypoints are modified they must be pushed to the vehicle again before the vehicle will begin following the new instructions.

8 Alarm Manager

Alarms alert the operator when a signal crosses a user defined warning threshold. The Alarm Manager page is used to create and manage these alarms. Alarms can be placed on any signal, and a priority level can be set to reflect the severity of the situation. The alarm screen displays the current status of all alarms in the main panel and the history of all alarm notifications on the right.

Illustration 26: Alarm Manager is selected from the Navigation Bar.

The Alarm Manager page in the workspace provides a front end to the Alarm Manager. The Alarm Manager is the application that generates and manages alarms within the software system. If the Alarm Manager is not running, alarms can be neither configured nor managed. If the Alarm Manager application is not running, the static bar in the workspace will display a warning message where the alarm posting are usually listed. The application's status can also be monitored by opening the Alarm Manager page. If the application is running, the operator will be able to create and manage alarms. If not, the alarms on this page will be unclickable and gray.

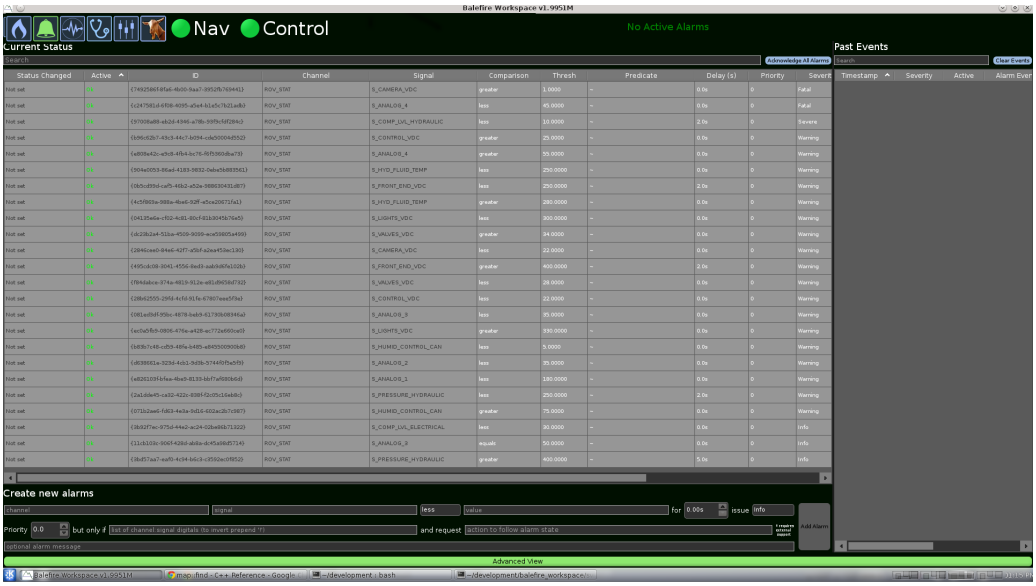


Illustration 27: The Alarm Manager screen.

8.1 Alarm Manager Buttons

Operators control the state of the Alarm Manager page with the buttons on the top. The Alarm Manager has four buttons across the top of the page that allow the operator to create and edit alarms, save alarms, load previously defined alarm groups, and enable the advanced view.

Button	*Description*
Enable Editing	Allows the operator to create and edit alarms.
Save Alarms	Saves the current set of alarms to a configuration file.
Restore Alarms	Allows the operator to restore a saved set of alarms from a configuration file.
Advanced View	Gives operator access to all editable alarm fields.

Table 10: Alarm Manager buttons and their functionality.

8.2 Adding an Alarm

To create an alarm, first determine what signal is to be monitored. Once the signal is specified, the operator can populate the channel, signal, comparator, trigger time, and alarm severity fields. After filling in these fields, simply click add alarm to place it into the alarm manager's queue. In the example below, the signal, "unix_time" on the channel "OPENINS_NAV_SOLUTION" is set to trigger a severe warning to the operator if the timestamp is greater than 1500 seconds for a trigger time of 0 seconds.

Alarm Field	*Description*
Channel	The channel on which the operator wishes to listen for an alarm. In this case, the desired channel is “OPENINS_NAV_SOLUTION”.
Signal	The specific signal the operator wishes to monitor. In this example, “unix_time” on the channel “OPENINS-NAV_SOLUTION” will be monitored.
Comparator	The three comparator options are “less”, “equal”, and “greater” and will trigger an alarm based off this setting. For this example, the comparator “greater” was chosen.
Trigger Time	The amount of time that the alarm conditions are allowed to persist before Alarm Manager triggers an alarm. For this example, the trigger time is set to 0 seconds.

Table 11: Alarm fields and descriptions.

Illustration 28: The alarm creation interface for the Alarm Manager.

8.3 Alarm Channels

The alarm channels are defined in the ventana_alarms.yml, located in the gss_configs/gss_alarm_manager/ directory. The complete field list is as follows:

Alarm Field Name	*Description*
ID object	An auto-generated unique ID.
Input	The command channel for the input.
Transform	The condition for the alarm can be “less than”, “equals”, or “greater than”.
Priority	The alarm priority is set with the toggle arrows in advanced view.
Severity	The severity of the alarm ranges from informational to fatal.
Message	An optional message to display with alarm.
Action	To publish an LCM command based on an alarm. publish=channel:signal.
Note	A field to hold any special notes for the alarm.

Table 12: Alarm channel fields and descriptions.

9 Data Plotter

Illustration 29: Data Plotter is selected from the Navigation Bar.

The Balefire Workspace provides powerful and comprehensive tools to monitor the health and status of the physical systems to which it interfaces as well as the software system itself. Two principle diagnostic tools provided by Balefire are the Data Plotter and the Process Viewer. The Data Plotter provides an oscilloscope-type plotter for viewing any signal currently in the system while the Process Viewer provides status on the processes currently managed by the Process Server and executed by the local Process Client.

9.1 Signal View

The left side of the workspace screen lists the active channels and signals eligible for plotting. Clicking the "+" icon to the left of the channel name will expand the channel to show all its associated signals. To minimize the view, click the "-" icon visible to the left of the channel name. The channels are keyword searchable and can be filtered by filling in the search bar above the list of data signals. To select a signal to be plotted, select the check box next to the signal name. Multiple signals can be plotted at once in a color coded fashion. When a signal is selected for plotting, the signal text is highlighted in the same color that it appears in the plotter utility.

The data plotter also provides several filters for sorting and displaying signals. Operators can filter by name simply by clicking on the "Channel/Signal" header button. Operators can also only display signals that are currently being plotted by clicking the "Display only plotted signals" button at the bottom of the signal list.

Understanding how the Data Plotter receives and manages signals is important to fully utilize the tool for diagnostics. The Data Plotter searches the network for published system data. As the data is found on the network, the Data Plotter creates a Channel/Signal entry and populates the list with the data. Once the data is in the list, it is available to plot. The Data Plotter updates the data in the list in real-time as data is published in the network. Using this understanding, operators can troubleshoot the system by seeing what data is available on the network from what publishers. If data is not listed, it is currently not being published. Clicking the "Clear" button next to the search field clears the list of data. Note though, the Data Plotter will not remove a signal from the list if it goes dormant so the list may provide signals that have gone dormant and are no longer available on the network. The "Clear" button will force the system to update and provide a fresh list of available data.

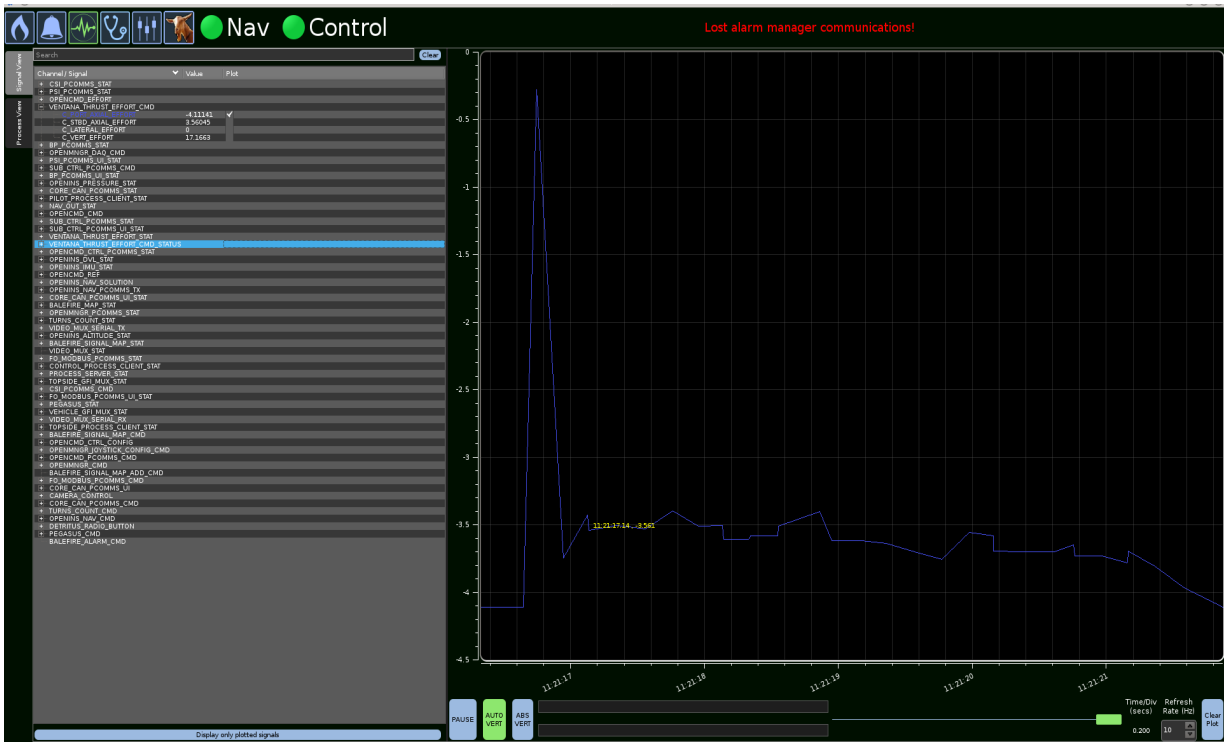


Illustration 30: The Balefire Data Plotter in "Signal View".

Function	Description
Clear	The Clear button will clear the active channels and signals. They will repopulate as Balefire receives their data. This does NOT clear the search field.
Display only Plotted Signals	The "Display only Plotted Signals" button will remove all signals that are not selected for plotting from the active channels and signals.
Pause	Pausing will stop the stream of real time data to the plotter. Real time data can again be observed by disengaging the pause button.
Auto Vertical	Select the y-axis maximum and minimum values from the data value default.
Absolute Vertical	Set user defined maximum and minimum y-axis values.
Use Local Time	Switch the plot reference from the data feed to local time, or the computer time.
Clear Plot	Clear the graphical plot.
Refresh Rate	Change the rate (Hertz) that the data is plotting with the toggle arrows.
Time/Division in seconds	Adjust the time rate division in seconds via the scroll bar at the bottom right of the plotter.
Zoom	Zoom in to the plot by clicking on the plot screen at the desired zoom point and dragging out. The graph will pause automatically. Press PAUSE again to continue plotting.

Table 13: Description of Data Plotter tools.

9.2 Process View

"Process View" shows all running applications that are managed through the process server. Through the "Publish Console" checkbox, the operator can choose to see the standard output of a given application. The "Process View" fields are described in detail below.

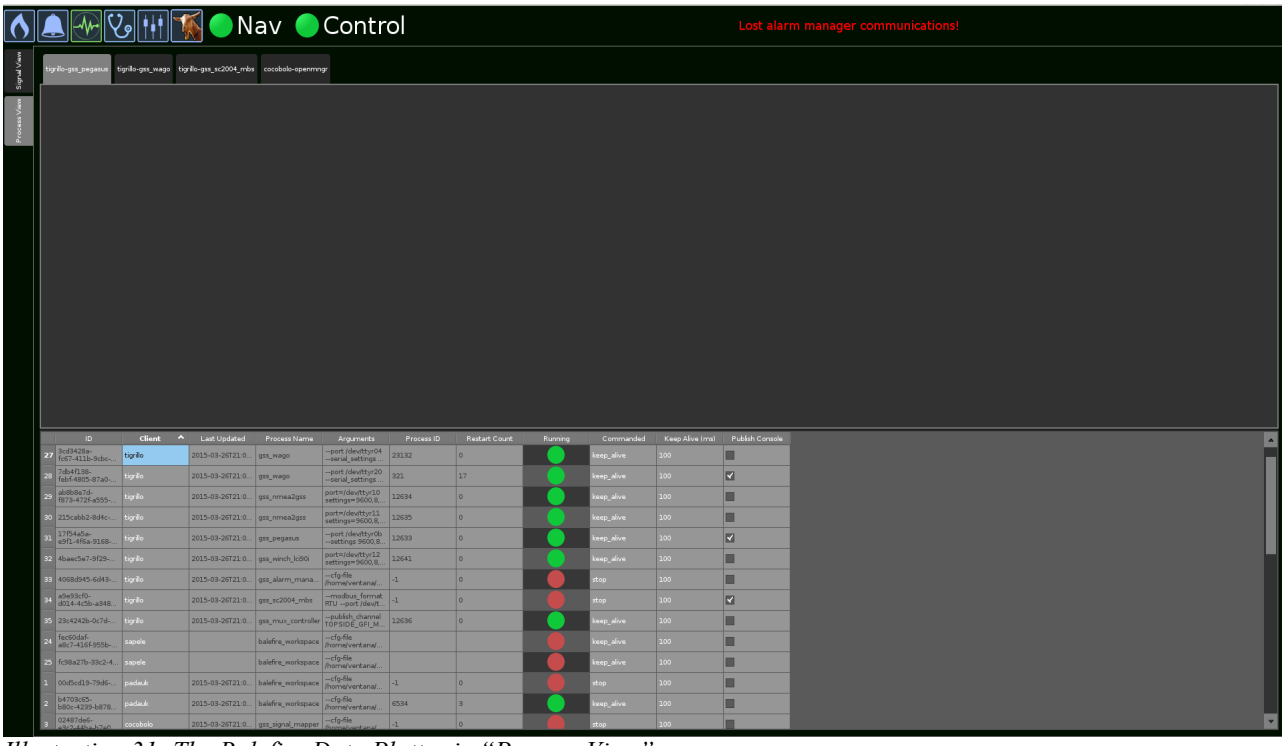


Illustration 31: The Balefire Data Plotter in "Process View".

Function	*Description*
ID	The ID used by the process server and Process Client to manage processes.
Client	The name of the Process Client to manage the process.
Last Updated	The time of the Process Client's last update.
Process Name	The specific application's name.
Arguments	The arguments used to call a given application.
Process ID	The OS identifier for a given process.
Restart Count	How many time a process has restarted.
Running	Status LED indicates if the application is running (green) or not running (red).
Commanded	Start, Keep Alive, Stop. "Start" is a one time start command given to the process. "Stop" will command a running process to stop. "Keep Alive" will keep a program alive by restarting it in the event that it stops.
Keep Alive (ms)	The amount of time a process can be dead before it will restart.
Publish Console	Publishes the standard output from the application to a frame above the process view window. This action can be duplicated by checking the standard output of that application from a console.

Table 14: The descriptions of "Process View" fields.

10 Signal Mapper

The Signal Mapper is a fundamental and powerful component of the Balefire Workspace and the overall *Ventana* software architecture. The signal mapping page in the Balefire Workspace provides the operator a graphical front end for configuring the Signal Mapper. An understanding of the *Ventana* Control System architecture is required to make effective use of this facility.

Illustration 32: Signal Mapper selected from the Navigation Bar.

The Signal Mapper monitors all channels and signals for operator-defined signal transformations. The page in the workspace provides the operator configurable assignment of physical controls to logical functions in addition to scaling of input/output values to or from more representative units. A range of transformations from simple inversion to scalable two-axis controls with configurable dead-bands are available. Any transformation may be chained with others to provide more complex behavior such as two-axis control with variable bias, stick locking and optional secondary or tertiary functions.

This section describes the use of the Signal Mapper by detailing the processes to scale an analog and a digital signal by example. The input channels, input signals, output channels, and output channels are specific to these examples but the process can be applied to any analog or digital signals on any set of user defined channels and signals.

10.1 Signal Mapper State Control Buttons

The Signal Mapper editor page has four buttons across the top of the page that control the state of the Signal Mapper application. These controls allow the operator to create and edit signal maps, save signal maps, and restore them. The buttons functions are outlined in the following table.

Button	*Description*
Enable/Disable Editing	Allows the operator to create and edit signal mappings. Button is disabled (blue) by default.
Save Signal Maps	Saves the current set of signal mappings to a configuration file.
Restore Signal maps	Allows the operator to restore a saved set of signal mappings from a configuration file.
Advanced View	Gives operator access to all editable signal mapping fields.

Table 15: Signal Mapper buttons and their functionality.

10.2 Signal Output Transform

The buttons located above the output transform are Set Min Deflection, Center Deadband, and Set Max Deflection. These buttons conveniently set the upper and lower limits of the input signal. The deadband can also be set using the Center Deadband button. Their functions are illustrated in the following sections.

- **Set Max Deflection:** To set maximum deflection, push the input to the full positive position. Hold the input in this position and then press the Set Max Deflection button. This aligns the software scaling for maximum deflection at the input's full positive position.
- **Set Min Deflection:** To set minimum deflection, push the input to the full negative position. Hold the input in this position and then press the Set Min Deflection button. This aligns the software scaling for minimum deflection at the input's full negative position.
- **Center Deadband:** To center the deadband, allow the input to return to it's NULL position. With the input null, press Center Deadband. The software will center the deadband around this value. This aligns the software deadband and the rest position of the input.

To the right of the graphic are the controls for the output scaling. The Max Output and Min Output are set using text boxes. The output null is adjusted using the slider, and can be centered by pressing the **Center Output Null** button. Set standard range is an auto-set value that resets the configuration. The most typical signal scaling is set to have a maximum output of 100.0, a minimum output of -100.0 and a centered null output. Of course, these values can be configured to your preferences.

The sliders beneath transform allow the Input Min, Deadband Min, Deadband Max and Input Max values to be easily altered. The Input Min slider adjusts the position at which the joystick will output the minimum value (Minimum value is set in the Min Output text box) The deadband min and max sliders allow the size of the deadband to be widened or narrowed.

The dual linear transform is the typical transform used for the joystick or bipolar input applications. Operators should ensure an adequately sized deadband (input position where you require the output to be zero). The dual linear transform provides a deadband region, a linearly increasing region from the null point to the maximum output, and a linearly decreasing region from the null point to the minimum output. The sizes of these regions can be adjusted using sliders.

Operators can visually adjust the deadband by moving the deadband sliders left and right. This will physically manifest itself in a much larger region of joystick reflection that will generate a null output. Likewise, operators can adjust the slope of the minimum and maximum regions by adjust the output sliders.

10.3 Configuration

The Signal Mapper maintains its definitions of signal maps in a configuration file. While operators may edit this file directly (when the Signal Mapper application is not running) it is preferred they use the provided facilities in the workspace to edit the Signal Mapper. A sample signal map definition is listed below.

```
- map_id: "{26b52428-3c6c-44fb-8028-6b8090c9699a}"
  input: VENTANA_THRUST_EFFORT_CMD:C_VERT_EFFORT
  output: SUB_CTRL_PCOMMS_CMD:c_vert_op
  profile: THRUSTER_CMDS
  description: transform gss thruster effort percent to WAGO analog command
  transform: AND(!PSI_PCOMMS_STAT:s_bellypack_enable)|LINEAR(163.8,0)
```

The Signal Mapper editor provides access to all configurable parameters of a signal map. Creating and defining a signal map requires the operator specify the input and output at a minimum. The Signal Mapper will populate the other parameters with default or NULL values as appropriate. Some of the configurable parameters for the Signal Mapper are listed in the following table.

Signal Mapper Field	*Description*
id_object	A universally unique identifier for the signal map. It may be omitted when hand-editing, whereupon Signal Mapper will assign one the next time the configuration files is read.
Input	The input channel and signal. Each time the signal is published over the channel, the signal map will be evaluated.
Output	The output channel and signal. If the transform does not drop a message, each time the input is published Signal Mapper will transform and publish over output immediately.
Profile	Signal maps can be assigned profiles. This provides both organization as well as a facility to enable and disable the signal maps in a profile as a group.
Description	A human readable description of the intent of the signal map. It is for operator or system designer use only.
Transform	Provides a configurable means to transform an input signal before publishing the result as output. Transforms can be chained, as shown in the example, using the '|' symbol, in which case they will be evaluated left to right. Some transforms drop messages while others modify the input value. The example below demonstrates one of each; the 'AND' filter drops a message unless each of the comma separated values digital signals are true (they support logical inversion by prepending '!') while the 'LINEAR' transform performs a linear transformation where the first argument is 'm' and the second is 'b' in the equation y = mx+b.

Table 16: Signal Mapper editor fields.

10.4 Transforms

The signal mapper, alarm manager and others share a library of transforms. These transforms, which may be chained with a '|' (the UNIX pipe symbol), together perform configurable transformations or filtering of input signals. Some transforms take additional parameters. Parameters should be specified as a comma-separated list inside of parentheses immediately following the transform name, e.g. GAIN(5). Many filters feature channel:signal pairs as parameters. A channel:signal pair is a string consisting of an LCM channel followed by a ':' followed by a signal that is published on that channel. Signals may be 'analogs' whose values are double precision floating point numbers, 'digitals', whose values are booleans, or 'strings' whose values are strings. Many transforms drop their input messages if certain conditions are not met, or if they receive an input of the incorrect type.

Section 14.1 of the Appendix holds a table describing the transforms available at the time this document was written, their parameters, and a brief description of their purpose and operation.

10.5 Map and Scale an Analog Signal

In this example, the analog signals from a joybox will be scaled. To fully scale all the analog signals this process must be repeated for each signal. The process is generally the same for each signal with some key differences. The input channel, “ADR2000_SAMPLES” will remain the same while the input signal must be changed each time. Similarly, the output channel “OPENMNGR_DAQ _CMD” will contain the individual output signals for the joybox.

10.5.1 Locate the Signal to be Scaled

Select the desired signal to be scaled. The illustration for the Data Plotter page shows ADR2000_SAMPLES (**Note:** This input signal is specific to this particular application of Signal Mapper). At full forward deflection the

output signal “U_DAQ_JOY_X" measures an analog read of 4095. Therefore, U_DAQ_JOY_X"is the correct output signal to be mapped and scaled for this example.

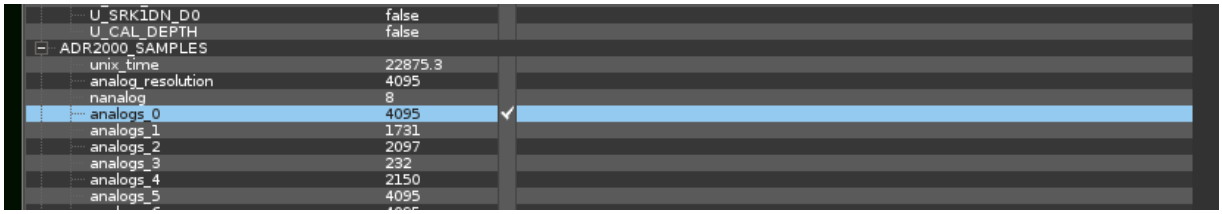


Illustration 33: Locate the signal by moving the joystick and locating the signal in the plot screen.

10.5.2 Generate Signal Mapping

The signal mapping and scaling is generated on the Signal Mapper page. Click **Enable Editing** to generate the mapping of a given signal. Under “Create Signap Mapping”, input necessary mapping information into the “Input channel”, “Input signal”, “Output channel”, and “Output signal” fields. In this instance, the information to generate the mapping is as follows:

Field	*Description*
Input Channel	ADR2000_SAMPLES (All signals, analog and digital, from the from the joybox are located here.)
Input Signal	analogs_0 (This signal controls +/- forward thrust per the example.)
Output Channel	OPENMNGR_DAQ_CMD (This is the channel where the scaled and mapped value will be stored.)
Output Signal	U_DAQ_JOY_X (This is the signal that is broadcast.)

Table 17: Signal mapping example parameters.

10.5.3 Scaling an Analog Signal

Once the signal mapping has been generated, the signal can be scaled. To enable scaling of a signal, click **Advanced View**. This will bring up new fields: “Transform”, “Input min”, “Deadband low”, “Deadband high”, “Input max”, “Output min”, “Output null”, and “Output max”. The advanced fields are used to automatically populate the DUAL_LINEAR transform. For example, DUAL_LINEAR(64,1097.459975,3100.540025,4095,-100,0,100) corresponds to an input min = (64, deadband low =1097.459975, deadband high = 3100.540025, input max = 4095, output min = -100, output null = 0, output max = 100.

This will set the input range of a 64 - 4095 analog output from -100 to 100 with a deadband from 1097-3100 respectively and an output null of 0. These values are adjustable as necessary to appropriately scale a given device's output. With these user inputs in place, the signal is now fully mapped and scaled.

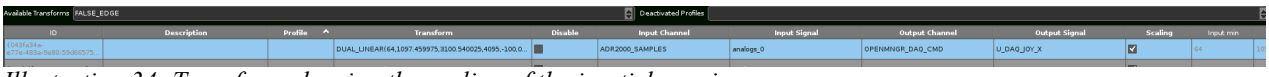


Illustration 34: Transform showing the scaling of the joystick x-axis.

10.5.4 Testing Scaled Signals

To test a given signal's scaling, open “Signal View” from within the Data Plotter. Select the output channel and signal (In this case the output channel: OPENMNGR_DAQ_CMD, output signal: U_DAQ_JOY_X) to be tested. Click the "Plot" checkbox to the right of the output signal name. A time trace of the signal will be plotted in the

Balefire Data View grid. The **Display only plotted signals** button can be pressed to minimize all output channels/signals that are not plotted in the Balefire Data View grid.

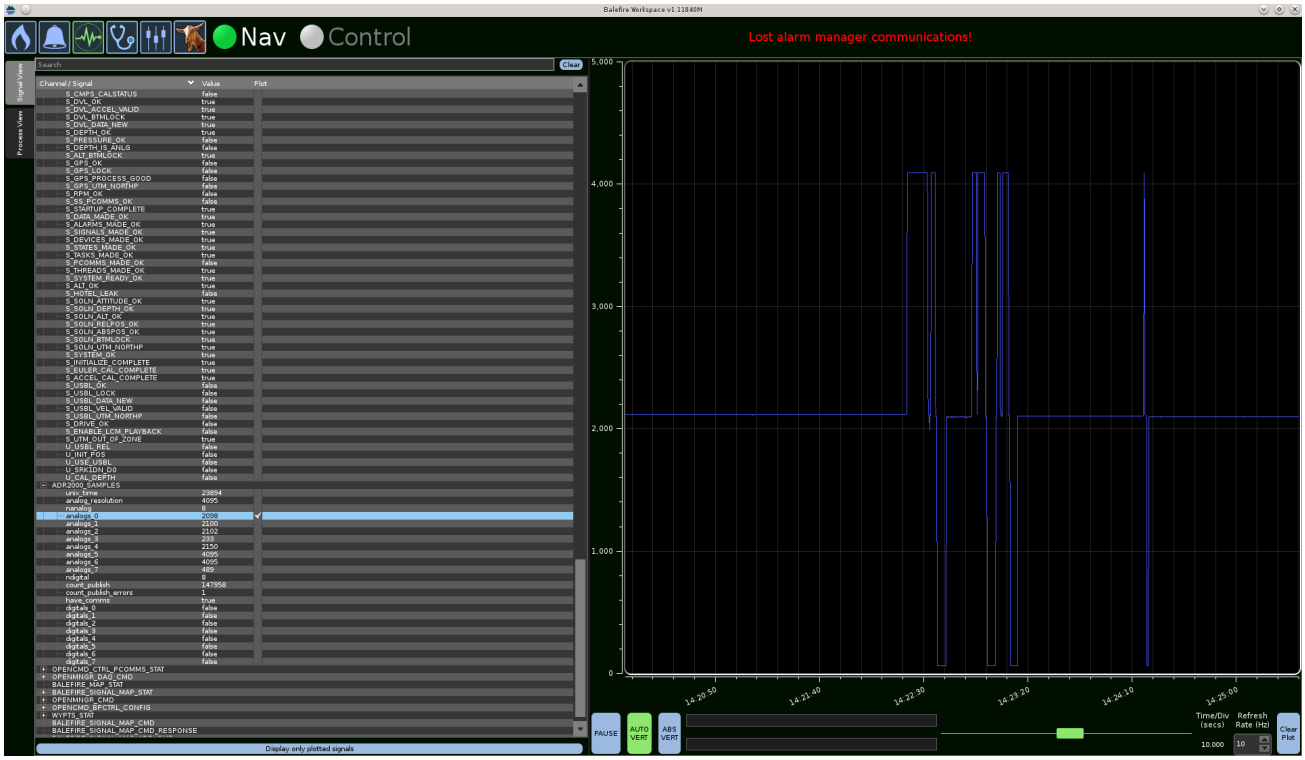


Illustration 35: Test scaled signals in the plot window.

10.5.5 Save Analog Signal Mapping

Back on the Signal Mapper page, click **Save Signal Maps** to save the signal mappings.

10.6 Profiles

Profiles are groups of signal maps that may be activated or deactivated together. Profiles are activated by publishing a digital pcomms signal on LCM channel "BALEFIRE_SIGNAL_MAP_CMD". The signal name should be "ACTIVATE_PROFILE(<profile>)" where <profile> is the name of the profile whose status should be changed. If the value of the digital signal is true the profile in question will be activated, if the value is false, the profile will be deactivated. Multiple profiles can be activated or deactivated with a single command by providing a comma separated list of the profiles, e.g "ACTIVATE_PROFILE(profile1,profile2,profile3)"

For example, the hydraulic tools are managed by creating eight profiles, one for each of the seven hydraulic tools and one for "None". To activate a profile a button is created in the balefire workspace configuration that publishes a true signal with name ACTIVATE_PROFILE(HYD_TOOL1) on channel BALEFIRE_SIGNAL_MAP_CMD when it is selected.

The following example is for pc_hydraulic_crtl_tools_btn_1:

pc_hydraulic_crtl_tools_btn_1:

text: "Port Drawer"

output_channel: BALEFIRE_SIGNAL_MAP_CMD

output_signal: ACTIVATE_PROFILE(HYD_TOOL1)

radio_channel: HYD_TOOLS_RADIO_BUTTON

radio_signal: hyd_tool1

Next, a signal map is created so that when this profile is activated, the other seven get deactivated. Because this mapping has the INVERT transform it will publish a digital with the opposite value of that published by the button, so when the button is selected and publishes true, this mapping will generate a digital signal with a value of false which will deactivate the listed profiles.

map_id:

input: BALEFIRE_SIGNAL_MAP_CMD:ACTIVATE_PROFILE(HYD_TOOL1)

output:
BALEFIRE_SIGNAL_MAP_CMD:ACTIVATE_PROFILE(HYD_TOOL2,HYD_TOOL3,HYD_TOOL4
, \

HYD_TOOL5,HYD_TOOL6,HYD_TOOL7,HYD_TOOL8))

profile: HYDRAULIC_MAPPING

description: Hydraulic mapping to deactivate unselected profiles

transform: INVERT

11 Diagnostics

Diagnostics is a split-screen that illustrates the current health and status of devices for both the topside and the vehicle. The icon is a stethoscope, which is located in the Navigation Bar.

Illustration 36: Diagnostics is selected from the Navigation Bar.

11.1 Topside Devices

The “Topside Devices” tab gives location data via GPS, USBL, and the ship compass. The vessel's location is given via GPS and the ROV's position is given via USBL.

11.1.1 Topside GPS

The topside GPS pane contains useful data concerning the ship's GPS fix. The operator can see if satellite lock has been established, the timestamp, last fix time, fix type, latitude, longitude, elevation, amount of satellites in use, and the vessel's speed over ground.

Topside Field Name	*Description*
Satellite Lock	Shows whether a GPS fix has been established (green LED) or not (gray LED). This requires communication with at least 3 GPS Satellites.
Timestamp	The Unix time stamp accompanying the most recent GPS fix.
Last Fix Time	The last GPS fix given in Universal Coordinated Time (UTC).
Fix Type	2D or 3D fix. 2D fix can be determined if communicating with 3 satellites. 3D fix is more accurate and requires communication with at least 4 satellites.
Latitude	The latitude of the ship given in degrees.
Longitude	The longitude of the ship given in degrees.
Elevation	The elevation of the ship with respect to sea level. Even when operating at sea level, the elevation may not display as 0 due to GPS limitations.
Satellites in Use	How many satellites are in use to determine the GPS fix.

Table 18: Topside field names and descriptions.

11.1.2 Topside Compass

The “NMEA Ship Heading” screen reads current compass data. Four different values are displayed:

Heading Field Name	*Description*
Timestamp	The Unix time stamp of the most current compass reading.
Heading	The current heading in degrees.
Pitch	The vehicle pitch, or rotation about the y-axis, in degrees.
Roll	The vehicle roll, or motion about the x-axis, in degrees.

Table 19: Compass field names and descriptions.

11.1.3 Ultra-Short Baseline

Ultra-Short Baseline (USBL) is an acoustic positioning method to locate the vehicle subsea where GPS in unavailable. The USBL pings on the map are shown graphically as small circles. The positioning shows data for both the topside and the USBL.

USBL Field Name	*Description*
Timestamp	The Unix time stamp for the most current location fix.
Latitude	The location in latitude degrees.
Longitude	The location in longitude degrees.
Range	Distance of the vehicle to the ship.

Table 20: USBL field names and descriptions.

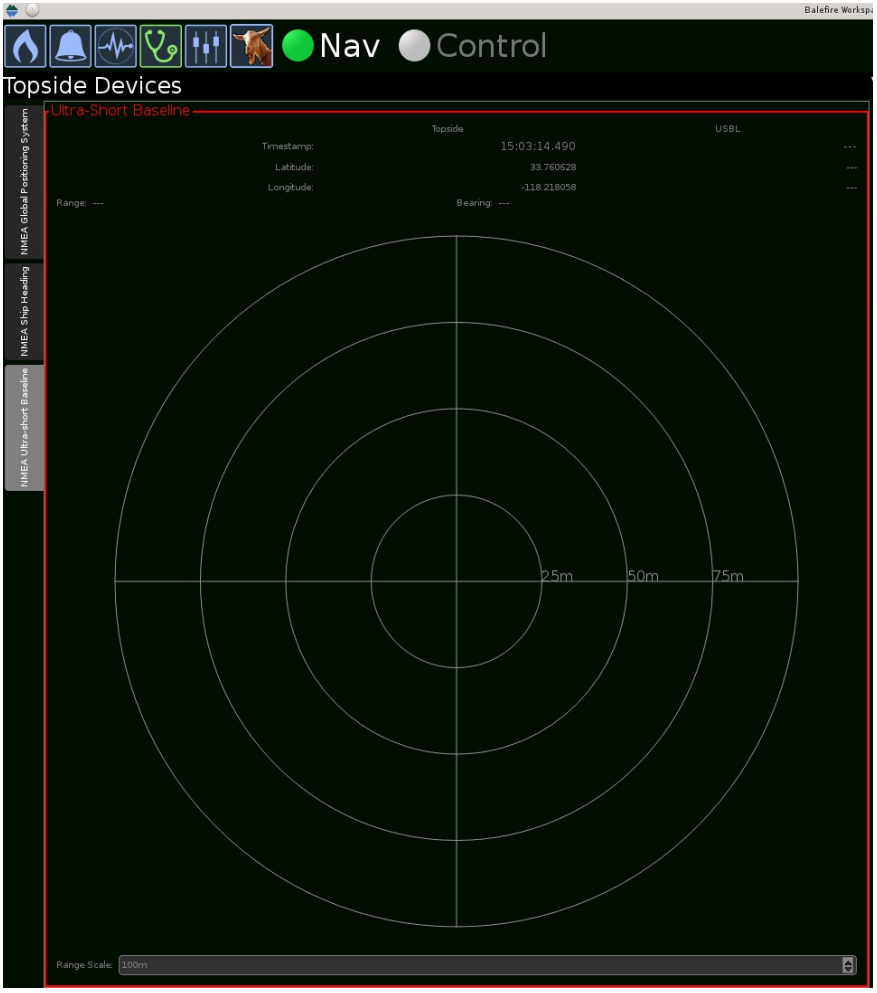


Illustration 37: USBL screen without data.

11.2 Vehicle Devices

"Vehicle Devices" shows aiding sensor data for whatever devices your vehicle has. There is a tab for each device and each tab displays sensor data as it's being collected.

11.2.1 Inertial Measurement Unit

The IMU sensor screen (found within the "Compass" tab) shows the attitude of the vehicle. Several attitude values are displayed:

IMU Attitude Field Name	*Description*
Timestamp	The Unix time stamp of the most current IMU reading.
Heading	The heading relative to the topside of the vehicle.
Pitch	The pitch of the vehicle measured in degrees.

IMU Attitude Field Name	*Description*
Roll	The roll of the vehicle measured in degrees.
Heading Rate	The velocity rate of the current heading for the vehicle.
Pitch Rate	The pitch rate of the vehicle.
Forward Acceleration:	The acceleration value for the ROV in the forward direction.
Starboard Acceleration	The acceleration rate for the starboard side of the ROV.

Table 21: Attitude field names and descriptions.

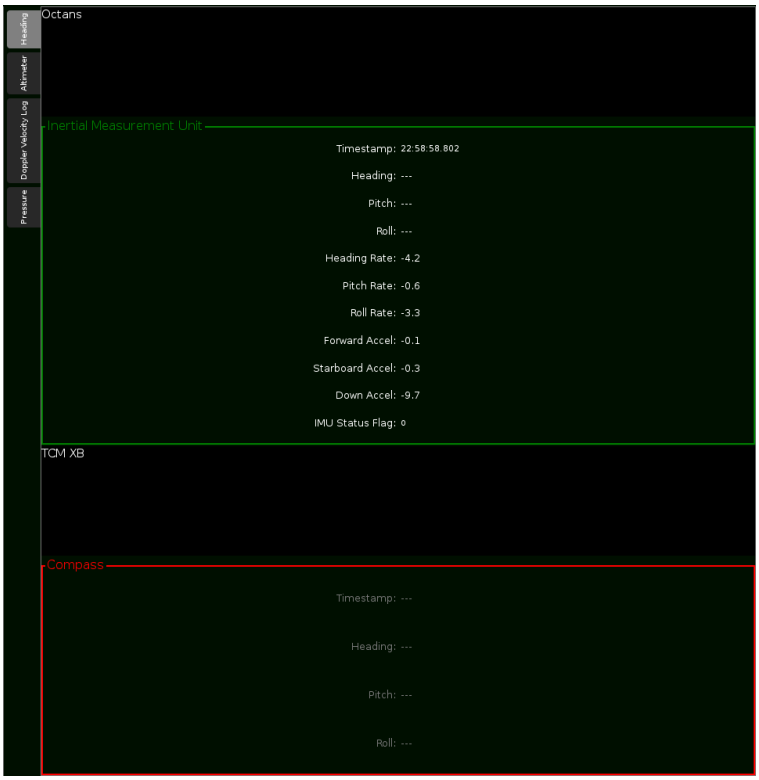


Illustration 38: "Vehicle Devices" screen showing IMU information in the selected "Heading" tab.

11.2.2 Pressure

"Pressure" shows three different readings:

Pressure Sensor	*Description*
Timestamp	The Unix time stamp of the most current pressure reading.
Count	The count of pressure readings received.
Pressure	The current pressure reading.

Table 22: Pressure sensor field names and descriptions.

Illustration 39: "Pressure" is selected with no data displayed.

11.2.3 Heading

"Heading" reads current compass data. Four different values are displayed:

Heading Field Name	*Description*
Timestamp	The Unix time stamp of the most current compass reading.
Heading	The current heading in degrees.
Pitch	The vehicle pitch, or rotation, about the y-axis in degrees.
Roll	The vehicle roll, or motion about the x-axis in degrees.

Table 23: Heading field names and descriptions.

Illustration 40: "Heading" screen shows compass data.

11.2.4 Altimeter

"Altimeter" displays the following data:

Altimeter Field Name	*Description*
Timestamp	The Unix time stamp of the most current altimeter reading.
Bottom Lock	The altimeter has a reference lock on the sea floor.
Altitude	The measurement (in meters) of the vehicle's altitude with reference to sea level.

Table 24: Altimeter field names and descriptions.

Illustration 41: Altimeter screen snapshot

11.2.5 Doppler Velocity Log

Incoming data from the DVL:

DVL Field Name	*Description*
Bottom Lock	The altimeter has a reference lock on the sea floor.
Timestamp	The Unix time stamp of the most current DVL reading.
Forward Velocity	The rate of the vehicle's forward motion.
Starboard Velocity	The rate of motion of the starboard side of the vehicle.
Down Velocity	The rate at which the vehicle is descending.
Mean Altitude	The average recorded altitude. Altitude is measured in distance above sea level. This default measurement is in meters.

Table 25: DVL field names and descriptions.

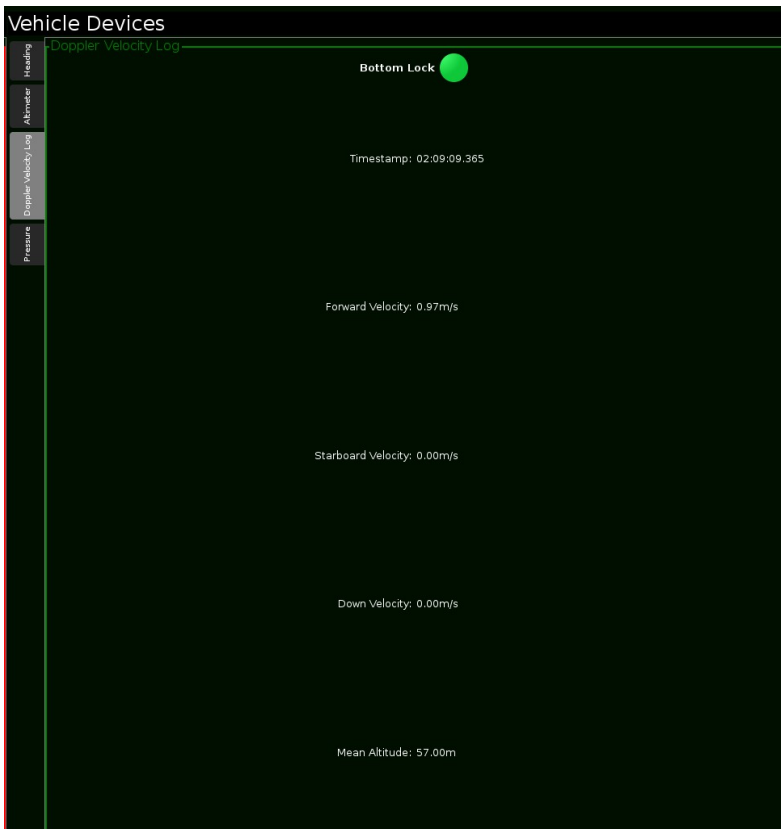


Illustration 42: Incoming DVL information is displayed here.

12 The “Goat” Screen

This pilot-customizable screen has tabbed pages that offer detailed views of “Pilot Controls”, “Pilot Info”, “Science View”, “Video Switch”, and “Insite Control”. The current views of the Goat screen reflect MBARI preferences as of installation, May 2015.

Illustration 43: The "Goat" Screen is selected on the Navigation Bar.

The balefire_workspace _cfg.yml contains all of the data for customizing the Goat screen. In the config file there are sections noted by a comment starting with a # sign.

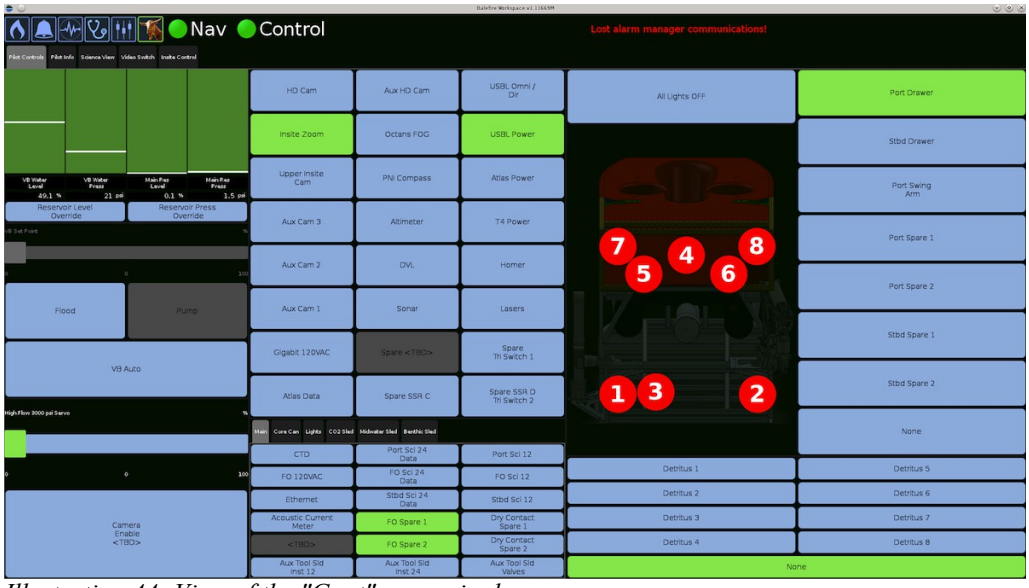


Illustration 44: View of the "Goat" screen is shown.

12.1 Pilot Controls

“Pilot Controls” provides button and slider control for the auxiliary components, lights and cameras.

Most of the buttons are simple on/off buttons, however the lights, hydraulic, detritus and variable buoyancy system all have more complicated behavior described below.

- **Lights:** Pushing the lights button turns them on and off. However, it is important to note that they are interlocked with the light isolation buttons on the lights tab. Before a light can be isolated it needs to be turned off. Once a light is isolated it can't be switched on.

The Hydraulic Tools and Detritus have manual and automated modes of operation. As a safety measure, they start in a “none” position. This ensures that they are not accidentally triggered at start-up.

- **Hydraulic Tools:** To engage hydraulic tools a pilot must manually turn on the tool with a rocker switch, or joystick trigger. If you want a hydraulic tool to be active, you select it. Then you have control with the rocker. There is no “auto” mode for the hydraulic tools. The configuration is set so that one detritus sampler or one hydraulic function can be selected at a time. For each button group there is also a "None" option. When "None" is selected the trigger and rocker buttons actuate tool sled devices.

- **Detritus:** To engage the detritus a pilot must manually turn on the tool with a trigger. If you want the detritus to be active, you select it. Then you have control with the trigger. There is no “auto mode” for the detritus.

The Variable Buoyancy (VB) system can be in automatic or manual mode. In manual mode, an operator can press the

flood button or the pump button and the VB chamber is either flooded or pumped accordingly. When they select automatic mode there is a simple controller that floods/pumps until the chamber is filled to the set point that is specified by a slider.

12.2 Insite Control

"Insite Pegasus Camera Control" has a series of buttons to manipulate the selected camera. The current selected camera displays in the top center, and a status LED shows whether or not there are camera communications. The LED will be green when communications are being received; it will be red when comms are lost.

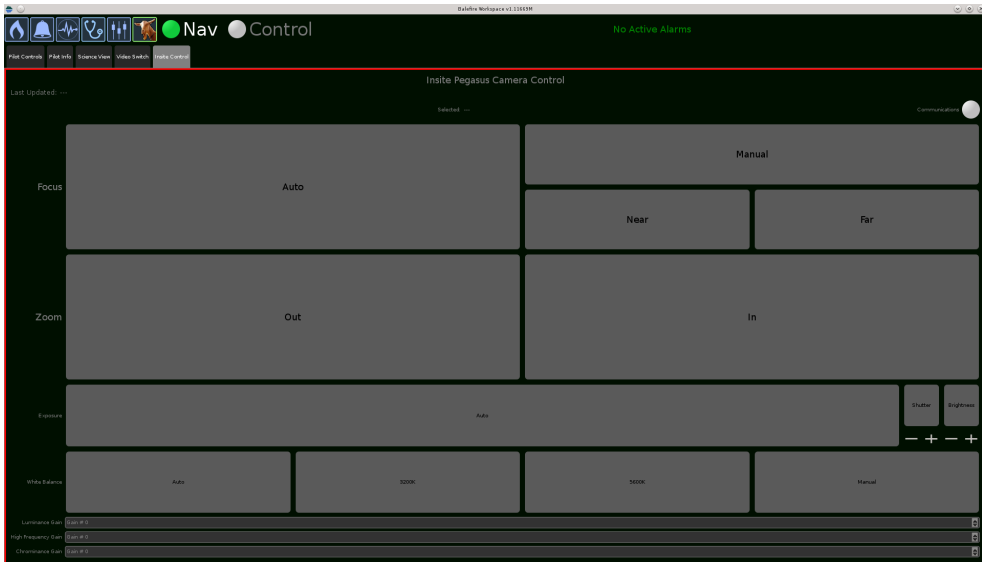


Illustration 45: The Insite control with no active cameras.

Some general notes:

- The buttons are green when selected and blue when not active.
- To focus the camera in auto mode simply push **Auto**. You can leave it in manual if you prefer.
- Zoom can be incrementally pulled in or out.
- The exposure can be tailored with a button click from **Auto** to increments.
- White balance is also correctable with an **Auto** setting, or selected values. "Luminance", "High Frequency" and "Chrominance" gains can be altered with selectable drop-down values.

12.3 Pilot Info

"Pilot Info" details the running state of devices and sensors for the *Ventana*. The green LEDs show interfaces with current communication. The tabbed menu below the controls provides data information for the topside and the environment such as status and temperature.

Illustration 46: A view of the "Pilot Info" screen.

12.4 Science View

"Science View" has a split screen showing heading, depth and altitude, as well as a tabbed menu with axillary controls.

13 Appendix

13.1 Transforms

Below is a table describing the transforms available at the time this document was written, their parameters, and a brief description of their purpose and operation.

Filter Name	*Parameters*	*Description*
AND	Comma-separated list of digital channel:signal pairs (optionally prepended with '!' for logical inversion).	Outputs a digital with the value of the logical AND of the input signal and all of the parameter signals.
OR	Comma-separated list of digital channel:signal pairs (optionally prepended with '!' for logical inversion).	Outputs a digital with the value of the logical OR of the input signal and all of the parameter signals.
ANY	Comma-separated list of digital channel:signal pairs (optionally prepended with '!' for logical inversion).	Drops the input signal unless one of the parameter signals is true.
ALL	Comma-separated list of digital channel:signal pairs (optionally prepended with '!' for logical inversion).	Drops the input signal unless all of the parameter signals are true.

Filter Name	*Parameters*	*Description*
BIAS	Scalar value or analog channel:signal pair.	Adds the scalar or the value of the analog channel:signal to the input value. If the input is digital it is converted to analog with the argument bias.
BOUNDS	Two integer values, min and max.	Provides value bounds defined by minimum and maximum limits.
CONCAT	String or channel:signal containing a string.	Append fixed string or value of string channel:signal to the input string. Drops non-string inputs.
DELAY_EQUALS	First parameter is the comparison value, second is the delay period in seconds.	The comparison may be a fixed scalar, 'true' or 'false', a string or a channel:signal. The message is dropped unless the input value matches the type and value of the comparison for greater than delay seconds.
DELAY_GREATER	First parameter is the comparison value, second is the delay period in seconds.	The comparison may be a fixed scalar or an analog channel:signal. The message is dropped unless the input value is analog and is greater than the comparison for greater than delay seconds. Drops non-analog inputs.
DELAY_LESSER	First parameter is the comparison value, second is the delay period in seconds.	The comparison may be a fixed scalar or an analog channel:signal. The message is dropped unless the input value is analog and is less than the comparison for greater than delay seconds. Drops non-analog inputs.
DUAL_LINEAR	The parameters, in order, are - input minimum - deadband low - deadband high - input maximum - output min - output null - output max	Provides complete functionality for a bi-directional proportional control with dead-bands. Often used to transform an analog input to a percent deflection output. Input values are clamped to [input_minimum,input_maximum]. If they fall within [deadband low, deadband high] then output null is returned, otherwise they are transformed from either [input minimum, deadband low) to [output min, output null) or (deadband high, input maximum] to (output null, output max]. Drops non-analog inputs.
EQUALS	First parameter is the comparison value, second is the delay period in seconds.	Same behavior as DELAY_EQUALS without a delay.
FALSE	None.	Converts any input to a digital with value false.
TRUE	None.	Converts any input to a digital with value true.
GAIN	Scalar value or channel:signal.	Multiply a scalar or the value of an external analog channel:signal by the input value.
GREATER	First parameter is the comparison value, seconds is the delay period in seconds.	Same behavior as DELAY_GREATER without a delay.
IGNORE	None.	Drops all inputs. Used to temporarily disable a transform.

Filter Name	*Parameters*	*Description*
INCREMENT	Scalar representing the maximum counter value.	Used to encapsulate a wrapping counter to provide a monotonically increasing output number. Drops non-analog inputs.
INVERT	None.	Inverts digital inputs or the sign of analog inputs. Drops all others.
LESSER	First parameter is the comparison value, second is the delay period in seconds.	Same behavior as DELAY_LESSER without a delay.
LINEAR	First parameter is gain (m), the second parameter is bias (b). The parameters 'm' and 'b' may presented as scalar values or analog channel:signal strings.	Performs a linear transformation of analog input values where output = input * m + b.
LOCK	A single digital channel:signal.	When the parameter signal becomes true it will 'lock' the output value to last input value up to that point. This transform is often used to provide joystick locking functionality.
MIX	Comma separated list of analog channel:signals.	Output the largest absolute value of the analog input signal and parameter signals. Drops non-analog inputs. This transform is often used to combine multiple inputs to single control output; E.g. pilot and copilot pan & tilt joysticks.
ON_CHANGE	None.	Passes the input once each time a signal value changes
FALSE_EDGE	None.	Passes the input once each time a digital input changes from true to false. Drops non-digital inputs.
TRUE_EDGE	None.	Passes the input once each time a digital input changes from true to false. Drops non-digital inputs.
POLAR_DIFF	A single scalar or analog channel:signal parameter whose value is in degrees.	Outputs the signed shortest distance in degrees from the input value, also understood to be in degrees, to the value of the parameter. As the bearing values are normalized the distance will never be greater than 180. Drops non-analog values.
PREPEND	A string or a string channel:signal pair.	Prepend the fixed string or value of string channel:signal to the input string. Drops non-string inputs.
RATE_OF_CHANGE	A value representing the number of seconds to store values with which to calculate a rate of change or a signal from which to read a value.	Performs a calculation for rate of change and returns that value.
TOGGLE	A single digital channel:signal pair.	Outputs the opposite of the last value of its parameter channel:signal. Drops non-digital inputs.

Filter Name	*Parameters*	*Description*
TRIG_ACOS	None.	Outputs the arc-cosine of the input. Drops non-analog values.
TRIG_ASIN	None.	Outputs the arc-sine of the input. Drops non-analog values.
TRIG_COS	None.	Outputs the cosine of the input. Drops non-analog values.
TRIG_SIN	None.	Outputs the sine of the input. Drops non-analog values.
TRIG_TAN	None.	Outputs the tangent of the input. Drops non-analog values.
WINDOWED_CV	The size of the window as a scalar or an analog channel:signal.	Outputs the coefficient of variation of the input over the samples in the window. Drops non-analog values.
WINDOWED_MAX	The size of the window as a scalar or an analog channel:signal.	Outputs the maximum input value over the samples in the window. Drops non-analog values.
WINDOWED_MEAN	The size of the window as a scalar or an analog channel:signal.	Outputs the mean of the input values over the samples in the window. Drops non-analog values.
WINDOWED_MIN	The size of the window as a scalar or an analog channel:signal.	Outputs the minimum input value over the samples in the window. Drops non-analog values.
WINDOWED_STD	The size of the window as a scalar or an analog channel:signal	Outputs the standard deviation of the input values over the samples in the window. Drops non-analog values.
WINDOWED_VAR	The size of the window as a scalar or an analog channel:signal	Outputs the variance of the input values over the samples in the window. Drops non-analog values.

Table 26: Transforms, their parameters, and brief descriptions.

The next table shows examples of usage for the same filters:

Filter Name	*Usage*
ALL	ALL(CHAN1:sig1,CHAN2:sig2,...)
AND	AND(CHAN1:sig1,CHAN2:sig2)
ANY	ANY(CHAN1:sig1,CHAN2:sig2)
BIAS	BIAS(bias) or BIAS(CHAN:sig)
BOUNDS	BOUNDS(int min, int max)
CONCAT	CONCAT(CHAN:sig) or CONCAT(value)
DELAY_EQUALS	DELAY_EQUALS(compare_val,delay)
DELAY_GREATER	DELAY_GREATER(compare_val,delay)
DELAY_LESSER	DELAY_LESSER(compare_val,delay)
DUAL_LINEAR	DUAL_LINEAR(input_min,deadband_low,deadband_high,input_max,output_min,out

Filter Name	*Usage*
	put_null,output_max)
EQUALS	EQUALS(compare_val,delay)
FALSE	FALSE() no parameters
GAIN	GAIN(gain) or GAIN(CHAN:sig)
GREATER	GREATER(compare_val,delay)
IGNORE	IGNORE() no parameters
INCREMENT	INCREMENT(max_value)
INVERT	INVERT() no parameters
LESSER	LESSER(compare_val,delay)
LINEAR	LINEAR(gain, bias) or LINEAR(CHAN:sig1, CHAN:sig2)
LOCK	LOCK(CHAN:sig)
MIX	MIX(CHAN:SIG1,CHAN:sig2,...)
FALSE_EDGE	FALSE_EDGE() no parameters
ON_CHANGE	ON_CHANGE() no parameters
OR	OR(CHAN1:sig1,CHAN2:sig2)
POLAR_DIFF	POLAR_DIFF(val) or POLAR_DIFF(val)
PREPEND	PREPEND(CHAN:sig)
RATE_OF_CHANGE	RATE_OF_CHANGE(CHAN:sig) or RATE_OF_CHANGE(val)
TOGGLE	TOGGLE(CHAN:sig)
TRIG_ACOS	TRIG_ACOS() no parameters
TRIG_ASIN	TRIG_ASIN() no parameters
TRIG_COS	TRIG_COS() no parameters
TRIG_SIN	TRIG_SIN() no parameters
TRIG_TAN	TRIG_TAN() no parameters
TRUE	TRUE() no parameters
TRUE_EDGE	TRUE_EDGE() no parameters
WINDOWED_CV	WINDOWED_CV(CHAN:sig) or WINDOWED_CV(val)
WINDOWED_MAX	WINDOWED_MAX(CHAN:sig) or WINDOWED_MAX(val)
WINDOWED_MEAN	WINDOWED_MEAN(CHAN:sig) or WINDOWED_MEAN(val)
WINDOWED_MIN	WINDOWED_MIN(CHAN:sig) or WINDOWED_MIN(val)
WINDOWED_STD	WINDOWED_STD(CHAN:sig) or WINDOWED_STD(val)
WINDOWED_VAR	WINDOWED_VAR(CHAN:sig) or WINDOWED_VAR(val)

13.2 Process Management

Balefire uses a distributed software architecture with many small purpose build applications running simultaneously. Balefire also uses a process sever/client model to manage the starting, stopping, and diagnosis of these applications. The process sever maintains the list of processes required by the system and communicates this information to specific Process Clients. Process clients manage starting and stopping these processes.

13.2.1 Process Server

The process server stores all of the processes and process configurations of the entire system and communicates this information to specific Process Clients. The list of processes and associated configurations is called a 'process profile'. Process profiles are stored in a single configuration file.

13.2.1.1 Basic Process Server Configurations

The process sever offers a number of configurations specifying how to start a process. Below is a table outlining the available configurations:

Available Configuration	*Description*
client_name	The name of the client you intended to run the process.
process_name	The name of the process to start. This must be an executable file.
arguments	The processes arguments required for proper operation.
process_cmd	A command for how the process will be run. Available process commands: • start (Start the process once) • stop (Stop the process) • keep_alive (Restart the process if it dies)
publish_console	Publish the console output. Available arguments: • 0 (Do not publish console output) • 1 (Publish the console output)
keep_alive_ms	The number of milliseconds the keep_alive command sleeps before restarting the process.
uuid	Universally unique identifier. A unique identifier to specify the process. If not defined this will be automatically defined.

Table 27: Available configurations and descriptions for the process server.

```
process_server:
  commands:
    - client_name: your_client
      process_name: gss_nmea2gss
      arguments:
        - port=/dev/ttyS0
        - baud=9600
      process_cmd: start
      publish_console: 0
      keep_alive_ms: 1000
```

Illustration 47: Simple process server configuration file.

To add a process to the configuration file, add an additional command. Below is an example of a two-process configuration.

```
process_server:
  commands:
    - client_name: your_client
      process_name: gss_nmea2gss
      arguments:
        - port=/dev/ttyS0
        - baud=9600
      process_cmd: start
      publish_console: 0
      keep_alive_ms: 1000
    - client_name: your_client
      process_name: gss_nmea2gss
      arguments:
        - port=/dev/ttyS1
        - baud=4800
      process_cmd: keep_alive
      publish_console: 1
      keep_alive_ms: 1000
```

Illustration 48: Two-process server.

13.2.1.2 Multiple Process Severs

Multiple process severs can be used to provide a secondary process management system. Secondary process management systems can be used to provide a backup process manager on another processor or can be used to provide a second process profile. There can only be one active process server on a network. When a process sever starts up it listens for data from other process severs on the network. If another process sever is detected, the process sever will go into a dormant state and will not actively manage processes. The process sever will remain in the dormant state until the sever no longer detects other severs. At this time, the process sever will go into an active state and begin managing processes.

13.2.2 Process Client

Process Clients mange the processes of the process server. Each Process Client is started with a unique name. The process sever uses this name to arbitrate which client is intended to run each process. Process Clients do not maintain a configuration file and simply listen for commands from process servers. There can be multiple Process Clients on a single machine and across a network.

13.3 Tuning Systems

Tuning is done to enable optimum autopilot performance and requires changing parameter values in the configurations file to suit your system. openCMD reads the configuration file on startup and sets the parameters for all controllers.

The dynamics of your particular vehicle dictate the values. Different vehicles have different masses, buoyancy, sensors, and thrusters, and those variables have a legitimate impact on gains.

If you have an accurate hydrodynamic model of your system you can analytically determine the controller gains for a plant. If you don't, you have to use other methods for tuning.

13.3.1 How to Tune

To access tuning, go to Mission View and press Control +Alt+M. The tuning screen will display on the map grid.

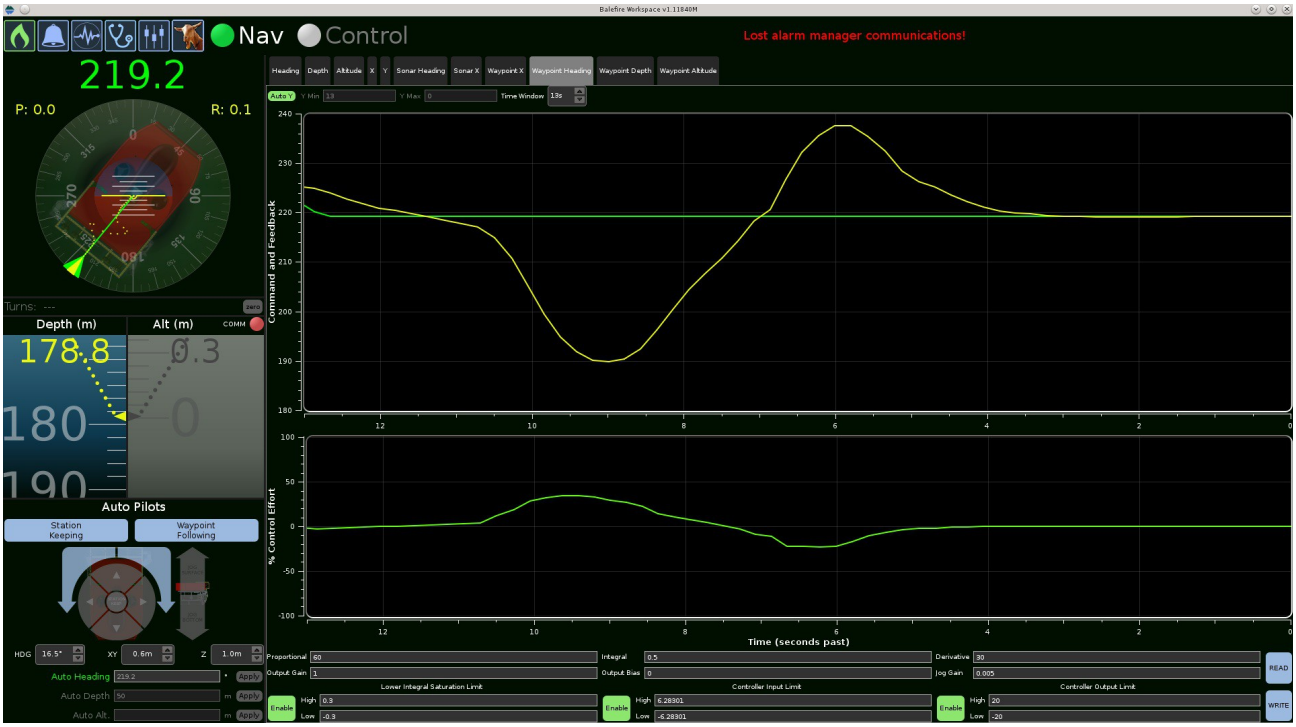


Illustration 49: Control Tuning utilities.

Select the value you wish to tune from the Autopilot Tuning Console at the bottom of the screen and click READ. The current settings will display.

Note: Begin tuning with proportional gain ('P'), located in the lefthand corner of the console.

You want the output to overshoot the signal and oscillate about the setpoint. When you're happy with 'P', you should move on to derivative gain to iron out the oscillations. Integral gain is then changed if there's an offset between the setpoint and feedback signal.

Choose a controller to begin tuning and plot the relevant control signals using the tuning widget. Press **WRITE** to save your changes.

Place the vehicle into auto mode and check the response. One way to do this is by using the main Mission View page. You can try step responses when tuning, like for heading at 15º steps at a time. It will also help to check dynamic control responses in addition to steps. To do this, turn on an autopilot like heading and drive forward or laterally. You'll want to check the chart in the widget view to see how close or far the feedback is from the command signal. For heading you will look at the compass rose, for depth you'll look at the depth dial, and so on.

13.3.2 Tuning Fields

The lower portion of the tuning screen has text fields to edit for the user to tune the system. To get the current configurations, press the **READ** button. This button is blue when active and green when selected. Once a field has been modified, the **WRITE** button will set the new configurations and the result can be visualized in the graphic display.

Tuning Field Name	*Description*
Proportional	Proportional gain ('P'): The proportional gain weights the proportional amount of error between the signal and the feedback, and maps it directly to the effort. The more error you have the more effort will be output.
Output Gain	Final scaler the controller output is multiplied by.
Integral	Integral Gain ('I'): The integral gain weights the sum of error over time. This is important because a proportional error weighted at 5% or less may not seem like much, but it keeps growing over time. Letting the integral gain build up to a high number means you'll end up carrying around a large bias that could weigh the vehicle down.
Output Bias	Scaler value added to controller output.
Derivative	Derivative Gain ('D'): The derivative gain weights the rate of error change over time.
Jog Gain	Rate of change of set point during a jog.
Lower Integral Saturation Limit	Click Enable to set the High and Low Limits. Limit on the error integral to prevent wind up.
Controller Input Limit	Click Enable to set the High and Low Limits. Max limits of feedback value
Controller Output Limit	Click Enable to set the High and Low Limits. Max value a controller can output.

Table 28: Tuning field names and descriptions.

13.4 Balefire Workspace Configuration

The Balefire Workspace is designed to allow the user to configure the appearance and functionality of the interface. The core of the workspace is simply a blank layout where users can place individual Balefire modules. Think of the workspace as an open canvas for controlling and visualizing vehicles systems. Balefire employs many different modules, providing a fully featured vehicle control, navigation, and diagnostic package. From the simplistic signal indication LED to the integrated Mission View and mapping module, Balefire's comprehensive set of modules give you the tools to efficiently and effectively conduct complex operators.

13.4.1 Balefire Widgets

Balefire's widgets give you the capability to display and control vehicle systems. To accomplish this, a set of core configurable modules has been developed around the primary communications type, PCOMMS. Below is a table of Balefire's core widgets:

Module Name	*Description*
Pcomms_led	PCOMMS-based status LED. A tristate LED with the following states: • Gray: No data received. • Red: False signal. • Green: True signal.
Pcomms_label	PCOMMS-based text label. A label for displaying static or dynamic text.
Pcomms_bargraph	PCOMMS-based bar graph. A bar graph for displaying the state of an analog signal.
Pcomms_btn	PCOMMS-based button. A configurable button for transmitting values on any of the native PCOMMS types; digital_t, analog_t, or message_t.
Pcomms_slider	PCOMMS-based slider. An analog slider for transmitting values in a specified range.
Gss_2axis_plot	PCOMS-based two-axis plotting utility.

Table 29: Balefire Widgets

13.4.2 Grid Layout Configuration

Balefire Workspace pages are generally configured using a grid layout. Widgets are often placed within these grids at locations specified within the workspace's configuration file. The process to alter a widget's location and properties is listed in the following sections.

13.4.2.1 PCOMMs LED

To place a PCOMMs LED in the workspace, you must first specify the item type. In this case we are placing a Pcomms_led. The object “top_bar_nav_led” is first declared within the balefire_bar_panel grid of the workspace.

Properties	*Available Arguments*	*Description*
item_type	Pcomms_led	Specifies what item type is desired.
object_name	desired_name_led	Specifies name for the item. In .yml file, the object is defined later on in the “desired_name_led” section.
row_start	Integer	Specifies the row start location of the LED. If row_start: 0, then the LED will be in the first row; row_start: 1, the LED will be in the second row and so on.
column_start	Integer	Specifies the column start location of the LED. If column_start: 0, then the LED will be in the first column; column_start: 1, the LED will be in the second column and so on.

Table 30: PCOMMs LED properties, available arguments, and descriptions.

Example Code from a PCOMMs LED in balefire_mbari_config_007.yml:

```
balefire_bar_panel:              # Grid that contains the Pcomms LED.
  items:
    - item_type: Pcomms_led      # Widget in balefire_bar_panel grid.
      object_name: top_bar_nav_led  # Widget's user defined name.
      row_start: 0               # Row start, within grid.
      column_start: 0            # Column start, within grid. specified.
```

Once the PCOMMs LED has been specified, you will want to set the properties of the LED. The PCOMMs LED has eight properties that can be used to specify the channel it listens to, name, size and alignment.

Property	*Available Arguments*	*Description*
status_channel	Relevant channel	Specifies the channel from which the LED will base its status.
status_signal	Relevant signal	Specifies the signal from which the LED will base its status.
description	User Defined Name	Specifies the LED's name. If “description: Nav” the LED will appear next to the word 'NAV' within the workspace.
label_position	North, South, East, West	Specifies position of label with respect to the LED: • North: Places Label above the LED • South: Places Label below the LED • East: Places Label to the right of the LED • West: Places Label to the left of the LED
label_halign	Left, right, center	Specifies horizontal alignment of the label.
label_valign	Top, bottom, center	Specifies vertical alignment of the label.
led_pixel_size	Integer	Specifies LED size in pixels.
label_pixel_size	Integer	Specifies label size in pixels.

Table 31: PCOMMS LED properties.

Example Code from “top_bar_nav_led” in balefire_mbari_config_007.yml:

```
# monitors openINS navigation solution reception
top_bar_nav_led:                 # Pcomms LED Definition follows.
  status_channel: OPENINS_NAV_SOLUTION # Status channel.
  description: Nav               # LED name, displays on workspace
  led_pixel_size: 35             # LED size.
  label_pixel_size: 35           # Label size.
  label_position: east           # Location of label wrt LED.
```

13.4.2.2 PCOMMs Label

To place a PCOMMs label in the workspace, you must first specify the proper item type and its location. The object “pi_vert_rpm_label” is first declared within the thruster_grid grid of the workspace.

Properties	*Available Arguments*	*Description*
item_type	Pcomms_label	Specifies what item type is desired.
object_name	desired_name_label	Specifies name for the item. In .yml file, the object is defined later on in the “desired_name_label” section.
row_start	Integer	Specifies the row start location of the label.
column_start	Integer	Specifies the column start location of the label.

Table 32: PCOMMS label properties, available arguments, and descriptions.

Example Code from a PCOMMs Label in balefire_mbari_config_007.yml:

```
thruster_grid:                  # Grid in which label is declared.
  row_stretches: 0 1
  items:
    - item_type: Pcomms_label       # Item type declared.
      object_name: pi_vert_rpm_label # User defined object name.
      row_start: 0                  # Row start, within grid.
      column_start: 1                 # Column start, within grid.
```

Once the PCOMMs label has been specified, you will want to set its properties. The PCOMMs label has eight properties that can be used to specify the channel it listens to, name, size, and alignment. They are as follows:

Property	*Available Arguments*	*Description*
status_channel	User defined channel	Specifies the channel from which the label will base its status.
status_signal	User defined signal	Specifies the signal from which the label will base its status.
description	User Defined Name	Specifies the label's name.
units	String	Units that the label displays.
precision	Integer	Number of decimal points displayed for analog values.
value_pixel_size	Integer	Set vertical height of the font used to display values.
description_pixel_size	Integer	Set vertical height of description.
units_pixel_size	Integer	Set vertical height of units.
description_north	True, False	If true, display description above; If false, display description to left.
datetime_format	String	Qt format specifier for datetime.
local_datetime	True, False	If true, display local date & time; If false, UTC date & time.
background_color	Color	Color used to render background.
text_color	Color	Text color.
label_halign	Left, right, center	Specifies horizontal alignment of the label.
timeout_sec	Integer	Set timeout time for label.
hide_inactive	True, False	If true, the value won't display when timed out; If false, the value displays.

Table 33: PCOMMS label properties, available arguments, and descriptions.

Example Code from “pi_vert_rpm_label” in balefire_mbari_config_007.yml:

```
pi_vert_rpm_label:                  # Definition of object follows
  #description: Vertical RPM          # Label name.
  status_channel: SUB_CTRL_PCOMMS_UI_STAT # Status channel.
  #units: RPM                       # Label Units.
  status_signal: s_vert_rpm_counter_value # Status signal.
  value_pixel_size: 14              # Value size in pixels.
  description_pixel_size: 14          # Description size in pixels.
  label_halign: center              # Label's horizontal alignment
```

13.4.2.3 PCOMMs Bar graph

To place a PCOMMs bar graph in the workspace, you must first specify the proper item type and its location.

Properties	*Available Arguments*	*Description*
item_type	Pcomms_bargraph	Specifies what item type is desired.
object_name	desired_name_bargraph	Specifies name for the item. In .yml file, the object is defined later on in the "desired_name_bargraph" section.
row_start	Integer	Specifies the row start location of the bar graph.
column_start	Integer	Specifies the column start location of the bar graph.

Table 34: PCOMMS bar graph item properties, available arguments, and descriptions.

Example Code from a PCOMMs Bar graph in balefire_mbari_config_007.yml:

```
tms_cam_left_column_grid:          # Declaration of widget in grid
  column_stretches: 1 1 0 0
  items:
    - item_type: Pcomms_bargraph     # Type of widget declared.
      object_name: pi_cam_bargraph_2  # User defined object name.
      column_start: 0              # Column start, within grid.
      row_start: 5                 # Row start, within grid.
```

Once the PCOMMs bar graph has been specified, you will want to set its properties. The PCOMMs bar graph has 11 properties that can be used to specify the channel it listens to, name, size and alignment. They are as follows:

Property	*Available Arguments*	*Description*
status_channel	User defined channel	Specifies the channel from which the bar graph will base its status.
status_signal	User defined signal	Specifies the signal from which the bar graph will base its status.
description	User Defined Name	Specifies the name that is associated with the bar graph in the workspace.
Units	String	Specifies the units that the bar graph displays. E.g., %, psi, meters.
minimum	Integer	Minimum value the bar graph displays. Often set to 0.
maximum	Integer	Maximum value bar graph displays. Often set to 100.
precision	Integer	Number of decimal places displayed on bar graph.
horizontal	True, False	Draw display horizontally if true; if false, vertical.
label_north	True, False	Display label above bar graph if true; if false, don't display above.
minimumWidth	Integer	If specified, passed through to Qt. See Qt documentation.
label_pixel_size	Integer	Set size of label in pixels.

Table 35: PCOMMS bar graph properties, available arguments, and descriptions.

Example Code from "pi_cam_bargraph_2" in balefire_mbari_config_007.yml:

```
pi_cam_bargraph_2:                 # Object definition follows.
```

```
  status_channel: PSI_PCOMMS_UI_STAT # status channel.
  status_signal: u_main_cam_shld_deg # Status signal.
  description: "Shld Cmd"          # User defined name, displays.
  units: deg                     # Bargraph Units
  minimum: 0                       # Bargraph minimum.
  Maximum: 90                      # Bargraph maximum.
#  minimumWidth: 50                 # Bargraph minimum width.
  MaximumWidth: 75                 # Bargraph maximum width.
  horizontal: false              # Display horizontal or vertical.
```

13.4.2.4 PCOMMs Button

To place a PCOMMs button in the workspace, you must first specify the proper item type and its location.

Properties	*Available Arguments*	*Description*
item_type	Pcomms_btn	Specifies what item type is desired.
object_name	desired_name	Specifies name for the item. In .yml file, the object is defined later on in the "desired_name_button" section.
row_start	Integer	Specifies the row start location of the button.
column_start	Integer	Specifies the column start location of the button.

Table 36: PCOMMS button item properties, available arguments, and descriptions.

Example Code from a PCOMMs Button in balefire_mbari_config_007.yml:

```
pi_gfi_tab_1_grid:                  # Grid declaration for btn.
  Items:
    - item_type: Pcomms_btn              # Item type declared.
      object_name: pi_topside_gfi_hold_btn_1 # User defined object name.
      row_start: 1                   # Row start, within grid.
      column_start: 2                  # Column start, within grid
```

Once the PCOMMs button been specified, you will want to set its properties. The PCOMMs button has 18 properties that can be used to specify the channel it listens to, name, size and alignment. They are as follows:

Property	*Available Arguments*	*Description*
status_channel	User defined channel	LCM channel monitored to set the button's state.
status_signal	User defined signal	LCM signal monitored to set the button's state.
output_channel	User defined channel	LCM channel the button will use to publish button presses.
output_signal	User defined signal	LCM signal the button will use to publish button presses.
active_color	Color	Color that is displayed when the button is active.
Ready_color	Color	Color that is displayed when the button is ready.
predicate	Chansig	Condition that must be true before the button can be activated.
ui_signal	User defined signal	Set workspace signal this button monitors for the state of its output signal.
cmd_signal	User defined signal	Command signal currently monitored to set the button's state.
require_confirmation	True, False	If true, state changes require operator confirmation. False, and buttons will click freely.
confirmation_message	String	Message that appears when button click is confirmed.
on_message	String	Message published when button is in “on” state.
off_message	String	Message published when button is in “off” state.
instant_post	True, False	If true, post instantly; If false, don't post instantly.
text_pixel_height	Integer	Set height of font in pixels.
radio_channel	User defined channel	Channel that radio button sends radio messages on.
radio_signal	User defined signal	Signal the radio button sends radio messages on.
radio_default	True, False	Set default value for radio button. If true, set button to true state.

Table 37: PCOMMS button properties, available arguments, and descriptions.

Example Code from “auto_alt_tuner_enable” in balefire_mbari_config_007.yml:

```
pi_topside_gfi_hold_btn_1:              # Button definition follows.
  text: "Hold"                          # Button text.
  output_channel: TOPSIDE_GFI_MUX_UI_CMD   # Output channel.
  output_signal: u_select_channel_0       # Output signal.
  predicate:                            # Predicate.
  radio_channel: TOP_GFI_HOLD_RADIO_BUTTON # Radio channel.
  radio_signal: topside_hold_1          # Radio signal.
```

13.4.2.5 PCOMMs Slider

To place a PCOMMs slider in the workspace, you must first specify the proper item type and its location.

Properties	*Available Arguments*	*Description*
item_type	Pcomms_slider	Specifies what item type is desired.
object_name	desired_name_slider	Specifies name for the item. In .yml file, the object is defined later on in the "desired_name_slider" section.
row_start	Integer	Specifies the row start location of the slider.
column_start	Integer	Specifies the column start location of the slider.
column_span	Integer	Specifies span of the slider in columns.

Table 38: PCOMMS slider item properties, available arguments, and descriptions.

Example Code from a PCOMMs Slider in balefire_mbari_config_007.yml:

```
pc_reservoir_vb_control_grid:                  # Slider's grid.
  items:
    - item_type: Pcomms_slider                 # Item type.
      object_name: pc_res_vb_ctrl_vb_setpoint_slider # Slider name.
      row_start: 0                             # Row start.
      column_start: 0                          # Column start.
      column_span: 2                           # Column span.
```

Once the PCOMMs slider has been specified, you will want to set its properties. The PCOMMs slider has 14 properties that can be used to specify the channel it listens to, name, size and alignment. They are as follows:

Property	*Available Arguments*	*Description*
output_channel	User defined channel	Specifies channel that slider output will be on.
output_signal	User defined signal	Specifies signal that slider output will be on.
predicate	Chansig	Condition that must be true before the button can be activated.
status_channel	User defined channel	Set the channel that will be monitored to set the state of the control.
ui_signal	User defined signal	Set the workspace signal the slider monitors for the state of its output signal.
status_signal	User defined signal	Set the signal that the slider monitors for the state of its output signal.
cmd_signal	User defined signal	Set the commanded signal the slider will monitor for the state of its output signal.
description	String	Set the descriptive text next to the value label.
units	String	Slider units.
precision	Integer	Set the number of decimal points displayed for analog values.
minimum	Integer	Set the minimum output analog control value.
maximum	Integer	Set the maximum output analog control value.
click_to_publish	True, False	If true, publishing will only happen if clicked; publish immediately if false.
double_click_reset	True, False	If true, double-click the slider to reset; if false, double-click does nothing.

Example Code from “pc_res_vb_ctrl_vb_setpoint_slider” in balefire_mbari_config_007.yml:

```
# define the buttons for the variable buoyancy compensator
pc_res_vb_ctrl_vb_setpoint_slider:              # Definition follows.
  description: "VB Set Point"                   # Name of slider.
  output_channel: BALEFIRE_SIGNAL_MAP_CMD       # Output channel.
  output_signal: SET_MAP_STATUS(vb_auto_set_point)# Output signal.
  Precision: 0                                  # Slider Precision.
  Minimum: 0                                    # Slider Minimum
  Maximum: 100                                  # Slider Maximum
  units: "%"                                    # Slider Units
  click_to_publish: true                        # Publish upon click.
  predicate: "BALEFIRE_MAP_STAT:vb_auto_mode"   # Predicate.
```

13.4.2.6 Greensea Two-Axis Plot

To place a two_axis plot in the workspace, you must first specify the proper item type and its location.

Properties	*Available Arguments*	*Description*
item_type	Gss_2axis_plot	Specifies what item type is desired.
object_name	desired_name	Specifies name for the item. In .yml file, the object is defined later on in the “desired_name” section.
row_start	Integer	Specifies the row start location of the two-axis plot.
column_start	Integer	Specifies the column start location of the two-axis plot.
column_span	Integer	Specifies span of the plot in columns.

Table 39: Two-axis plot item properties, available arguments, and descriptions.

Example Code from a GSS two-axis Plot in balefire_mbari_config_007.yml:

```
tms_cam_left_column_grid:       # Grid for 2axis plot widget.
  column_stretches: 1 1 0 0
  items:
    - item_type: Gss_2axis_plot   # Item type declaration.
      object_name: pi_cam_2axis_1 # User defined name.
      column_start: 2           # Column start, within grid.
      row_start: 5              # Row start, within grid.
#       row_span: 6             # Row span, within grid.
      column_span: 2            # Column span, within grid.
```

Once the GSS two-axis plot has been specified, you will want to set its properties. The two-axis plot has several properties, 13 are listed below:

Property	*Available Arguments*	*Description*
title	User defined title	Sets the plot's title.
xlabel	User defined label	Sets x-axis label.
ylabel	User defined label	Sets y-axis label.
auto_scale	True, False	Autoscale y-axis if true; do not autoscale y-axis if false.
xmin	Integer	Set the minimum value of the fixed x scale.
xmax	Integer	Set the maximum value of the fixed x scale.
ymin	Integer	Set the minimum value of the fixed y scale.
ymax	Integer	Set the maximum value of the fixed y scale.
background_color	Color	Set background color of plot.
show_grid	True, False	Show grid if true, do not show grid if false.
show_xscale	True, False	Show x scale if true, do not show if false.
show_yscale	True, False	Show y scale if true, do not show if false.
Markers – This sub section within the two-axis plot defines the markers	item_type, status_channel, x_signal, y_signal, marker_width, marker_color	item_type: instant_plot status_channel: SUB_CTRL_PCOMMS_UI_STAT x_signal: us_main_cam_pan_deg y_signal: us_main_cam_tilt_deg marker_width: 1 marker_color: whit

Table 40: Two-axis plot properties, available arguments, and descriptions.

Example Code from “pi_cam_2axis_1” in balefire_mbari_config_007.yml:

```
pi_cam_2axis_1:                    # Plot definition follows.
  xlabel: Pan                      # X axis label.
  ylabel: Tilt                     # Y axis label.
  Xmin: -90                        # X minimum value.
  Xmax: 90                         # X maximum value.
  Ymin: -90                        # Y minimum value.
  Ymax: 90                         # Y maximum value.
  title: Main Cam                  # Plot title
  markers:                         # Define plot markers.
    - item_type: instant_plot      # Item type
      status_channel: SUB_CTRL_PCOMMS_UI_STAT # Status-channel.
      x_signal: us_main_cam_pan_deg          # X signal, to be plotted.
      y_signal: us_main_cam_tilt_deg         # Y signal, to be plotted.
      marker_width: 1                        # Marker size, width.
      marker_color: white                    # Marker color, white.
    - item_type: instant_plot      # Same style Marker below
      status_channel: PSI_PCOMMS_UI_STAT
      x_signal: u_main_cam_pan_deg
```

```
y_signal: u_main_cam_tilt_deg
marker_width: 1
marker_color: green
```

13.4.3 Layouts

Balefire uses configurable layouts to organize and display modules. Modules must be contained within a layout to be displayed. Below is a table of the available configurable layouts:

Layout Name	*Description*
Gss_configurable_splitter	Configurable splitter.
Gss_configurable_grid	Configurable grid.
Gss_configurable_tabs	Configurable tabs.

Table 41: Balefire layouts.

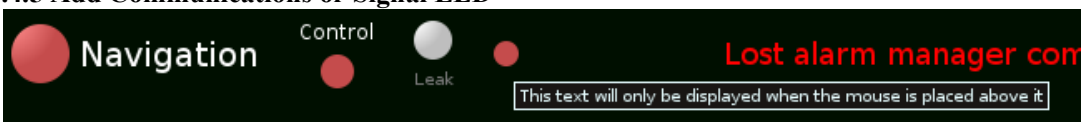
13.4.4 General Configurable Properties

Below is a table of the common properties available to any configurable module:

Property	*Description*	*Example*
visible	Hide module from viewing. [boolean]	visible: true
sizePolicy	Set the size policy of the module. Available options: • Fixed • Expanding • Minimum • Maximum • Preferred • MinimumExpanding • Ignored	sizePolicy: Fixed
width	The width of the module. [integer]	width: 20
enabled	This property holds whether the module is enabled. [boolean]	enabled: true

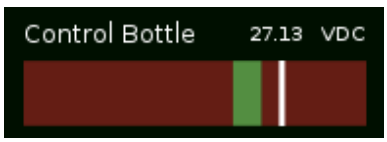
Table 42: Common properties available to configurable modules.

13.4.5 Add Communications or Signal LED




```
# the top bar panel is located directly to the right of the page
# selection icons.  Place items here that you want visible no
# matter the selected page
top_bar_panel:
  num_columns: 15 # pick a large number so we don't go to another row
  items:
    - item_type: Pcomms_led
      object_name: top_bar_nav_led
    - item_type: Pcomms_led
      object_name: top_bar_ctrl_led
    - item_type: Pcomms_led
      object_name: top_bar_leak_led
    - item_type: Pcomms_led
      object_name: top_bar_tooltip_led
# demonstrates a label-less led
top_bar_tooltip_led:
  status_channel: TEST_DATA
  description: This text will only be displayed when the mouse is placed above it
  label_position: tooltip
# monitors openINS navigation solution reception
top_bar_nav_led:
  status_channel: OPENINS_NAV_SOLUTION
  description: Navigation
  led_pixel_size: 35
  label_pixel_size: 20
# monitors openCMD control reception
top_bar_ctrl_led:
  status_channel: OPENCMD_CTRL_PCOMMS_STAT
  description: Control
  led_pixel_size: 20
  label_pixel_size: 12
  label_position: north
# Specific leak detector LED
top_bar_leak_led:
  status_channel: ROV_TELNET_STAT status_signal: S_LEAK_OK
  label_pixel_size: 10
  description: Leak
  label_position: south
```

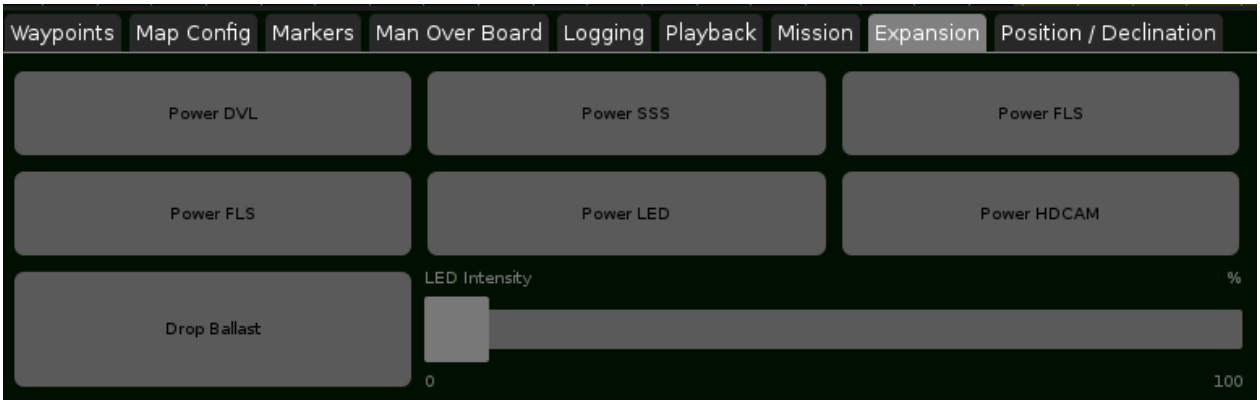
13.4.6 Add Signal Status Widgets



```
# bottom_controls is located on the flight page beneath the station keep widget
# like all configurable panels if it doesn't have any status or control widgets
# it keeps itself hidden
bottom_controls:
  num_columns: 3
  items:
    # add a text signal display - works for digital or analog signals
    - item_type: Pcomms_label
      object_name: test_label
    # add an analog bargraph
    - item_type: Pcomms_bargraph
      object_name: hotel_volts_bargraph
# if we have bottom lock the label will look like
# 'BTMLCK: True'
test_label:
  status_channel: OPENINS_NAV_SOLUTION
  status_signal: have_bottom_lock
  description: "BTMLCK:"

# this will display the current value of VEHICLE_STAT:S_HOTEL_VDC
# as well as any alarms configured for that signal in the alarm manager
hotel_volts_bargraph:
  status_channel: VEHICLE_STAT
  status_signal: S_HOTEL_VDC
  description: Hotel
  units: VDC
  precision: 2 # two decimal places
  minimum: 0   # min and max are for the graph bounds only
  maximum: 36
```

13.4.7 Add controls



(Configuration file only contains a subset of the pictured controls.)

```
1: # this will add two buttons and one slider to the expansion panel
```

```
 2:
 3:
 4: # located on mission view page beneath the map
 5: mv_expansion_panel:
 6:   num_columns: 2
 7:   items:
 8:     - item_type: Pcomms_btn
 9:       object_name: rov_dvl_pwr_btn
10:     - item_type: Pcomms_btn
11:       object_name: rov_ballast_drop_btn
12:     - item_type: Pcomms_slider
13:       object_name: rov_led_intensity_slider
14:       column_span: -1 # this makes the slider take the entire bottom row
15: rov_dvl_pwr_btn:
16:   text: Power DVL
17:   output_channel: ROV_TELNET_CMD
18:   output_signal: pwr_board1_en
19:   status_channel: ROV_TELNET_STAT
20:   status_signal: pwr_board1_en
21: rov_ballast_drop_btn:
22:   text: Drop Ballast
23:   output_channel: ROV_TELNET_CMD
24:   output_signal: pwr_board7_en
25:   status_channel: ROV_TELNET_STAT
26:   status_signal: pwr_board7_en
27:   # activate optional dialog that pops up requesting confirmation
28:   require_confirmation: true
29:   confirmation_message: "Dropping the ballast cannot be undone. Are you sure?"
30: rov_led_intensity_slider:
31:   description: "LED Intensity"
32:   output_channel: ROV_TELNET_CMD
33:   output_signal: led_intensity
34:   status_channel: ROV_TELNET_STAT
35:   status_signal: led_intensity
36:   precision: 0
37:   minimum: 0
38:   maximum: 100
39:   units: "%"
40:
41:
```

13.4.8 Creating a Layout

You can also place layouts within layouts to create more complex arrangements. Below is an example of adding configurable tabs to a custom page.

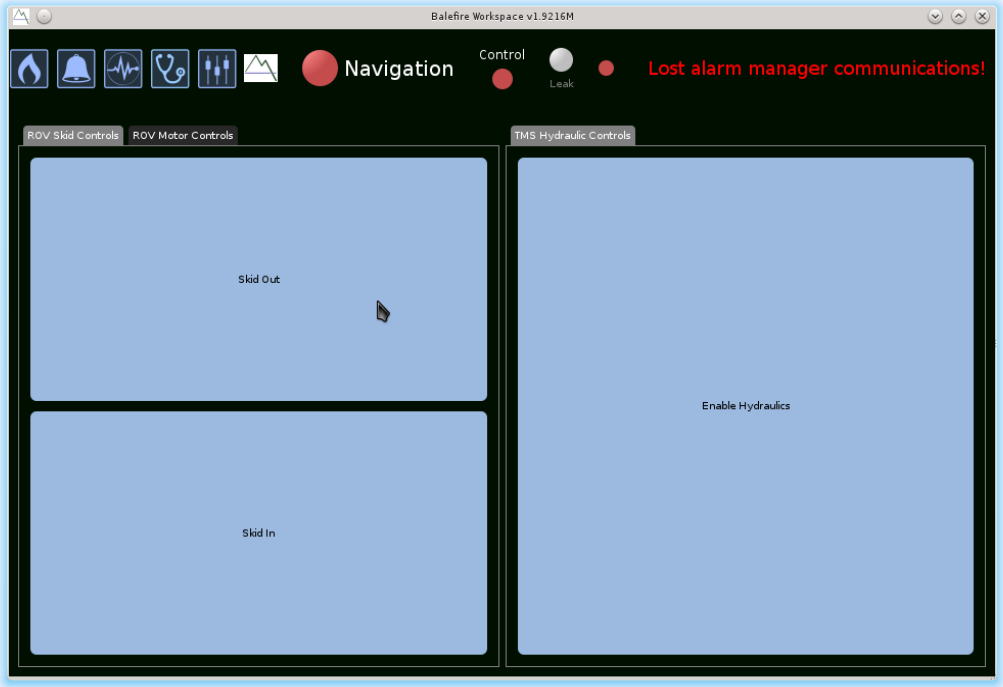


Illustration 50: Configurable tabs are added to a custom page.

```
# the control view page configuration (greensea icon)
controls_view:
  num_columns: 2
  items:
    - item_type: Gss_configurable_tabs
      object_name: control_view_left_tabs
    - item_type: Gss_configurable_tabs
      object_name: control_view_right_tabs
# the control view page, tabs on the LHS configuration
control_view_left_tabs:
  items:
    - item_type: Gss_configurable_grid
      object_name: control_view_rov_skid_controls_grid
      tab_text: ROV Skid Controls
    - item_type: Gss_configurable_grid
      object_name: control_view_rov_motor_controls_grid
      tab_text: ROV Motor Controls
# first tab on the lhs
control_view_rov_skid_controls_grid:
  num_columns: 1
  items:
```

```
    - item_type: Pcomms_btn
      object_name: rov_control_view_skid_out_btn
    - item_type: Pcomms_btn
      object_name: rov_control_view_skid_in_btn
# the first button on the first tab on the LHS
rov_control_view_skid_out_btn:
  text: Skid Out
  output_channel: ROV_CMD
  output_signal: U_ROV_SKID_OUT
# the second button on the first tab on the LHS
rov_control_view_skid_in_btn:
  text: Skid In
  output_channel: ROV_CMD
  output_signal: U_ROV_SKID_IN
# second tab on the lhs
control_view_rov_motor_controls_grid:
  num_columns: 1
  items:
    - item_type: Pcomms_btn
      object_name: rov_control_view_motors_en_btn
# the only button on the second tab on the lhs
rov_control_view_motors_en_btn:
  text: Enable Motors
  output_channel: ROV_CMD
  output_signal: U_ROV_MOTORS_EN
# the tab widget on the right
control_view_right_tabs:
  items:
    - item_type: Gss_configurable_grid
      object_name: control_view_tms_hydraulic_control_grid
      tab_text: TMS Hydraulic Controls
# grid in the single tab on the right hand side
control_view_tms_hydraulic_control_grid:
  num_columns: 1
  items:
    - item_type: Pcomms_btn
      object_name: tms_control_view_hydraulic_en_btn
# the single button on the single grid in the single tab on the right hand side
tms_control_view_hydraulic_en_btn:
  text: Enable Hydraulics
  output_channel: TMS_CMD
  output_signal: U_TMS_HYDRAULIC_EN
```

13.5 System Configuration Table

Configuration File	*Description*	*Reference Document*
balefire_mbari_config_007.yml	Operators Balefire Workspace configuration File.	Ventana_workspace_operator manual: Balefire Workspace Configuration
mbari_hud.yml	Video Overlay Balefire Workspace configuration file.	Ventana_workspace_operator manual: Balefire Workspace Configuration
process_server_config.yml	Ventana process sever configuratoin file.	Ventana_workspace_operator manual: Process Management
ventana_thrust.yml	Thrust mixing configuration file.	opencmd_icd
ventana_alarms.yml	Ventana alarm manager configuration file.	Ventana_workspace_operator manual: Alarm Manager
seabird.conf	CSV to PCOMMS device application for the Seabird CTD. This configuration file defines the signal names for the fields in the received CSV string.	Ventana_workspace_operator manual: gss_csv_to_pcomms
read_config.yml	FO Micro Modbus read configuration file.	Ventana_workspace_operator manual: Modbus Configuration
write_config.yml	FO Micron Modbus write configuration file.	Ventana_workspace_operator manual: Modbus Configuration
ventana_maps.yml	Ventana signal mapper configuration file.	Ventana_workspace_operator manual: Modbus Configuration
bp_config.yml	Pan and tilt bi-polar configuration file.	opencmd_icd
pid_config.yml	Pand and tilt PID controller configuration file.	opencmd_icd
belly_pack_read.yml	Belly pack Modbus read configuration file.	Ventana_workspace_operator manual: Modbus Configuration
control_system_read.yml	CSI Modbus read configuration.	Ventana_workspace_operator manual: Modbus Configuration
control_system_write.yml	CSI Modbus write configuration.	Ventana_workspace_operator manual: Modbus Configuration
corecan_read.yml	Corecan Modus read configuration.	Ventana_workspace_operator manual: Modbus Configuration
corecan_write.yml	Corecan Modbus write configuration.	Ventana_workspace_operator manual: Modbus Configuration
pilot_station_read.yml	PSI Modbus read configuration.	Ventana_workspace_operator manual: Modbus Configuration
vehicle_can_read.yml	Vehicle can Modbus read configuration.	Ventana_workspace_operator manual: Modbus Configuration
vehicle_can_write.yml	Vehicle can Modbus write configuraiton.	Ventana_workspace_operator manual: Modbus Configuration

Configuration File	*Description*	*Reference Document*
opencmd_bp_auto_hdg2xy.yml	openCMD bi-polar controller configuration file for auto mode.	opencmd_icd
opencmd_bp_ecef_decoupled.yml	openCMD bi-polar controller configuration file for FBW mode.	opencmd_icd
opencmd_lcm_config.yml	openCMD LCM subscription configuration file.	opencmd_icd
opencmd_motion_auto_hdg2xy.yml	openCMD FWB controller configuration file.	opencmd_icd
opencmd_motion_ecef_decoupled.yml	openCMD Auto mode controller configuration file.	opencmd_icd
opencmd_scaling.yml	openCMD signal scaling file.	opencmd_icd
opencmd_signals.yml	openCMD signal mapping file.	opencmd_icd
opencmd_state_alarms.yml	openCMD state alarms file.	opencmd_icd
opencmd_task_alarms.yml	openCMD task alarm file.	opencmd_icd
ventana_app.yml (opencmd)	openCMD application file.	opencmd_icd
openins_gains.yml	openINS filter gains file.	openins_icd
openins_lcm_config.yml	openINS LCM subscription file.	openins_icd
openins_scaling.yml	openINS signal scaling file.	openins_icd
openins_signals.yml	openINS signal mapping file.	openins_icd
openins_state_alarms.yml	openINS state alarms file.	openins_icd
openins_task_alarms.yml	openINS task alarms file.	openins_icd
openins_task_config.yml	openINS task configuration file.	openins_icd
openins_xforms.yml	openINS device transfromation file.	openins_icd
ventana_app.yml (openins)	openINS application file.	openins_icd
openmngr_joystick.yml	openMNGR joystick configuration file.	openmngr_icd
openmngr_lcm_config.yml	openMNGR LCM subscription file.	openmngr_icd
openmngr_signals.yml	openMNGR signal scaling file.	openmngr_icd
openmngr_state_alarms.yml	openMNGR state alarm file.	openmngr_icd
openmngr_task_alarms.yml	openMNGR task alarm file.	openmngr_icd
openmngr_task_config.yml	openMNGR task configuration file.	openmngr_icd
ventana_app.yml (openmngr)	openMNGR application file.	openmngr_icd
waypoints.yml	openMNGR active waypoint file.	openmngr_icd

Table 43: System configuration files, descriptions, and references.

13.6 Modbus Configurations

Modbus is a communications protocol used by many commercially available devices, for more information on Modbus see the website at www.modbus.org. openSEA has support for most Modbus protocols. gss_modbus and all applications based on gss_modbus gives users the ability to configure and control how Modbus registers are handled. gss_modbus uses the pcomms_t type for both commanding a register to output a signal and to publish the current state of a register. When configuring a Modbus application you must define two configuration files, one for reading and one for writing. Both configurations have identical formats.

13.6.1 Modbus Configuration File

Below is a table outlining the available configurations included in the gss_modbus application suite:

Available Configuration	*Description*
reg_num	The register number to command or request status.
name	The name of the pcomms_t signal to either command the register or publish the status of the register.
value	The default value of the register.
reg_type	The Modbus register type. Available configurations: • holding_reg • input_reg • discrete_input • coil
data_type	The Modbus register data type. This will define how gss_modbus reads the register. Available configurations: • ETYPES_Bool • ETYPES_Int8 • ETYPES_Int16 • ETYPES_Int32 • ETYPES_Ieeefloat
interval	For read configurations. The interval is the desired polling rate in seconds. For write configurations. The interval is the amount of time in seconds the system requires a user to command the register before defaulting to the configured default value. If the interval value is set to 0 the register will be set to the default on start up but never again. If the interval configuration is less than zero or missing from the configuration the register will never be set to default.

Table 44: Available Modbus configuration files.

Below is an example of a gss_modbus configuration:

```
holding_regs:
  regs:
    - reg_num: 0x000
      name: s_reg_1
      data_type: ETYPES_Int16
      reg_type: holding_reg
      interval: .05
    - reg_num: 0x001
      name: s_reg_2
      data_type: ETYPES_Int16
      reg_type: holding_reg
      interval: .05
```

Illustration 51: Modbus configuration.

13.7 NMEA Data in Balefire

Balefire provides facilities for brining standard NMEA data into the system. openSEA provides extensive support for receiving and transmitting NMEA sentences. In general, users must publish the data using LCM to display, log, and map data in Balefire. The basic process for using NMEA data is processing the incoming data and publishing the data over LCM. The process for building and transmitting NMEA data is just the opposite. LCM messages must be received , built into NMEA messages, and transmitted to the end user. Balefire provides support for both receiving and transmitting NMEA data.

13.7.1 gss_nmea2gss

gss_nmea2gss is a serial application designed to receive incoming NMEA data and transmit the data over LCM for use in the Balefire system. gss_nmea2gss provides arguments to specify serial settings and how to convert incoming NMEA data into different LCM types. You may specify multiple output channels to publish different data types. Below is a table describing gss_nmea2gss's available arguments:

Argument	*Description*	*Example*
port=	The serial port to open for receiving NMEA data.	port=/dev/ttyS0
baud=	The serial baud rate.	baud=9600
settings=	The serial settings to use for the serial port. The format is a comma separated list of the following: • Baud Rate • Data bits • Parity • Stop bits • Flow Control	settings=9600,8,n,1,off
logging_level=	The error logging level of the application. Available arguments: • debug • info	logging_level=debug

Argument	*Description*	*Example*
	• warning • severe • fatal	
cfg_file=	The path to the serial configuration file.	cfg_file=~/path/to/your_cfg_file.yml
gps_channel=	Transmit a gps_data_t LCM type on reception of relevant NMEA strings.	gps_channel=YOUR_CHANNEL
pose_channel=	Transmit a pose_stat_t LCM type on reception of relevant NMEA strings.	pose_channel=YOUR_CHANNEL
alt_channel=	Transmit a altitude_stat_t LCM type on reception of relevant NMEA strings.	alt_channel=YOUR_CHANNEL
depth_channel=	Transmit a depth_stat_t LCM type on reception of relevant NMEA strings.	depth_channel=YOUR_CHANNEL
hdg_channel=	Transmit a compass_stat_t LCM type on reception of relevant NMEA strings.	hdg_channel=YOUR_CHANNEL
gga_only=	Only transmit data when a GGA message is received.	gga_only=true
use_fix_time=	Use the fix time received in the NMEA message as the LCM messages time stamp.	use_fix_time=true
help=	Application usage.	help=true

Table 45: Available arguments and descriptions for gss_nmea2gss.

13.7.1.1 gss_nmea2gss Example

Below is an example of running gss_nmea2gss:

gss_nmea2gss port=/dev/ttyS0 baud=4800 pose_channel=TEST_NMEA_POSE

13.7.2 gss_gss2nmea

gss_gss2nmea is an application designed to transmit NMEA data from Balefire data. gss_gss2nmea subscribes to channels within the Balefire system and retransmits the data using the NMEA protocol over both serial and UDP. When using gss_nmea2gss you must specify the types gss_gss2nmea subscribes to and the NMEA message to be output. Below is a table describing the available arguments for the application:

Argument	*Description*	*Example*
port=	The serial port to open for receiving NMEA data.	port=/dev/ttyS0
baud=	The serial baud rate.	baud=9600
logging_level=	The error logging level of the application. Available arguments: • debug • info • warning • severe	logging_level=debug

Argument	*Description*	*Example*
	• fatal	
cfg_file=	The path to the serial configuration file.	cfg_file=~/path/to/your_cfg_file.yml
subscribe_channel=	The LCM subscribe channel for receiving data to convert to NMEA.	subscribe_channel=DATA_CHANNEL
publish_hz=	The maximum NMEA output data rate. [Hz]	publish_hz=5
serial_message_out=	The NMEA output message format. This argument is for both serial and UDP. You may use multiple message output arguments to output multiple NMEA strings on the same port. Available options: • GGA • HDT • RVR • VTG • DPT • HPR • TSS1 • PAUV • VHW • RMC	serial_message_out=GGA
input_type=	The LCM data type that will be received on the subscription channel. Available options: • NAV_SOLN • GPS • COMPASS • IMU	input_type=NAV_SOLN
udp_port=	The UDP port the NMEA data will be transmitted to.	udp_port=60000
udp_src_adr=	The source IP address of the UDP port.	udp_src_adr=192.168.1.234
udp_dst_adr=	The destination IP address of the UDP messages.	udp_dst_adr=192.168.1.235
transport=	NMEA message transport type. Available options: • SERIAL • UDP	transport=SERIAL
help=	Application usage.	help=true

Table 46: Available arguments, descriptions, and arguments for gss_gss2nmea.

13.7.2.1 gss_gss2nmea Example

```
gss_gss2nmea port=/dev/ttyS0 baud=4800 subscribe_channel=OPENINS_NAV_SOLUTION
input_type=NAV_SOLN serial_message_out=GGA serial_message_out=HPR
```

13.8 gss_csv_to_pcomms

gss_csv_to_pcomms is a serial application designed to input a serial CSV string and transmit the data as a pcomms_t. This application uses a configuration file to identify and name the elements in the CSV string. The configuration file has a name on each line in the order of the CSV fields. Below is an example of the configuration file structure:

```
Temperature
Conductivity
Pressure
Analog0
Analog1
Analog2
Analog3
```

Illustration 52: gss_csv_to_pcomms configuration file.

This configuration file expects the following csv format:

temperature,conductivity,pressure,analog0,analog1,analog2,analog3

13.9 Application Shortcuts

Press the F1 key to access application shortcuts. The shortcuts are displayed in a pop-up over the map.

13.9.1 Application Keyboard Shortcuts

Command	*Result*
Ctrl + Q	Quits application
Alt+ Shift + D	Toggles Data View button
Alt + Shift + R	Resets window configuration

Table 47: Application keyboard shortcuts.

13.9.2 Mission View Keyboard Shortcuts

Command	*Result*
Alt + B	Toggles Measure mode
Alt + T	Toggles Center on Vehicle
Double left-click	Zoom in
Double left-click + shift key	Zoom out
Rotate mouse wheel	Zoom in and out

Table 48: Mission View keyboard shortcuts.

14 Glossary of Terms

Term	*Definition*
Alarm	Configured system warning.
Bearing	The direction from the vehicle to an object (such as the ship).
DVL	Doppler Velocity Log. Device that determines sound waves bouncing off the sea floor to determine the velocity vector of the vehicle.
ECEF	Earth Centered Fixed Navigation Frame. Common frame of reference used by openSEA in both openINS and openCMD applications. A transform configuration provides the relationship of each sensor to the vehicle frame and openINS transforms the vehicle frame to ECEF.
HUD	Heads-up Display. The Map HUD shows data superimposed on your map that displays the vehicle name, the current location in degrees latitude/longitude, the pointer (your cursor) location in map coordinates, measurements taken, and the map's grid measurement.
IMU	Inertial Measurement Unit. The primary sensor for directly sensing vehicle motion. Provides accelerations and rotational rates. openINS uses data from the IMU as the high-rate process in its estimate filter.
INS	Inertial Navigation System. The set of software specific to providing the navigation and sensor fusion functions for the vehicle as well as a final state estimate.
LBL	Long Base Line underwater acoustic positioning.
Marker	A plotted point assigned to a hazardous point on the grid.
MOB	Man Overboard. A special purpose shortcut that marks the vehicle's current location on the map.
PCOMMS	Primary Communications. “Primary Communications”, or PCOMMS, in openSEA parlance refers generally to the traditional communications from a topside operator interface to the subsea system. Roughly, it is the generic communications between the vehicle components (devices, navigation, and control) and topside components (operator interfaces).
Pitch	The vehicle's movement around the y-axis. For example the vehicle will 'nose' up or down.
Roll	The vehicle's rotation about the x-axis. This can be seen in body tilt from side to side.
Tolerance	The allowed space that is predefined between the vehicle and a waypoint that is considered to be arrival.
Workspace	The graphical user interface. The means by which a user and system interact.
USBL	Ultra-Short Baseline underwater acoustic positioning.
Velocity	Rate of change of an object's position.
Down Velocity	The rate of travel in downward direction.
Mean Velocity	The average velocity.
Waypoint	A plotted point assigned as a destination marker for vehicle's path.
Yaw	The vehicle's rotational travel from the z-axis (vertical) or crablike movement.

Table 49: Glossary of terms used in the document.

End Page