
# Ventana Control System Upgrade

*Software Architecture*

***Subject:*** *This document defines the software architecture for the Ventana control system upgrade. It provides a list of applications running on the various computing nodes and describes how the applications communicate with each other to meet system requirements. Taken as a whole, it provides a complete picture of the network communications that are necessary for the software component of the control system to operate.*

***Proprietary and Confidential Information***

*This document, and the technical information contained herein, is solely the property of Greensea Systems, Inc. and MBARI. Unauthorized distribution or copying of this document or the information it contains is prohibited without approval from Greensea Systems, Inc. or MBARI.*

## Revision History

### Revision 009

| *Revision* | *Initials* | *Date* | *Comments* |
|---|---|---|---|
| 001 | Apo | 12/08/14 | Document initiated, content defined. |
| 002 | Jdr | 12/09/14 | Reviewed and clarified. |
| 003 | Apo | 03/30/15 | Updated after integration. |
| 004 | Ljd | 05/05/15 | Edited content. |
| 005 | Clr | 06/04/15 | Reviewed. |
| 006 | Sno | 06/07/15 | Edited content. |
| 007 | Ljd | 06/11/15 | Added content. |
| 008 | Ljd | 06/18/15 | Added and edited content. |
| 009 | Meb | 06/22/15 | Proofread and edited. |

*Table 1: Revision history.*

## Table of Contents

## Index of Tables

## Illustration Index

# Ventana Control System Upgrade

## *Software Architecture*

## 1 Introduction

This Software Architecture document gives an overview of the distinct applications that work together to provide the software component of *Ventana*'s control system. Individual applications communicate using a publish-subscribe network messaging mechanism. Applications publish information over a channel to which one or more other applications may subscribe to receive information. Individual units of information published on a channel are called signals. This document will outline the responsibility of each application, the computer where it executes, the channels it publishes, and those to which it subscribes. This will enable the reader to both understand the system as a whole and quickly isolate malfunctions to the offending computer and application.

## 2 Computing Nodes

The applications that comprise the software component of the vehicle control system are distributed over several computing nodes. There are four primary computing nodes by function in the *Ventana* control system design: a pilot computer, a copilot computer, the topside computer and the control system computer. While this document proposes a distribution of applications across these computing nodes for load balancing and other purposes, it should be noted that as Greensea applications use a network messaging scheme, a given application can be run on any computing node to produce a functionally identical system (especially since serial access is provided through MOXA serial servers).

## 3 Topside Applications

The topside applications are responsible for aiding operator awareness of *Ventana*'s state and to forward operator commands to the vehicle control system. The topside applications may broadly be divided into operator workspace, Alarm Manager, and Signal Mapping functional groups. Each group is described below.

### 3.1 Operator Workspace

The operator workspace provides graphical representations of *Ventana*'s state, configurable as well as predefined controls, and a means to view and configure other select applications in the control system. The operator workspace is organized around 'views' of the system that combine context-specific system state and controls. The organizing principle behind each view is listed below.

#### 3.1.1 Flight View

The flight view provides the primary operator controls for vehicle flight. The current attitude, position, and on-screen flight controls for the vehicle are displayed here. The operator may import navigational charts, as well as define navigational waypoints and markers, in addition to many other features.

#### 3.1.2 Alarm View

The alarm view provides operator control of the Alarm Manager system. Using this view, the operator may create and edit alarms as well as view the current state of all alarms in addition to past alarm events.

#### 3.1.3 Signal Mapping View

The signal mapping view provides operator control of the signal mapping system. Using Signal Mapper, the operator may list, search, create, and edit signal maps. Signal maps are used to map input to output for buttons, joystick positions,

transforms, conversions, and many other control system actions.

### 3.1.4 Device View

The device view provides dedicated displays of the current state of selected hardware devices in the system. Data is dynamically updated from the topside and vehicle with active communications and displayed. All raw device-specific data is displayed when available, along with a green indicator for active data feed. This page only displays the real time data.

### 3.1.5 Data View

The data view is a two-tabbed screen containing signal plot and process data. "Signal View" provides value inspection, searching and plotting of all channels and signals used by the control system. "Process View" displays the current information about all of the processes that comprise the system. The data displayed consists of the program UUID(ID), the system-created process ID, program status (Running and Last Updated), the program's client, the process' name (Process Name), its arguments, a program's restart count, and specific commands to the program's run status and publishing method.

### 3.1.6 Control View

Provides a full-screen area for user-defined control and status elements. This view has been designed in conjunction with MBARI staff to provide an interface similar to their existing control system.

## 3.2 openSEA Core Applications

The following diagram illustrates the channels the operator workspace uses to communicate with other openSEA core applications, as well as applications specific to the *Ventana*:

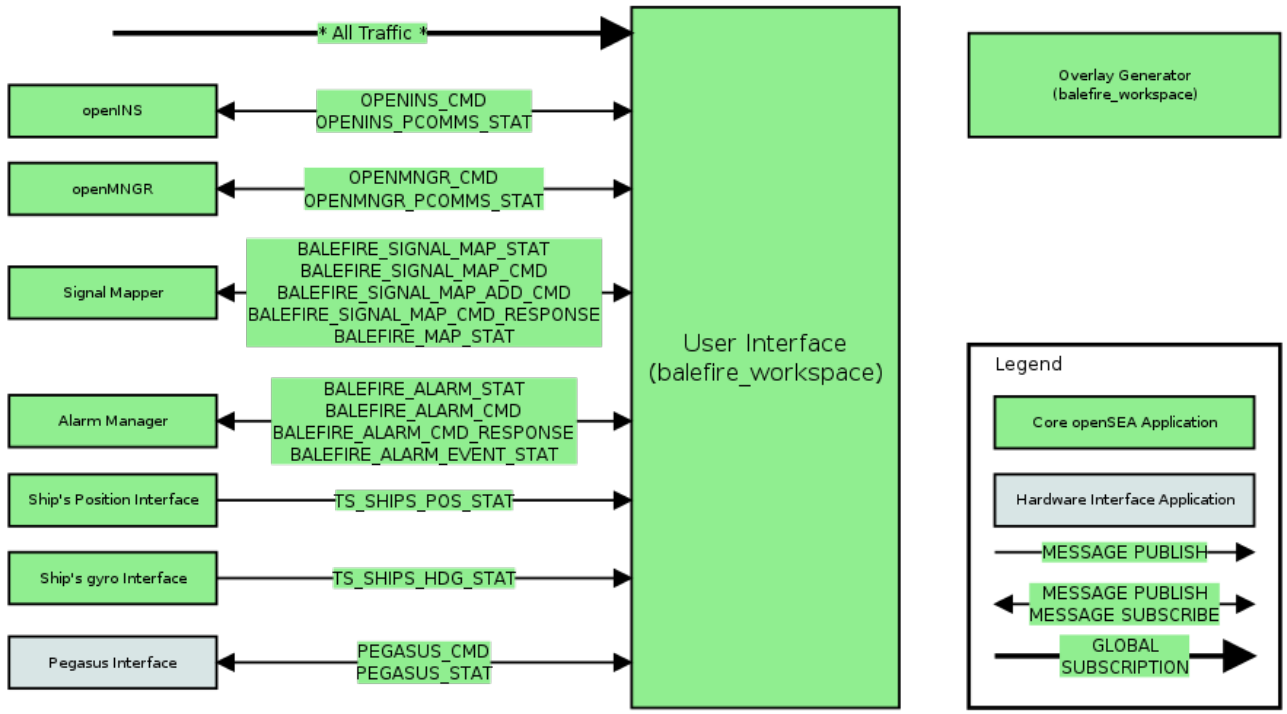


*Illustration 1: Operator workspace primary communication channels diagram.*

### 3.3 Alarm Manager

The alarm management system monitors all channels and signals for operator-defined exceptional situations. The complete state of all defined alarms is periodically published over the channel BALEFIRE_ALARM_STAT. Alarm Manager monitors the channel BALEFIRE_ALARM_CMD for alarm edits, deletions, and remote configuration file save and restore commands. The result of these operations will be published over the channel BALEFIRE_ALARM_CMD_RESPONSE.

Alarms may be configured to publish a Boolean signal that follows the active state of the alarm. These signals may be forwarded by Signal Mapper to effect configurable automatic responses to exceptional situations.

The following diagram illustrates the primary network connections that comprise the alarm management system:

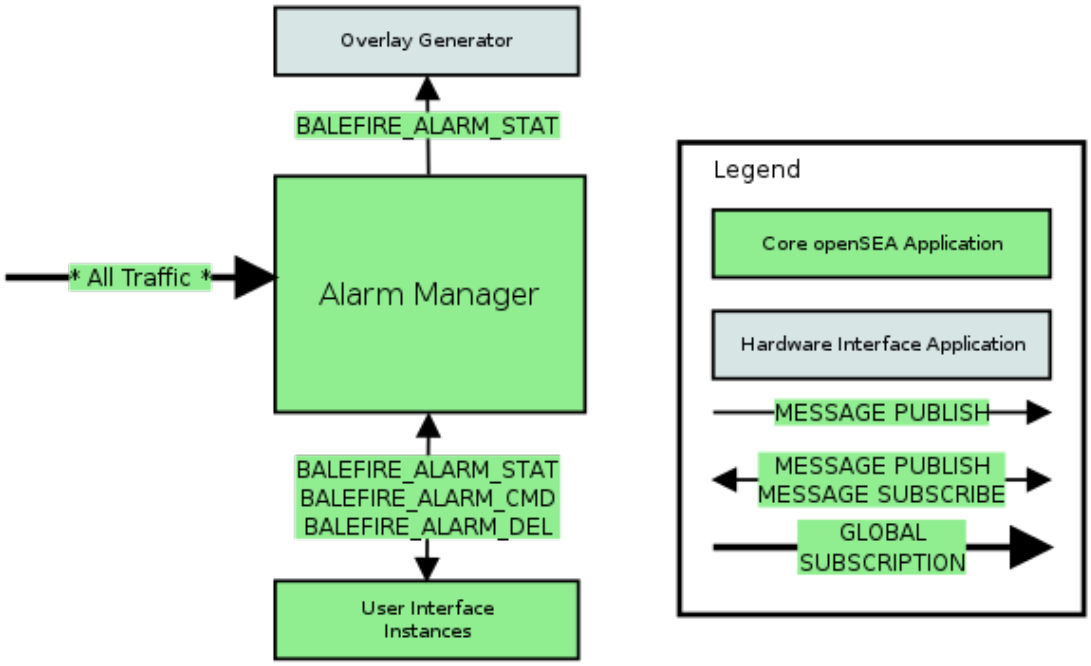


*Illustration 2: Alarm management system primary communication channels diagram.*

### 3.4 Configuration

Alarm Manager maintains its definitions of exceptional situations in a configuration file. While operators may edit this file directly (when Alarm Manager is not running) it is preferred they use the provided facilities in the workspace to affect alarm edits. A sample alarm configuration is listed below:

```
- id_object: "{8fa7d42c-f8b2-4f10-9f28-8be5416bb284}"
  input: VEHICLE_GFI_MUX_UI_STAT:us_vehicle_gfi_11
  transform: GREATER(8191)
  severity: Info
  priority: 0
  message: FO ISO 12VDC GROUND FAULT
  action: publish=!VEHICLE_GFI_ALARM_STAT:vehicle_gfi_11
  note: ""
```

The required fields are "input" and "transform". Alarm Manager will create suitable defaults for any missing value.

"id_object" is a universally unique identifier for the alarm. It may be omitted when hand-editing, whereupon Alarm

Manager will assign one the next time the configuration file is read.

"input" is the alarm input channel and signal. Each time the signal is published over the channel, the alarm will be evaluated.

"transform" defines the operation to apply to the value of "input" to determine if "input" is in an alarm state. Alarms use the same set of transforms as Signal Mapper. An alarm is considered in active state of the input signal value is not filtered by one of the transforms. In the example above, the "GREATER" filter drops values that are not strictly greater than 8191.

"severity" defines the importance of the alarms. Severity can be "Info", "Warning", "Severe" or "Fatal". Alarm Manager orders alarms by severity so the operator is alerted to more dire circumstances first.

"priority" allows the operator to rank alarms within a severity. Higher priority numbers will be displayed before others.

"message" is the human-readable message that is displayed when the alarm is active.

"action" is an optional step to take when the alarm is active. Applications may monitor BALEFIRE_ALARM_EVENT_STAT for an alarm or action of interest. Alarm Manager also supports a 'publish' action that publishes a digital signal that follows the state of the alarm.

"note" is an optional alarm note displayed on the alarm manager page of the operator workspace.

# 4 Signal Mapper

The signal mapping system monitors all channels and signals for operator-defined signal transformations. This system provides the operator configurable assignment of physical controls to logical functions in addition to scaling of analog 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.

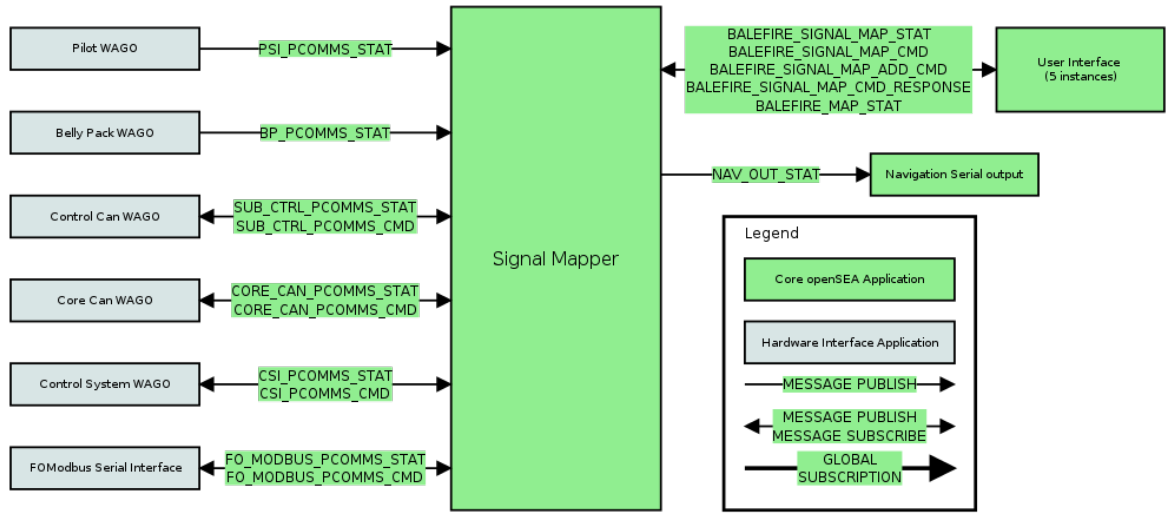


*Illustration 3: Signal mapping system primary communication channels diagram.*

## 4.1 Configuration

Signal Mapper maintains its definitions of signal maps in a configuration file. While operators may edit this file directly (when Signal Mapper is not running) it is preferred they use the provided facilities in the operator workspace to affect edits. 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 required fields are "input" and "output". Signal Mapper will create suitable defaults for any missing value.

"id_object" is 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" is the input channel and signal. Each time the signal is published over channel, the signal map will be evaluated.

"output" is 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" is 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 above 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.

### 4.2 Transforms

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. 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. |
| 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's converted to analog with the argument |

| Filter Name | Parameters | Description |
|---|---|---|
| | | bias. |
| 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<br>- input minimum<br>- deadband low<br>- deadband high<br>- input maximum<br>- output min<br>- output null<br>- 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. |
| 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. |

| *Filter Name* | *Parameters* | *Description* |
|---|---|---|
| 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. |
| TOGGLE | A single digital channel:signal pair. | Outputs the opposite of the last value of its parameter channel:signal. Drops non-digital inputs. |
| 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. |

| *Filter Name* | *Parameters* | *Description* |
|---|---|---|
| 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. |
| BOUNDS | Two scalar values: min and max. | Provides value bounds defined by minimum and maximum limits. |
| RATE_OF_CHANGE | An optional channel:signal pair representing containing the value to calculate the rate of change over. | Performs a calculation for rate of change and returns that value. |
| WINDOWED_SNR | The size of the window as a scalar or an analog channel:signal. | Calculates and returns the signal-to-noise ratio. |

### 4.3 openPROCESS

openPROCESS provides the system designer a means to describe the complete set of applications, their arguments, configuration files, and the host where each should execute. Collectively, this information defines a specific deployment of the software component of the *Ventana* control system.

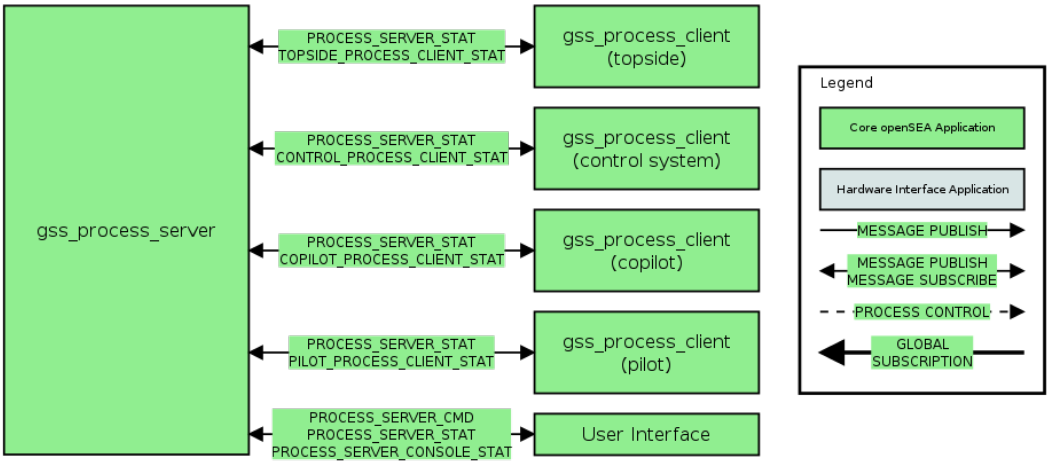


*Illustration 4: openPROCESS primary communication channels diagram.*

### 4.4 Architecture

openPROCESS utilizes a client-server architecture. The server maintains the desired configuration of the complete system: the name of each process, the client who should run it, optional arguments, and whether the client should restart the process if it stops. The client, in turn, is responsible for monitoring the server's command channel and starting, restarting, and stopping processes if its current state does not match the latest command.

Either the client or server processes themselves may be (re)started at any time. In the absence of a server, the clients continue executing their last command. In the absence of a client, the server continues to issue commands even though no one is there to execute them. If a client is stopped, all of the applications it started will be stopped as well. As soon as a client is started, and the client receives a command from the server, its applications will all be started.

### 4.5 Configuration

The process server is configured via a YAML-formatted file. An example entry from a 'live' file is provided below:

```
- time_unix_sec: 1426208171.35397
  count_publish: 1
  sender_id: mbari:11566
  client_name: padauk
  uuid: 00d5cd19-79d6-4ec5-9798-e869462cdb56
  process_name: balefire_workspace
  arguments:
    - --cfg-file
    - /home/ventana/gss_configs/balefire_mbari_config_007.yml
    - --state-name
    - pilot_lower_left_workspace
  process_cmd: start
  publish_console: 0
  keep_alive_ms: 100
```

**NOTE**: If editing this file by hand, take care when copying and pasting. Undefined behavior will result if duplicate 'uuid' values are contained in the configuration file. If in doubt, delete this field and the process server will provide a new one.

Several of these fields may be ignored during hand-edits. The required fields are described below:

"process_name" is the application name, as it would be invoked in an interactive terminal, the client should start.

"client_name" is the name of the client who should execute the process of "process_name".

Useful but optional fields:

"arguments" is a list of arguments that should be provided to "process_name" when it is invoked. **NOTE:** There should be no spaces within a single argument.

"process_cmd" should be one of 'start', 'stop', or 'keep_alive'. 'start' will start the program, but not restart it if it crashes. 'stop' stops the process if it is running. 'keep_alive' starts the process and will restart it after 'keep_alive_ms' milliseconds have passed if it crashes. If omitted, this field defaults to 'start'.

"publish_console": if set to '1' the standard output and error of the process will be published and be available for viewing in the operator workspace. Care should be taken with the value as a chatty process can consume considerable bandwidth.

"keep_alive_ms" is the number of milliseconds to wait before restarting a crashed process. This can also be used to periodically run a process that runs to completion then exits at fixed intervals.

User edits of "time_unix_sec", "count_publish", or "sender_id" fields are always ignored. The minimal equivalent configuration entry for the example process above would be:

```
- client_name: padauk
  process_name: balefire_workspace
```

arguments:

- --cfg-file
- /home/ventana/gss_configs/balefire_mbari_config_007.yml
- --state-name
- pilot_lower_left_workspace

# 5 Vehicle Applications

The vehicle applications encapsulate the vehicle hardware of the *Ventana* control system. They provide low-level device status and control, high-level navigation estimates, and single degree-of-freedom autopilots as well as Station Keeping. Vehicle applications are broadly divided into vehicle navigation and vehicle control.

## 5.1 Vehicle Navigation: openINS

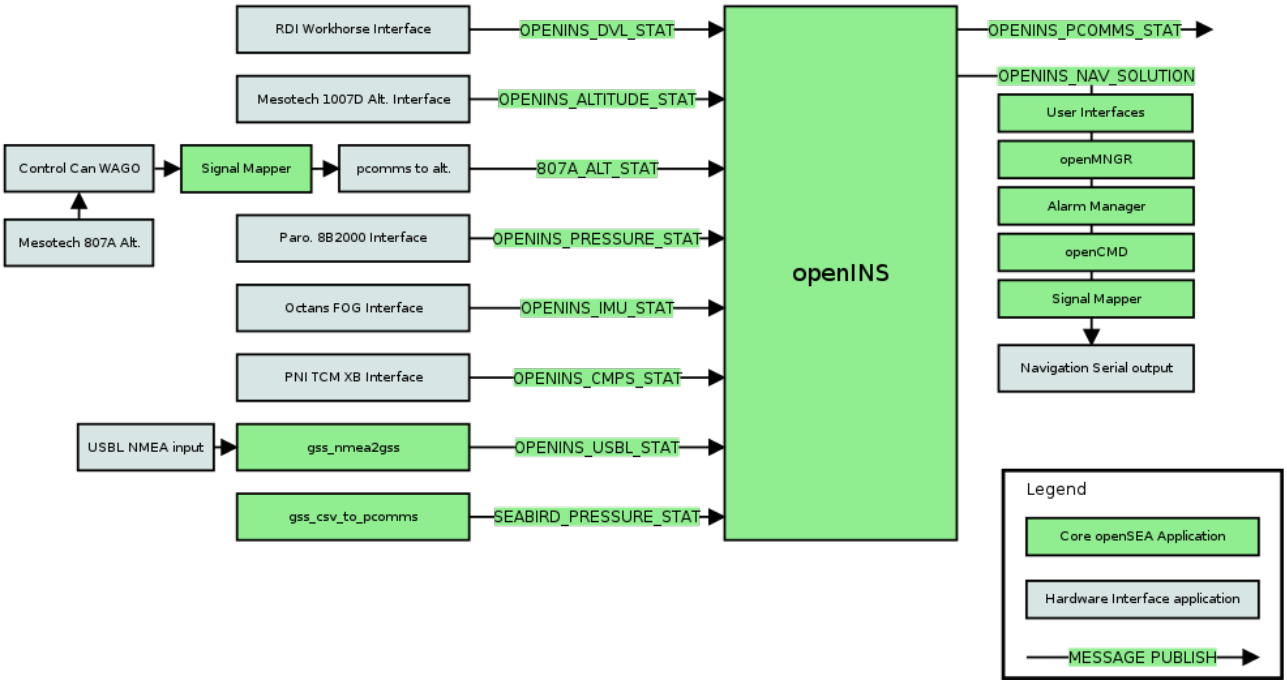


*Illustration 5: Navigation diagram.*

The vehicle navigation group is responsible for providing vehicle position and attitude estimates to the operator as well as to the vehicle control group. Each navigation sensor has a dedicated application that forwards the sensor information to the openINS application — the core of the vehicle navigation group. The individual sensor measurements are then fused by openINS to provide optimal estimates of vehicle position and attitude for the rest of the *Ventana* system.

## 5.2 Vehicle Control: openCMD and openMNGR

The vehicle control group is responsible for providing both open and closed-loop control of *Ventana*'s thrusters. Open-loop operator commands are collected from the console, transformed to percent deflection, and then forwarded to openMNGR. Closed-loop mode and set point commands can originate from the operator workspace, touch screen, or other control, but will be communicated over the OPENMNGR_CMD channel.

openMNGR is responsible for forwarding operator intentions to openCMD where the closed-loop controllers are implemented. In open-loop mode, openCMD will simply forward the joystick deflection percent as percent effort for a given degree of freedom. In closed-loop, openCMD will calculate thruster efforts based on the navigation solution, autopilot set points and other information. In both cases, the ventana_control application will convert percent efforts for all degrees of freedom into percent

effort for each of *Ventana*'s thrusters. This effort is then scaled to appropriate DAQ commands using Signal Mapper before being forwarded to a WAGO as an analog output.

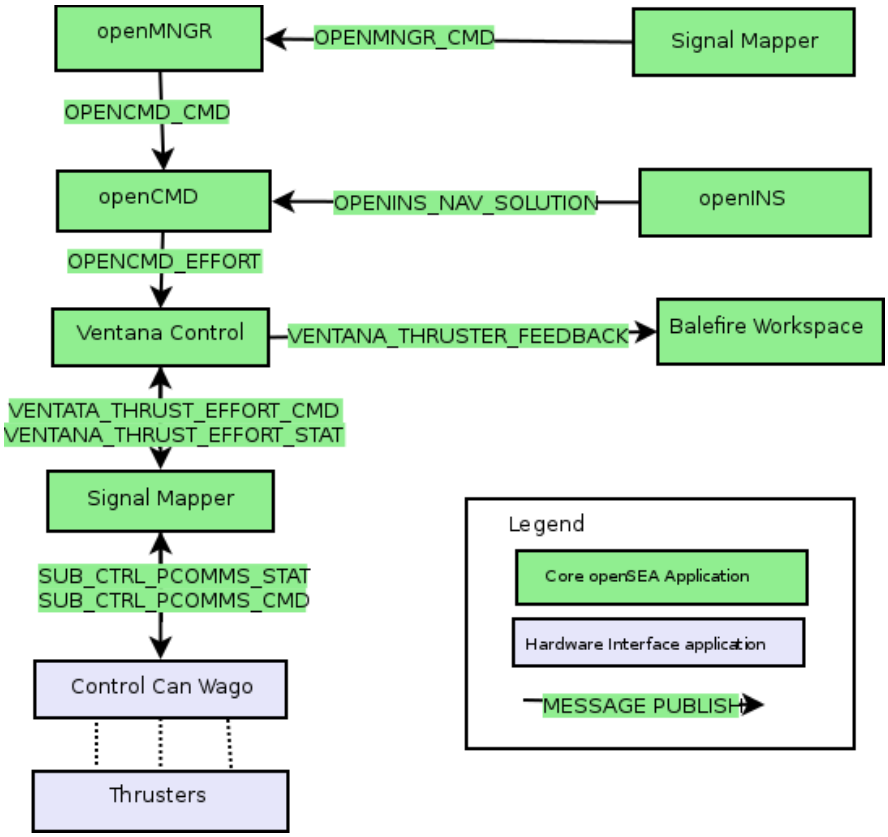


*Illustration 6: Ventana control diagram.*

### 5.3 Specific Architecture Elements

This section provides processing and dataflow details for elements within the *Ventana* control system.

#### 5.3.1 Serial Device Flow (TODO)

The general flow for a serial device is described in the following diagram:

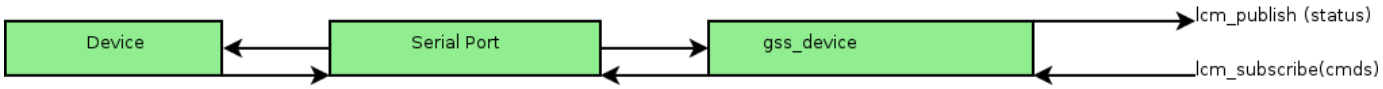


*Illustration 7: General serial device flow.*

All serial devices in the *Ventana* system have a similar architecture to the general device flow illustrated above. The devices communicate in a half or full duplex structure. Full duplex has a listener that subscribes to an LCM command channel and publishes data on an LCM status channel specific to that device.

### 5.3.2 Main Camera Pan/Tilt/Shoulder

The main camera pan/tilt shoulder commands originate from potentiometers on the pilot station. Voltages across the potentiometers are captured by the pilot station WAGO and read by an instance of the gss_wago application. The values read by the WAGO are published on the channel PSI_PCOMMS_STAT and then converted into commanded positions, in degrees, by Signal Mapper. The commanded positions in degrees are then published on channel PSI_PCOMMS_UI_STAT. The commands in degrees are subscribed to by the workspace and by the gss_ventana_pan_tilt application. The workspace displays the commanded pan and tilt positions on a two-axis plot, and displays the commanded shoulder position on a bar graph. The actual positions of the pan, tilt, and shoulder actuators is read from Hall effect sensors by the Control Can WAGO. Another instance of gss_wago reads the voltages from the WAGO and publishes the voltages from the Hall effect sensors on channel SUB_CTRL_PCOMMS_STAT. Signal Mapper converts these signals into degrees, and then publishes the degrees on channel SUB_CTRL_PCOMMS_UI_STAT.

The workspace and the gss_ventana_pan_tilt application both subscribe to the actual positions of the axes in degrees. The workspace displays the positions of the pan and tilt axes on the same two-axis plot as the commanded positions, and displays the actual shoulder position on a bar graph next to the one displaying the commanded shoulder position. The gss_ventana_pan_tilt application uses the commanded positions and the feedback from the actual positions to calculate commands that should be sent to the actuators. These commands are published on channel SUB_CTRL_PCOMMS_CMD. The subsea Control Can gss_wago instance subscribes to these commands and writes the commanded values to the subsea Control Can WAGO.

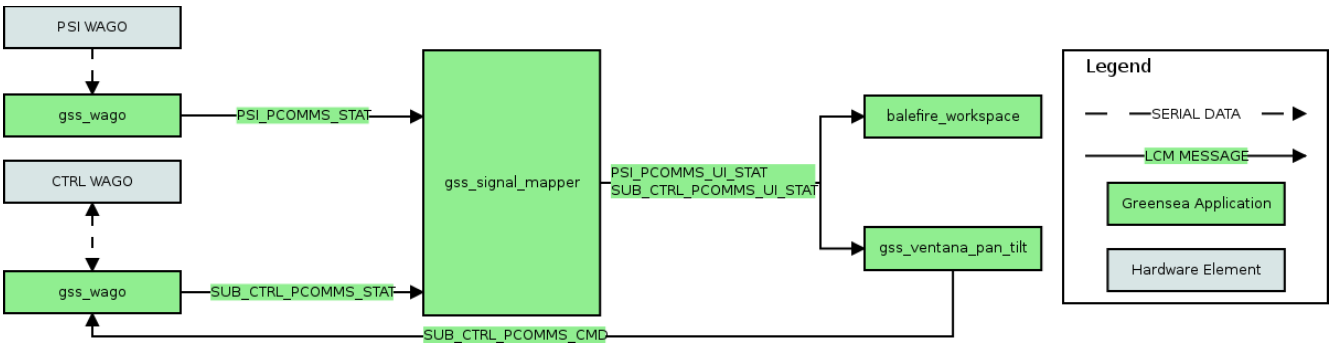


*Illustration 8: Main camera pan/tilt/shoulder control data flow.*

### 5.3.3 Variable Buoyancy System

The Variable Buoyancy (VB) system allows the user to adjust buoyancy manually through a widget controlling a valve and pump. Auto or Manual modes originate from a user input in the Balefire Workspace. A button press will select a mode. Manual mode allows the operator to press flood or pump to alter the water level. Auto mode allows the operator to select the level of fill with the slider and the system will then fill to that level. When a slider is moved, or a mode button is pressed, the status command signal is published published on an LCM channel, subscribed to by Signal Mapper. The input channel will vary depending on the VB command. Signal Mapper transforms the input signal to the designated profile and then passes the signal out through the output channel as a modified signal. The workspace and the Control Can WAGO-subscribe to the published signals to actuate buoyancy and display the changes. The workspace subscribes to SUB_CTRL_PCOMMS_UI_STAT to show the  set and fill status. The WAGO subscribes to SUB_CTRL_PCOMMS_STAT polling the pump and gss_fo_micro subscribes to FO_MODBUS_PCOMMS_CMD polling the flood.

VB profiles are used to define the properties of the VB settings established in Signal Mapper. The profiles are a group of transforms set together to describe the desired VB levels. The first table shows the profile for VB_AUTO, and the second for VB_MANUAL. A complete list of the configurations for the VB profiles as well as other *Ventana* maps can be found in Signal Mapper's configuration file.

A table of profile configurations for VB is located in the appendix of this document.

### 5.3.4 Hydraulic Tools and Detritus Samplers

Commands for the hydraulic tools and detritus samplers originate from a rocker switch and the trigger on the joystick, respectively. The signals from the joystick are captured by the PSI WAGO and read by the PSI WAGO's instance of the gss_wago application. The application publishes the signals on the channel PSI_PCOMMS_STAT, and the signals are consumed by Signal mapper. The user can select which hydraulic tool or detritus sampler to actuate via buttons in the workspace. These buttons publish commands on the BALEFIRE_SIGNAL_MAP_CMD channel. These commands select different signal mapping profiles. The different profiles contain mappings that route the joystick trigger or rocker button signals to different detritus samplers or hydraulic tools depending on which profile is selected. The commands for the detritus samplers and hydraulic tools are published on the SUB_CTRL_PCOMMS_CMD channel. Another instance of the gss_wago application subscribes to this channel, and in turn writes the commands received to the subsea Control Can WAGO.

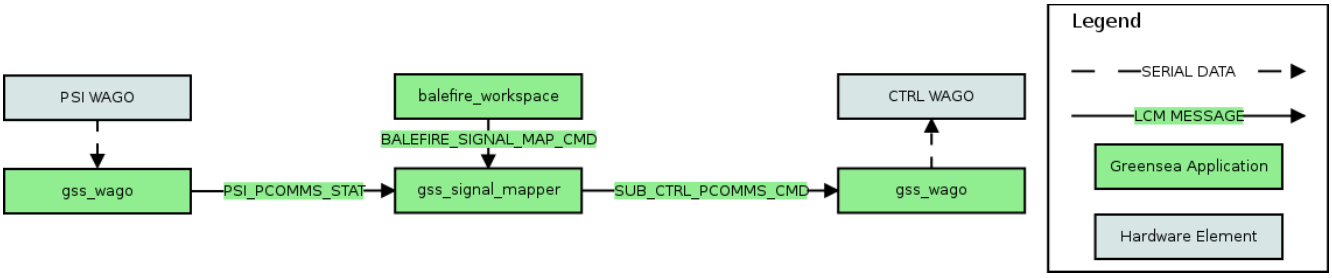


*Illustration 9: Hydraulic tools data flow.*

The hydraulic tools are profiled with a number, 1-9, and the detrirus sampler is numbered 1-9 in Signal Mapper. The tables in the appendix shows the profile option configuration for each tool.

### 5.3.5 Lights

An event on the workspace (button press) is published by balefire_workspace when the user presses a light control button. The balefire_workspace events are published to Signal Mapper, which publishes to the gss_wago Core Can.

A full table of light signal profiles are listed in the appendix of this document.

### 5.3.6 Video Overlay

The Video Overlay is implemented as an instance of balefire_workspace with a custom configuration. The appearance and content of the overlay can be modified by editing the configuration file in gss_configs/balefire_workspace/mbari_hud.yml.

### 5.3.7 FO Video Mux

Commands for the video mux originate from the workspace. They are transmitted on channel VIDEO_MUX_CMD to the gss_mbari_video_mux application. The gss_mbari_video_mux application generates a serial message to implement the command, then transmits this serial command via LCM to the gss_fo_micro application on channel VIDEO_MUX_SERIAL_RX. The gss_fo_micro then transmits this serial command using the serial over Modbus protocol. The gss_fo_micro application uses the same protocol to read the stat of the video mux. The gss_fo_micro application transmits the data it reads from the video mux to the gss_mbari_video_mux application via LCM on channel VIDEO_MUX_SERIAL_TX. The gss_mbari_video_mux application publishes the state of the video mux to the workspace on channel VIDEO_MUX_STAT.

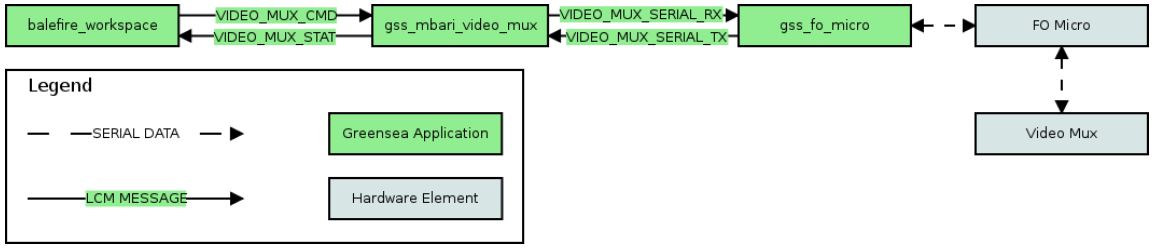


*Illustration 11: FO VideoMux data flow diagram.*

### 5.3.8 Insite Camera Control

Control signals for the Insite cameras originate from both the pilot station and Balefire Workspace. The pilot station provides controls for camera select, zoom, and focus. These are acquired by the PSI WAGO, read by an instance of gss_wago, then published on the PSI_PCOMMS_STAT channel. The camera select, zoom, and focus commands from the PSI are routed through Signal Mapper, then published to the gss_pegasus application on channel PEGASUS_CMD. The Balefire Workspace provides controls for zoom, focus, and white balance, as well as luminance, chrominance and high-frequency gain. These commands are transmitted directly from the workspace to the gss_pegasus application on the PEGASUS_CMD channel. The workspace subscribes to status information that is published by the gss_pegasus application on channel PEGASUS_STAT. The gss_pegasus application sends the commands it receives to the last selected Insite camera via a serial connection.

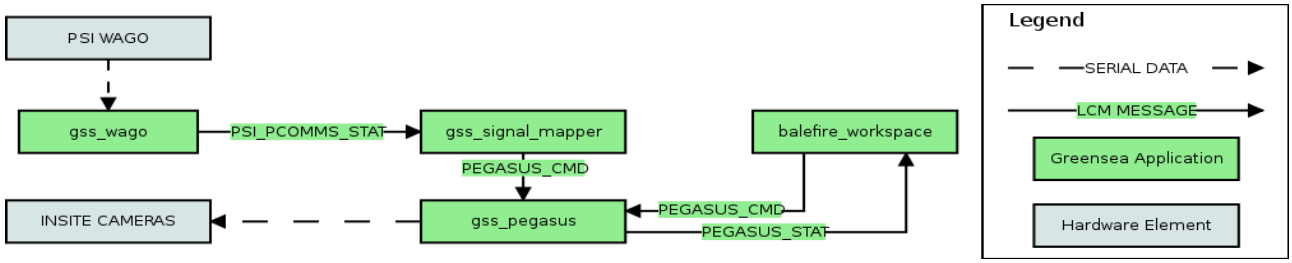


*Illustration 12: Data flow for the Insite camera control.*

### 5.3.9 Belly Pack

The Belly Pack (BP) device includes control buttons and switches as well as an LCD touchscreen. The BP interfaces with the Ventana Control System utilizing the serial MODbus. The signals represent commands for  Port, Starboard, lateral, vertical, Auto Heading, Bypass, vehicle kill and 14 spares. The spares are not currently in use but are designated, should a need arise.

The BP is enabled by a button on the pilot chair for launch and recovery purposes. The enable event is received by the BP WAGO instance of gss_wago. The WAGO publishes the control and status signals from the BP on BP_PCOMMS_STAT lcm channel.

When BP is enabled, the Pilot station is disabled. When BP kill is actuated, the signal c_bellypack_kill is inverted and mapped to c_bellypack_kill output on the CSI_PCOMMS_CMD channel to the gss_wago Control Can  instance to handle the kill command. The kill command is implemented by shutting off the vehicle hydraulics.

The control signals are subscribed to by Signal Mapper, transformed, and published as new signals. The port, starboard, lateral and vertical signals are transformed linearly. Auto heading is a pass through signal, not modified in Signal Mapper. BP kill is inverted in Signal Mapper and sent out as transformed signal. The BP status signals published are noted below.

| *Function* | *STAT Signal* |
|---|---|
| Port Command | s_port_cmd |
| Starboard Command | s_stbd_cmd |
| Lateral Command | s_lateral_cmd |
| Vertical Command | a_vertical_cmd |
| Auto Heading | s_auto_hdg |
| Vehicle Kill | s_vehicle_kill |
| Spare DI*n* | s_spared*i*n_bp * where *n* is the spare numbered 1-14 (allocated but not used) |

*Table 2: BP status signals.*

The BP LCD touchscreen subscribes to various LCM channels and writes them to the screen. The input and output signals, channels, and the transforms for the the BP commands and thrusters are outlined in the table located in the appendix of this document.

### 5.3.10 Hydraulic Bypass Interlock

The hydraulic bypass interlock is a hydraulic interlock that lets the pump dump off pressure on thruster start up to allow the thruster to run by actuating a valve. The valve is controlled by a button located on the pilot controls. The workspace displays status, however the control is through the pilot actuating the button only. The gss_wago instance of the Control Can publishes signal SUB_CTRL_PCOMMS_STAT with the signal s_confirmation_pump_bypass and Signal Mapper passes the signal on CSI_PCOMMS_CMD channel with a signal name of c_bypass_interlock. The CSI_PCOMMS_CMD channel is subscribed to by the thruster waiting to start.

### 5.3.11 Low Oil Bypass Interlock

A pressure sensor monitors oil pressure, publishing analog signal s_hp_oil_press on SUB_CTRL_PCOMMS_STAT. Signal Mapper transforms the signal from DAQ to STAT and publishes the us_hp_oil_press on SUB_CTRL_PCOMMS_UI_STAT which is displayed by the workspace. In the event that the oil pressure drops to trigger a low oil warning , the thruster shut off. The bypass is controlled by the operator, by pressing the low oil interlock push button. This enables the thrusters to run with low oil.

### 5.3.12 Groundfaults topside and subsea

The ground faults signals are output on the CSI_PCOMMS_CMD channel to the Control Can WAGO by the gss_mux_controller interface.

The ground faults are monitored by 16 sense lines that are polled by a bit select to the gss_wago Control Can instance, on the Control Can GF_Analog signals. The bit select ranges from 0B0000 for the Hotel 120 ground fault to 0B1111 for the Core Cam + GF. See the following table for a full list. The sense lines are polled for resistance below return and chassis ground. The resistance of 50 to 600 Ohms is scaled to a percentage of 0-100. There are two alarms on each line that register as either warning or failure. The workspace displays the percentage of the polls in a bar graph. The two modes available are 'Autocycle', which polls through all of the inputs, and 'Select', which polls one specific channel.

| *GFI Select Bit Number* | *Corresponding GFI* |
|---|---|
| 0B0000 | HOTEL 120 |
| 0B0001 | CORE 120 |
| 0B0010 | TOOLSLED 120 |
| 0B0100 | LIGHTS 120 |
| 0B0101 | CTR INST +5 |
| 0B0110 | CTR INST +12 |
| 0B0111 | CTR INST +48 |
| 0B1000 | CTR VALVES +24 |
| 0B1001 | Spare |
| 0B1010 | FO ISO +24 |
| 0B1011 | FO ISO +12 |
| 0B1100 | FO HD +24 |

| *GFI Select Bit Number* | *Corresponding GFI* |
|---|---|
| 0B1101 | FO ATLAS +24 |
| 0B1110 | CORE +24 |
| 0B1111 | CORE CAM+ |

*Table 3: GFI Bit Select.*

The Ground Fault multiplexors are mapped through Signal Mapper and have profiles assigned to them. A complete table of GFI_MUX profile configurations is listed in the appendix of this document.

### 5.3.13 Upper Camera Pan and Tilt

The commands for the upper camera pan and tilt can come from either the CSI foot pedals or the PSI joysticks. The signals from the foot pedals are captured by the CSI WAGO and read by an instance of the gss_wago application which publishes them on channel CSI_PCOMMS_STAT. The signals from the pilot station are captured by the PSI WAGO and read by a second instance of the gss_wago application which publishes on channel PSI_PCOMMS_STAT. The signals from the PSI and the CSI are consumed by Signal Mapper. Signal Mapper takes the logical OR of the pan and tilt commands from the CSI and the analogous commands from the PSI. The results of this operation are published on channel SUB_CTRL_PCOMMS_CMD. A third instance of gss_wago subscribes to SUB_CTRL_PCOMMS_CMD and writes the commands to the subsea control WAGO.

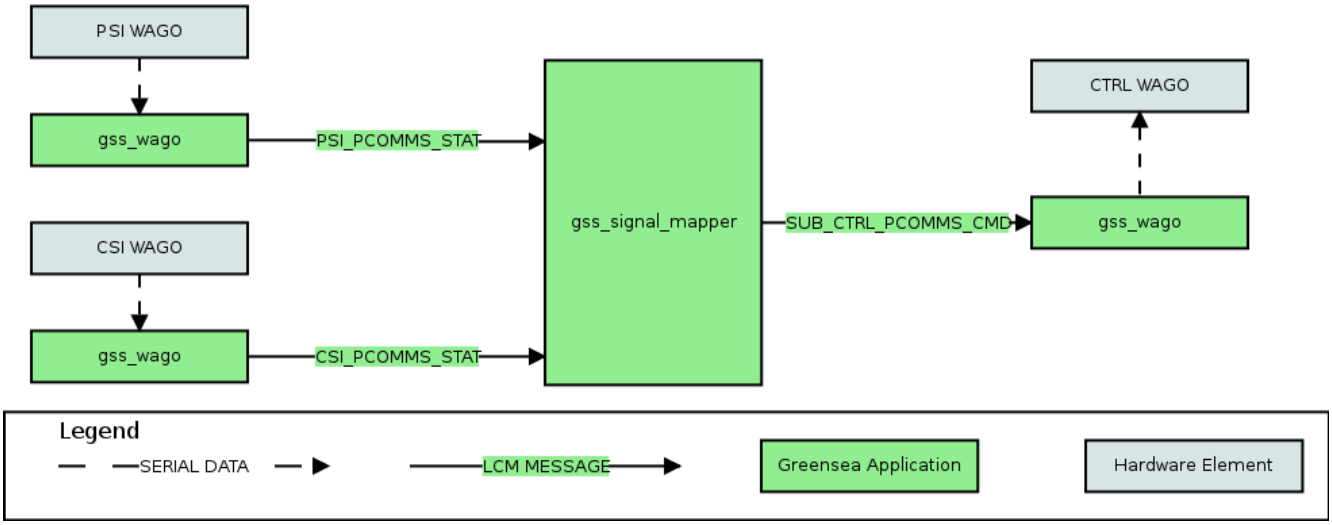


*Illustration 13: Upper camera pan/tilt data flow.*

Signal Mapper options configured for MAIN_CAM_PT_CMD profile are shown in the table located in the appendix of this document.

## 6 Application Descriptions by Computing Node

The following is a complete list of the applications that together comprise the software component of the *Ventana* control system. The responsibility of each application is described in addition to the computing node where it is intended to be executed.

| Application | Description | Responsibility |
|---|---|---|
| balefire_workspace | Pilot Lower-Left Touchscreen and Lower-Right Touchscreen | The primary application for user interaction, displaying current vehicle device and sensor data, navigation solution, and providing a means to control vehicle functions. Alarms and signal mapping are configured from the user interface. The primary means of interacting with the Workspace will be via touchscreen controls. The Pilot Lower-Left Touchscreen and Pilot Lower-Right Touchscreen applications differ only in the configuration file. |
| gss_process_client | openPROCESS client | The process client starts and restarts the operator interfaces. |

*Table 4: Table of Pilot (padauk) computer applications.*

| Application | Description | Responsibility |
|---|---|---|
| balefire_workspace | Pilot Upper-Left Touchscreen and Pilot Lower-Right Touchscreen | The primary application for user interaction, displaying current vehicle device and sensor data, navigation solution, and providing a means to control vehicle functions. Alarms and signal mapping are configured from the user interface. The primary means of interacting with the Workspace will be via touchscreen controls. The Pilot Upper-Left Touchscreen and Pilot Upper-Right Touchscreen applications differ only in the configuration file. |
| gss_process_client | openPROCESS client | The process client starts and restarts the operator interfaces. |

*Table 5: Table of Copilot (sapele) computer applications.*

| Application | Description | Responsibility |
|---|---|---|
| gss_wago | Pilot WAGO | This applications translates between the serial protocol from the WAGO data acquisition device and the Greensea publish-subscribe messaging format. The primary recipient is Signal Mapper. |
| gss_wago | Control Can WAGO | This applications translates between the serial protocol from the WAGO data acquisition device and the Greensea publish-subscribe messaging format. The primary recipient is Signal Mapper. |
| gss_wago | Belly Pack WAGO | This applications translates between the serial protocol from the WAGO data acquisition device and the Greensea publish-subscribe messaging format. The primary recipient is Signal Mapper. |
| gss_nmea2gss | Ship's Position Interface | This applications translates between the serial NMEA position sentences received from the MOXA and makes the position available to the user interface over the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | Ship's Gyro Interface | This applications translates between the serial NMEA position sentences received from the MOXA and makes the attitude available to the user interface over the Greensea publish-subscribe messaging format. |
| gss_pegasus | Pegasus Interface | This applications translates between the Greensea publish-subscribe messaging format and Insite Pacific Pegasus Camera. |
| gss_winch_lci90i | Winch Interface | This application makes the serial stream from the Measurement Technology Northwest's LCI-90i available in the Greensea publish-subscribe messaging format. |
| start_ventana_csv.sh | Navigation Output | This application takes user-defined navigation data, converts it to CSV format and sends it out a serial interface to the MBARI data logging application. |
| balefire_workspace | Overlay Generator | The output of this specially configured instance of the Balefire Workspace will be used as the input to the Miranda DVI-Ramp to create an overlay on the main video stream. |

| Application | Description | Responsibility |
|---|---|---|
| gss_sc2004_mbs | 4-line LCD | Provides an interface to write to the 4-line LCD on the bellypack. |
| gss_mux_controller | GFI Mux application | Maintains the state of each of the GFI monitored channels, cycles through the channels using bit select signals, and implements operator controls to cycle or stare at a particular GFI input. |
| gss_process_client | openPROCESS client | The process client starts and restarts the topside applications. |
| gss_alarm_manager | Alarm Manager | This application monitors all signals on the network and allows logical tests to be performed against them to trigger an alarm. |

*Table 6: Table of Topside (tigrillo) computer applications.*

| Application | Description | Responsibility |
|---|---|---|
| openmngr | openMNGR | This application takes high-level listens on the navigation solution and high-level commands such as waypoints and provides a desired destination for openCMD to command the vehicle to. |
| openins | openINS | This application takes inputs from all ROV navigation sensors and uses sensor fusion techniques to provide a best estimate of current navigation state including position, velocity, and attitude. |
| opencmd | openCMD | This application takes a commanded position from openMNGR and calculates the required effort in six degrees of freedom to reach that destination. |
| gss_wago | Control System WAGO | This applications translates between the serial protocol from the WAGO data acquisition device and the Greensea publish-subscribe messaging format. The primary recipient is Signal Mapper. |
| gss_wago | Core Can WAGO | This applications translates between the serial protocol from the WAGO data acquisition device and the Greensea publish-subscribe messaging format. The primary recipient is Signal Mapper. |
| gss_fo_micro | FOModbus Serial Interface | This applications translates between the FOModbus protocol and the Greensea publish-subscribe messaging format. |
| lcm_dvl | RDI Workhorse Interface | This application receives the PD5 serial data input from the DVL and converts the information to LCM format for openINS and other applications. |
| lcm_phins6000 | Octans FOG Interface | This application makes the Octans serial protocol available in the Greensea publish-subscribe messaging format. |
| lcm_parosci | Paro. 8B2000 Interface | This application makes the Paroscientific pressure sensor serial protocol available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | Mesotech 1007D Altimeter Interface | This application makes the Mesotech 10007D NMEA output available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | USBL Interface | This application makes the USBL NMEA output available in the Greensea publish-subscribe messaging format. |
| lcm_tcmxb | PNI TCM XB Interface | This application makes the PNI TCM XB serial protocol available in the Greensea publish-subscribe messaging format. |
| gss_signal_mapper | Signal Mapper | This application allows mapping and transforming of an input signal to an output signal. This is used to assign functionality to physical buttons and knobs on a control console and soft buttons on the user interface |

| Application | Description | Responsibility |
|---|---|---|
| | | through to outputs on the vehicle data acquisition and control modules. |
| ventana_control | Ventana Control | This application converts percent efforts for each degree of freedom received from openCMD into percent effort for each of *Ventana*'s thrusters. |
| gss_mbari_video_mux | Video MUX | This application translates between the Greensea publish-subscribe messaging format and the MBARI video mux protocol. |
| gss_pcomms_to_altitude_stat.py | Analog Altimeter | Converts the analog altimeter signal into a format understood by openINS |
| start_gss_csv_to_pcomms.sh | Seabird CTD | Shell script that executes generic 'gss_csv_to_pcomms' application. In this case, we parse Seabird CTD comma-separated values into formats understood by Signal Mapper nd openINS. |
| start_ventana_csv.sh | MBARI data logging | Shell script that invokes 'gss_ventana_csv' application while redirecting its output to a serial port. |
| gss_mux_controller | GFI Mux application | Maintains the state of each of the GFI monitored channels, cycles through the channels using bit select signals, and implements operator controls to cycle or stare at a particular GFI input. |
| gss_turns_count | Tether turns | Application that monitors vehicle and topside heading to calculate running tether turns count. |
| gss_process_client | openPROCESS client | The process client starts and restarts the topside applications. |
| gss_process_server | openPROCESS server | The process server issues commands to process clients. |

*Table 7: Table of control computer (cocobolo) applications.*

## 7 Table of Applications by Published Channel and Computing Node

The following is a complete list of channels published by each application organized by computing node. These tables, taken together, should enable the operator to associate specific instances of a given application with the data they produce.

**NOTE:** The channels listed below are those delivered by Greensea. Nearly all of the applications take the channel they publish as a command-line argument or by a configuration file, so this list may become inaccurate following user edits.

| Application | Publish Channel | Purpose |
|---|---|---|
| balefire_workspace | OPENMNGR_CMD | High-level autopilot commands originating from the operator workspace. |
| balefire_workspace | BALEFIRE_SIGNAL_MAP_CMD | Instruct Signal Mapper to edit or delete signal maps, save or load a saved configuration file. |
| balefire_workspace | BALEFIRE_SIGNAL_MAP_ADD_CMD | Instruct the Signal Mapper to add a signal map. |
| balefire_workspace | OPENDEVICE_CMD | Commands from the operator workspace to configure the openDEVICE service. |
| balefire_workspace | BALEFIRE_ALARM_CMD | Instruct Alarm Manager to edit or delete an alarm. |
| balefire_workspace | BALEFIRE_ALARM_ADD_CMD | Instruct Alarm Manager to edit or delete a signal map, save or load a saved configuration file. |
| balefire_workspace | OPENINS_CMD | The operator workspace sends commands to openINS, such as specify current vehicle position (latitude and longitude) and magnetic declination. |

| Application | Publish Channel | Purpose |
|---|---|---|
| balefire_workspace | PEGASUS_CMD | Commands from the operator workspace to configure the Pegasus Insite Pacific camera. |
| balefire_workspace | PROCESS_SERVER_CMD | Instruct the process server to start or stop a process, publish, or stop publishing its console output. |
| gss_process_client | COPILOT_PROCESS_CLIENT_STAT | Current status of all applications monitored by the copilot process client. |
| gss_process_client | PILOT_PROCESS_CLIENT_STAT | Current status of all applications monitored by the pilot process client. |
| gss_process_client | PROCESS_CLIENT_CONSOLE_STAT | Shared channel for publishing the console output of a monitored process. |
| balefire_workspace | TURNS_COUNT_CMD | Turns counter to zero and reset compromised messages. |

*Table 8: List of channels published by pilot and copilot computer applications and their purpose.*

| Application | Publish Channel | Purpose |
|---|---|---|
| gss_wago | PSI_PCOMMS_STAT | Polled status from the Pilot Station WAGO Stack made available in the Greensea publish-subscribe messaging format. |
| gss_wago | BP_PCOMMS_STAT | Polled status from the Belly Pack WAGO Stack made available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | TS_SHIPS_POS_STAT | The NMEA ship's position stream made available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | TS_SHIPS_HDG_STAT | The NMEA ship's attitude stream made available in the Greensea publish-subscribe messaging format. |
| gss_pegasus | PEGASUS_STAT | Polled status from the Insite Pacific Pegasus camera made available in the Greensea publish-subscribe messaging format. |
| gss_winch_lci90i | WINCH_STAT | Polled status from the LCI-90i line control instrument made available in the Greensea publish-subscribe messaging format. |
| gss_process_client | TOPSIDE_PROCESS_CLIENT_STAT | Current status of all applications monitored by the topside process client |
| gss_process_client | PROCESS_CLIENT_CONSOLE_STAT | Shared channel for publishing the console output of a monitored process |
| gss_alarm_manager | BALEFIRE_ALARM_STAT | Periodic publication of the complete list of all defined alarms and their status. |
| gss_alarm_manager | BALEFIRE_ALARM_EVENT_STAT | Immediate publication of a single alarm when it changes state. Used for low-latency notification of an alarm condition. |
| gss_alarm_manager | BALEFIRE_ALARM_CMD_RESPONSE | Alarm manager responses to the remote commands. |
| gss_mux_controller | CSI_PCOMMS_CMD | Outputs GFI bit-select signals to the control system interface WAGO |

*Table 9: List of channels published by the topside computer applications and their purpose.*

| *Application* | *Publish Channel* | *Purpose* |
|---|---|---|
| openmngr | OPENMNGR_PCOMMS_STAT | The internal state of the openMNGR application. |
| openmngr | OPENCMD_CMD | Autopilot commands from openMNGR to openCMD. |
| opencmd | OPENCMD_EFFORT | Desired thruster efforts in percent for each degree of freedom. |
| opencmd | OPENCMD_PCOMMS_STAT | The internal state of the openCMD application. |
| openins | OPENINS_PCOMMS_STAT | The internal state of the openINS application. |
| openins | OPENINS_NAV_SOLUTION | The vehicle navigation solution. This information is widely subscribed throughout the system as the authoritative source of vehicle position and attitude information. |
| gss_signal_mapper | NAV_OUT_STAT | Signal source for the configurable serial navigation output for use by MBARI. |
| gss_signal_mapper | BALEFIRE_MAP_STAT | Configurable status output. Primarily intended to aid advanced control and transformation options. |
| gss_signal_mapper | BALEFIRE_SIGNAL_MAP_STAT | A complete listing of all configured signal maps as well a listing of the current transform capabilities of Signal Mapper. |
| gss_signal_mapper | BALEFIRE_SIGNAL_MAP_CMD_RESPONSE | Responses to remote commands. |
| gss_signal_mapper | SUB_CTRL_PCOMMS_CMD | Accumulated output commands to the Control Can WAGO Stack. |
| gss_signal_mapper | CORE_CAN_PCOMMS_CMD | Accumulated output commands to the Core Can WAGO Stack. |
| gss_signal_mapper | CSI_PCOMMS_CMD | Accumulated output commands to the Control System Can WAGO Stack. |
| gss_signal_mapper | FO_MODBUS_PCOMMS_CMD | Accumulated output commands to the FOModbus serial interface. |
| gss_wago | CSI_PCOMMS_STAT | Polled status from the Control System WAGO Stack made available in the Greensea publish-subscribe messaging format. |
| gss_wago | CORE_CAN_PCOMMS_STAT | Polled status from the Core Can WAGO Stack made available in the Greensea publish-subscribe messaging format. |
| gss_wago | SUB_CTRL_PCOMMS_STAT | Polled status from the Control Can WAGO Stack made available in the Greensea publish-subscribe messaging format. |
| gss_fo_micro | FO_MODBUS_PCOMMS_STAT | Polled status from the FOModbus serial interface made available in the Greensea publish-subscribe messaging format. |
| gss_fo_micro | VIDEO_MUX_SERIAL_TX | Raw serial bytes for pass-through by the gss_mbari_video_mux application. |
| lcm_dvl | OPENINS_DVL_STAT | Polled status from the RDI Workhorse Navigator DVL made available in the Greensea |

| *Application* | *Publish Channel* | *Purpose* |
|---|---|---|
| | | publish-subscribe messaging format. |
| lcm_phins6000 | OPENINS_IMU_STAT | Polled status from the IXSEA Octans Fiber-optic Gyro made available in the Greensea publish-subscribe messaging format. |
| lcm_parosci | OPENINS_PRESSURE_STAT | Polled status from the Paroscientific 8B2000 pressure sensor made available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | OPENINS_ALTITUDE_STAT | Polled status from the Mesotech 10007D NMEA output available in the Greensea publish-subscribe messaging format. |
| lcm_tcmxb | OPENINS_CMPS_STAT | Polled status from the PNI TCM XB output available in the Greensea publish-subscribe messaging format. |
| gss_nmea2gss | OPENINS_USBL_STAT | Polled status from the NMEA USBL output available in the Greensea publish-subscribe messaging format. |
| start_gss_csv_to_pcomms.sh | SEABIRD_PRESSURE_STAT | Pressure information from the Seabird CTD made available in the Greensea publish-subscribe messaging format. |
| start_gss_csv_to_pcomms.sh | SEABIRD_CTD_STAT | One signal per column in the Seabird CTD CSV output. |
| gss_process_client | CONTROL_PROCESS_CLIENT_STAT | Current status of all applications monitored by the control system process client. |
| gss_process_client | PROCESS_CLIENT_CONSOLE_STAT | Shared channel for publishing the console output of a monitored process. |
| gss_process_server | PROCESS_SERVER_STAT | Desired global process configurations for clients to execute. |
| gss_turns_count | TURNS_COUNT_STAT | Tether turns count value and status. |
| ventana_control | VENTANA_THRUST_EFFORT_CMD | Thruster efforts in percent. |
| ventana_control | VENTANA_THRUST_EFFORT_STAT | Wago output voltage converted to thruster percent then formatted as thruster_stat_t. |
| gss_mbari_video_mux | VIDEO_MUX_STAT | Current video mux status. |
| gss_mbari_video_mux | VIDEO_MUX_SERIAL_RX | Raw serial bytes for pass-through by the gss_fo_micro application. |
| gss_mux_controller | VEHICLE_GFI_MUX_STAT | Accumulated GFI values. |
| gss_mux_controller | SUB_CTRL_PCOMMS_CMD | GFI bit-select messages. |
| gss_pcomms_to_altitude_stat.py | 807A_ALT_STAT | Analog altimeter messages converted to a format openINS understands. |

*Table 10: List of channels published by the control computer applications and their purpose.*

# 8 Appendix

## 8.1 Variable Buoyancy Profile

| *Input Channel:Input signal* | *Output Channel:Signal* | *Transform* | *Description* |
|---|---|---|---|
| BALEFIRE_MAP_STAT:vb_auto_set_point | VB_AUTO:vb_auto_set_point_lower | BIAS(5) | VB auto lower deadband. |
| BALEFIRE_MAP_STAT:vb_auto_set_point | VB_AUTO:vb_auto_set_point_upper | BIAS(5) | VB auto upper deadband. |
| SUB_CTRL_PCOMMS_STAT:s_confirmation_vb_pump_op | SUB_CTRL_PCOMMS_UI_STAT:s_vb_manual_pump_stat | ALL(BALEFIRE_MAP_STAT:vb_auto_mode)\|GREATER(0)\|TRUE | Button on/off if pump is running (auto). |
| SUB_CTRL_PCOMMS_UI_STAT:us_vb_level | SUB_CTRL_PCOMMS_CMD:c_vb_pump_op | ALL(BALEFIRE_MAP_STAT:vb_auto_mode)\|GREATER(VB_AUTO:vb_auto_set_point_upper)\|LINEAR(0,8191) | SUB_CTRL_PCOMMS_UI_STAT:us_vb_level. |
| SUB_CTRL_PCOMMS_UI_STAT:us_vb_level | SUB_CTRL_PCOMMS_CMD:c_vb_pump_op | ALL(BALEFIRE_MAP_STAT:vb_auto_mode)\|LESSER(VB_AUTO:vb_auto_set_point_upper)\|GAIN(0) | Pump off if level is lower than upper set point. |
| SUB_CTRL_PCOMMS_UI_STAT:us_vb_level | SUB_CTRL_PCOMMS_CMD:c_vb_pump_op | ALL(BALEFIRE_MAP_STAT:vb_auto_mode)\|GREATER(VB_AUTO:vb_auto_set_point_upper)\|LINEAR(0,8191) | Pump on if level greater than upper set point. Turns on to 2.5v. |
| SUB_CTRL_PCOMMS_UI_STAT:us_vb_level | FO_MODBUS_PCOMMS_CMD:c_vbflood_onoff | ALL(BALEFIRE_MAP_STAT:vb_auto_mode)\|LESSER(VB_AUTO:vb_auto_set_point_lower)\|TRUE\|ON_CHANGE | Flood on if level is less than lower set point. |

*Table 11: Variable Buoyancy Auto profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| BALEFIRE_MAP_STAT:vb_manual_pump | SUB_CTRL_PCOMMS_UI_STAT:s_vb_manual_pump_stat | ALL(! BALEFIRE_MAP_STAT:vb_auto_mode) | Button on/off if pump is enabled (manual). |
| FO_MODBUS_PCOMMS_UI:u_vbflood_onoff | FO_MODBUS_PCOMMS_CMD:c_vbflood_onoff | ALL(! BALEFIRE_MAP_STAT:vb_auto_mode)\|ON_CHANGE | manual vb_flood_onoff. |
| PSI_PCOMMS_UI_STAT:us_vb_cmd | SUB_CTRL_PCOMMS_CMD:c_vb_pump_op | GAIN(3276.7)\|ALL(! BALEFIRE_MAP_STAT:vb_auto_mode)\|BOUNDS(-32767,32767) | Scale voltage from pot to DAQ units for WAGO. Disable signal unless manual pump button is enabled and auto mode is disabled. |

*Table 12: Variable Buoyancy Manual profile configurations.*

### 8.2 Hydraulic Tools Profile Configurations

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_port_drawer_opn | ON_CHANGE | CP Port Drawer Open |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_port_drawer_cls | ON_CHANGE | CP Port Drawer Close |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_port_drawer_opn | ON_CHANGE | Port Drawer Open |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_port_drawer_cls | ON_CHANGE | Port Drawer Close |

*Table 13: HYD_TOOL1 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_stbd_drawer_opn | ON_CHANGE | CP Stbd Drawer Open |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_stbd_drawer_cls | ON_CHANGE | CP Stbd Drawer Close |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_stbd_drawer_opn | ON_CHANGE | Stbd Drawer Open |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_stbd_drawer_cls | ON_CHANGE | Stbd Drawer Close |

*Table 14: HYD_TOOL2 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_port_swing_in | ON_CHANGE | CP Port Swing In |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_port_swing_out | ON_CHANGE | CP Port Swing Out |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_port_swing_in | ON_CHANGE | Port Swing In |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_port_swing_out | ON_CHANGE | Port Swing Out |

*Table 15: HYD_TOOL3 profile configuration.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_port_spare1_a | ON_CHANGE | Port Spare1 position A |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_port_spare1_b | ON_CHANGE | Port Spare1 position B |
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_port_spare1_a | ON_CHANGE | CP Port Spare1 position A |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_port_spare1_b | ON_CHANGE | CP Port Spare1 position B |

*Table 16: HYD_TOOL4 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_port_spare2_b | ON_CHANGE | CP Port Spare2 position B |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_port_spare2_b | ON_CHANGE | CP Port Spare2 position B |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_port_spare2_a | ON_CHANGE | Port Spare2 position A |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_port_spare2_b | ON_CHANGE | Port Spare2 position B |

*Table 17: HYD_TOOL5 profile configuration.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_stbd_spare1_a | IGNORE\|ON_CHANGE | CP Stbd Spare1 position A |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_stbd_spare1_a | ON_CHANGE | Stbd Spare1 position A |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_stbd_spare1_b | ON_CHANGE | Stbd Spare1 position B |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_stbd_spare2_b | ON_CHANGE | CP Stbd Spare1 position B |

*Table 18: HYD_TOOL6 profile configuration.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_rckr_dply | SUB_CTRL_PCOMMS_CMD:c_stbd_spare2_a | ON_CHANGE | CP Stbd Spare2 position A |
| CSI_PCOMMS_STAT:s_cp_rckr_ret | SUB_CTRL_PCOMMS_CMD:c_stbd_spare2_b | ON_CHANGE | CP Stbd Spare2 position B |
| PSI_PCOMMS_STAT:s_rocker_deploy | SUB_CTRL_PCOMMS_CMD:c_stbd_spare2_a | ON_CHANGE | Stbd Spare2 position A |
| PSI_PCOMMS_STAT:s_rocker_retract | SUB_CTRL_PCOMMS_CMD:c_stbd_spare2_b | ON_CHANGE | Stbd Spare2 position B |

*Table 19: HYD_TOOL7 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| PSI_PCOMMS_STAT:s_rocker_deploy | CSI_PCOMMS_CMD:c_tlsld_rckr_dply | ON_CHANGE | Midwater toolsled tool |
| PSI_PCOMMS_STAT:s_rocker_retract | CSI_PCOMMS_CMD:c_tlsld_rckr_ret | ON_CHANGE | PSI rocker to toolsled rocker dply |

*Table 20: HYD_TOOL8 profile configurations.*

### 8.2.1 Detritus Profile configurations

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det1_cls | ON_CHANGE | Detritus 1 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det1_opn | ON_CHANGE | Detritus 1 trigger Open |

*Table 21: DETRITUS1 profile configuration.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det2_cls | ON_CHANGE | Detritus 2 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det2_opn | ON_CHANGE | Detritus 2 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det2_cls | ON_CHANGE | Detritus 2 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det2_opn | ON_CHANGE | Detritus 2 trigger Open |

*Table 22: DETRITUS2 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det3_cls | ON_CHANGE | Detritus 3 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det3_opn | ON_CHANGE | Detritus 3 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det3_cls | ON_CHANGE | Detritus 3 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det3_opn | ON_CHANGE | Detritus 3 trigger Open |

*Table 23: DETRITUS3 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det4_cls | ON_CHANGE | Detritus 4 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det4_opn | ON_CHANGE | Detritus 4 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det4_cls | ON_CHANGE | Detritus 4 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det4_opn | ON_CHANGE | Detritus 4 trigger Open |

*Table 24: DETRITUS4 Profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det5_cls | ON_CHANGE | Detritus 5 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det5_opn | ON_CHANGE | Detritus 5 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det5_cls | ON_CHANGE | Detritus 5 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det5_opn | ON_CHANGE | Detritus 5 trigger Open |

*Table 25: DETRITUS5 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det6_cls | ON_CHANGE | Detritus 6 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det6_opn | ON_CHANGE | Detritus 6 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det6_cls | ON_CHANGE | Detritus 6 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det6_opn | ON_CHANGE | Detritus 6 trigger Open |

*Table 26: DETRITUS6 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det7_cls | ON_CHANGE | Detritus 7 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det7_opn | ON_CHANGE | Detritus 7 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det7_cls | ON_CHANGE | Detritus 7 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det7_opn | ON_CHANGE | Detritus 7 trigger Open |

*Table 27: DETRITUS7 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det8_cls | ON_CHANGE | Detritus 8 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det8_opn | ON_CHANGE | Detritus 8 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det8_cls | ON_CHANGE | Detritus 8 trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det8_opn | ON_CHANGE | Detritus 8 trigger Open |

*Table 28: DETRITUS8 profile configurations.*

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CSI_PCOMMS_STAT:s_cp_det_cls | SUB_CTRL_PCOMMS_CMD:c_det9_cls | ON_CHANGE | Detritus 9 CP Det Close |
| CSI_PCOMMS_STAT:s_cp_det_opn | SUB_CTRL_PCOMMS_CMD:c_det9_opn | ON_CHANGE | Detritus 9 CP Det Open |
| PSI_PCOMMS_STAT:s_trigger_close | SUB_CTRL_PCOMMS_CMD:c_det9_cls | ON_CHANGE | Detritus 9 (none) trigger Close |
| PSI_PCOMMS_STAT:s_trigger_open | SUB_CTRL_PCOMMS_CMD:c_det9_opn | ON_CHANGE | Detritus 9 trigger Open |

*Table 29: DETRITUS9 profile configurations.*

### 8.3 Belly Pack Profile Configurations

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| BP_PCOMMS_STAT:s_lateral_cmd | OPENMNGR_DAQ_CMD:U_DAQ_JOY_Y | LINEAR(0.001678,-24.4988)\|GAIN(20)\|INVERT\|BOUNDS(-100,100) | Belly Pack lateral command input |
| BP_PCOMMS_STAT:s_port_cmd | BP_PCOMMS_UI_STAT:us_port_cmd | LINEAR(0.001781,-25.9456)\|GAIN(20)\|BOUNDS(-100,100) | Belly Pack port thruster command input |
| BP_PCOMMS_STAT:s_stbd_cmd | BP_PCOMMS_UI_STAT:us_stbd_cmd | LINEAR(0.0017,-24.8)\|GAIN(20)\|BOUNDS(-100,100) | Belly Pack stbd thruster command input |
| BP_PCOMMS_STAT:s_vehicle_kill | CSI_PCOMMS_CMD:c_bellypack_kill | ON_CHANGE\|INVERT | Belly Pack emergency vehicle shutdown switch |
| BP_PCOMMS_STAT:s_vertical_cmd | OPENMNGR_DAQ_CMD:U_DAQ_JOY_Z | LINEAR(0.0003379,-5)\|GAIN(-20)\|BOUNDS(-100,100) | Belly Pack vertical thruster command |
| BP_PCOMMS_STAT:s_auto_hdg | OPENMNGR_DAQ_CMD:U_AUTOHDG_EN | ON_CHANGE | Belly Pack auto heading mode |

*Table 30: BELLY_PACK profile configurations.*

**Note:** Further Detritus mapping is located in the ventana_maps.yaml

### 8.4 Lights Profile Configurations

| *Input Channel:Signal* | *Output Channel:Signal* | *Transform* | *Description* |
|---|---|---|---|
| CORE_CAN_PCOMMS_UI:u_aft_inc_iso | CORE_CAN_PCOMMS_CMD:c_aft_inc_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_aft_inc_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_aft_inc_onoff | CORE_CAN_PCOMMS_CMD:c_aft_inc_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_aft_inc_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_aft_inc_onoff | CORE_CAN_PCOMMS_CMD:c_aft_inc_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_aft_inc_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_cam_inc_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_port_lhmi_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_port_thmi_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_port_uhmi_onoff<br>profile: LIGHT_RELAYS | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_stbd_lhmi_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_stbd_thmi_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_all_lights_off | CORE_CAN_PCOMMS_UI:u_stbd_uhmi_onoff | FALSE | Turn off all lights |
| CORE_CAN_PCOMMS_UI:u_cam_inc_iso | CORE_CAN_PCOMMS_CMD:c_cam_inc_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_cam_inc_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_cam_inc_onoff | CORE_CAN_PCOMMS_CMD:c_cam_inc_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_cam_inc_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_cam_inc_onoff | CORE_CAN_PCOMMS_CMD:c_cam_inc_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_port_lhmi_iso | CORE_CAN_PCOMMS_CMD:c_port_lhmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_lhmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_port_lhmi_onoff | CORE_CAN_PCOMMS_CMD:c_port_lhmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_lhmi_iso) | Close non-isolating relay Only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_port_lhmi_onoff | CORE_CAN_PCOMMS_CMD:c_port_lhmi_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| CORE_CAN_PCOMMS_UI:u_port_thmi_iso | CORE_CAN_PCOMMS_CMD:c_port_thmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_thmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_port_thmi_onoff | CORE_CAN_PCOMMS_CMD:c_port_thmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_thmi_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_port_thmi_onoff | CORE_CAN_PCOMMS_CMD:c_port_thmi_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_port_uhmi_iso | CORE_CAN_PCOMMS_CMD:c_port_uhmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_uhmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_port_uhmi_onoff | CORE_CAN_PCOMMS_CMD:c_port_uhmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_port_uhmi_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_stbd_lhmi_iso | CORE_CAN_PCOMMS_CMD:c_stbd_lhmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_lhmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_stbd_lhmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_lhmi_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_stbd_lhmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_lhmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_lhmi_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_stbd_thmi_iso | CORE_CAN_PCOMMS_CMD:c_stbd_thmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_thmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_stbd_thmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_thmi_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_stbd_thmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_thmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_thmi_iso) | Close non-isolating relay only if isolating is closed |
| CORE_CAN_PCOMMS_UI:u_stbd_uhmi_iso | CORE_CAN_PCOMMS_CMD:c_stbd_uhmi_iso | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_uhmi_onoff)\|INVERT | Open isolating relay only if non-isolating is open |
| CORE_CAN_PCOMMS_UI:u_stbd_uhmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_uhmi_onoff | EQUALS(FALSE) | Open non-isolating relay at any time |
| CORE_CAN_PCOMMS_UI:u_stbd_uhmi_onoff | CORE_CAN_PCOMMS_CMD:c_stbd_uhmi_onoff | ALL(!CORE_CAN_PCOMMS_STAT:s_confirmation_stbd_uhmi_iso) | Close non-isolating relay only if isolating is closed |

*Table 31: LIGHTS profile configurations.*

### 8.5 GFI_MUX Profile Configurations

| *Input Channel:Signal* | *Output Channel:Signal* | *Transform* | *Description* |
|---|---|---|---|
| VEHICLE_GFI_MUX_UI_CMD:u_select_channel_1 | VEHICLE_GFI_MUX_CMD:SELECT_CHANNEL | BIAS(0) | GFI Mux hold on vehicle channel 1 |
| VEHICLE_GFI_MUX_UI_CMD:u_select_channel_2 | VEHICLE_GFI_MUX_CMD:SELECT_CHANNEL | BIAS(1) | GFI Mux hold on vehicle channel 2 |
| VEHICLE_GFI_MUX_UI_CMD:u_select_channel_3 | VEHICLE_GFI_MUX_CMD:SELECT_CHANNEL | BIAS(2) | GFI Mux hold on vehicle channel 3 |
| VEHICLE_GFI_MUX_UI_CMD:u_select_channel_4 | VEHICLE_GFI_MUX_CMD:SELECT_CHANNEL | BIAS(3) | GFI Mux hold on vehicle channel 4 |

*Table 32: GFI_MUX profile configurations.*

### 8.6 Main Camera Pan and Tilt Profile Configurations

| *Input Channel:Signal* | *Output Channel:Signal* | *Transform* | *Description* |
|---|---|---|---|
| SUB_CTRL_PCOMMS_STAT:s_main_pan_cos | SUB_CTRL_PCOMMS_UI_STAT:us_main_pan_cos | LINEAR(0.000194175,-1.73786)\|BOUNDS(-1,1) | "Hall effect to cos. **NOTE**: For logs that still have sin/cos swapped change the input to _sin". |
| SUB_CTRL_PCOMMS_STAT:s_main_pan_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_pan_sin | LINEAR(0.00019802,-1.77228)\|BOUNDS(-1,1) | "Hall effect to sin. **NOTE**: For logs that still have sin/cos swapped change the input to _cos". |
| SUB_CTRL_PCOMMS_STAT:s_main_shld_cos | SUB_CTRL_PCOMMS_UI_STAT:us_main_shld_cos | LINEAR(0.00019802,-1.77228)\|BOUNDS(-1,1) | "Hall effect to cos. **NOTE**: For logs that still have sin/cos swapped change the input to _sin". |
| SUB_CTRL_PCOMMS_STAT:s_main_shld_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_shld_sin | LINEAR(0.000194175,-1.73786)\|BOUNDS(-1,1) | "Hall effect to sin. **NOTE**: For logs that still have sin/cos swapped change the input to _cos". |
| SUB_CTRL_PCOMMS_STAT:s_main_tilt_cos | SUB_CTRL_PCOMMS_UI_STAT:us_main_tilt_cos | LINEAR(0.000186916,-1.6729)\|BOUNDS(-1,1) | "Hall effect to cos. **NOTE**: For logs that still have sin/cos swapped change the input to _sin". |
| SUB_CTRL_PCOMMS_STAT:s_main_tilt_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_tilt_sin | LINEAR(0.000188679,-1.69811)\|BOUNDS(-1,1) | "Hall effect to sin. **NOTE**: For logs that still have sin/cos swapped change the input to _cos". |
| SUB_CTRL_PCOMMS_UI_STAT:us_main_pan_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_cam_pan_deg | ATAN2(SUB_CTRL_PCOMMS_UI_STAT:us_main_pan_sin, SUB_CTRL_PCOMMS_UI_STAT:us_main_pan_cos,TRUE)\|BIAS(-100) | Main camera pan conversion from sin/cos to degrees. |
| SUB_CTRL_PCOMMS_UI_STAT:us_main_shld_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_cam_shld_deg | ATAN2(SUB_CTRL_PCOMMS_UI_STAT:us_main_shld_sin, | Main camera shld conversion from sin/cos to degrees. |

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| | | SUB_CTRL_PCOMMS_UI_STAT:us_main_shld_cos,TRUE)\|BIAS(-90) | |
| SUB_CTRL_PCOMMS_UI_STAT:us_main_tilt_sin | SUB_CTRL_PCOMMS_UI_STAT:us_main_cam_tilt_deg | ATAN2(SUB_CTRL_PCOMMS_UI_STAT:us_main_tilt_sin,SUB_CTRL_PCOMMS_UI_STAT:us_main_tilt_cos,TRUE)\|BIAS(100) | Main camera tilt conversion from sin/cos to degrees. |

*Table 33: Main camera pan and tilt profile configurations.*

### 8.7 Main Camera Command Profile Configurations

| Input Channel:Signal | Output Channel:Signal | Transform | Description |
|---|---|---|---|
| PSI_PCOMMS_STAT:s_main_cam_pan | PSI_PCOMMS_UI_STAT:u_main_cam_pan_deg | LINEAR(-0.0035473,60) | Pilot Main camera joystick pan control |
| PSI_PCOMMS_STAT:s_main_cam_shld | PSI_PCOMMS_UI_STAT:u_main_cam_shld_deg | LINEAR(0.00202703,30) | Pilot Main camera joystick shld control |
| PSI_PCOMMS_STAT:s_main_cam_tilt | PSI_PCOMMS_UI_STAT:u_main_cam_tilt_deg | LINEAR(0.00168919,-30) | Pilot Main camera joystick tilt control |

*Table 34: Main camera pan and tilt command (MAIN_CAM_PT_CMD) profile configurations.*

***End Page***