BCS PDI Builder

Release 6.12.41

Embention

2023-09-15

CONTENTS

1 Quick Start **3**

1.1 Download . . . 3

1.2 Installation . . . 3

2 Configuration **5**

2.1 Veronte . . . 5

2.1.1 Unit name . . . 5

2.1.2 Attitude . . . 6

2.1.3 Frequencies . . . 6

2.1.4 Operator position . . . 7

2.1.5 GPIO . . . 8

2.1.6 Status . . . 10

2.2 Connections . . . 10

2.2.1 Arbiter . . . 11

2.2.2 FTS . . . 11

2.2.3 GPIO . . . 12

2.2.4 I2C . . . 15

2.2.5 Others . . . 15

2.2.6 PWM . . . 16

2.2.7 Serial . . . 20

2.2.8 USB . . . 21

2.3 Sensors . . . 22

2.3.1 RPM . . . 22

2.3.2 Lidar . . . 23

2.3.3 Internest . . . 24

2.4 Input/Output . . . 25

2.4.1 I/O Setup . . . 25

2.4.2 CAN Setup . . . 35

2.4.2.1 Configuration . . . 36

2.4.2.2 Custom Messages . . . 39

2.4.2.3 Mailboxes . . . 40

2.4.3 Digital Input . . . 40

2.5 Control . . . 49

2.5.1 Phases . . . 49

2.5.2 Modes . . . 52

2.5.3 Arcade axis . . . 55

2.6 Automations . . . 57

2.6.1 New automation . . . 59

2.6.2 Other options . . . 62

2.6.2.1 Events . . . 64

2.6.2.1.1 Alarm . . . 67
2.6.2.1.2 Area . . . 68
2.6.2.1.3 Button . . . 69
2.6.2.1.4 Mode . . . 71
2.6.2.1.5 Phase . . . 71
2.6.2.1.6 Route . . . 72
2.6.2.1.7 Timer . . . 73
2.6.2.1.8 Variable . . . 75
2.6.2.2 Actions . . . 76
2.6.2.2.1 Atmosphere Calibration . . . 77
2.6.2.2.2 Change active sensor . . . 79
2.6.2.2.3 Command block . . . 79
2.6.2.2.4 Custom CAN TX . . . 82
2.6.2.2.5 Custom Serial TX . . . 83
2.6.2.2.6 DEM calibration . . . 84
2.6.2.2.7 Enable/Disable Wind Estimation . . . 85
2.6.2.2.8 FTS Activation . . . 86
2.6.2.2.9 Feature . . . 87
2.6.2.2.10 Format SD . . . 89
2.6.2.2.11 Go to . . . 90
2.6.2.2.12 Mode . . . 91
2.6.2.2.13 Obstacle avoidance . . . 91
2.6.2.2.14 Output . . . 92
2.6.2.2.15 Periodical . . . 94
2.6.2.2.16 Phase . . . 95
2.6.2.2.17 Ports . . . 96
2.6.2.2.18 Run block program . . . 96
2.6.2.2.19 Safety Bits . . . 98
2.6.2.2.20 Select Arcade axis . . . 99
2.6.2.2.21 Stick priority . . . 100
2.6.2.2.22 Terrain obstacle . . . 101
2.6.2.2.23 Track . . . 102
2.6.2.2.24 User Log . . . 105
2.6.2.2.25 Variable . . . 105
2.6.2.2.26 Yaw . . . 106
2.7 Communications . . . 107
2.7.1 Ports . . . 108
2.7.2 Comstats . . . 109
2.7.3 Iridium . . . 110
2.7.4 Veronte LOS . . . 111
2.8 Stick . . . 112
2.8.1 Transmitter (1-4) . . . 113
2.8.1.1 PPM . . . 113
2.8.1.2 Exponential . . . 115
2.8.1.3 Trim . . . 116
2.8.1.4 Output . . . 117
2.8.2 Virtual Stick . . . 118
2.9 Block Programs . . . 119
2.10 Devices . . . 119
2.10.1 Transponder/ADS-B . . . 121
2.10.2 Cameras . . . 123
2.10.3 Board . . . 127
2.11 Telemetry . . . 130
2.11.1 Telemetry . . . 130

2.11.1.1 Data to VApp 130
2.11.1.2 Onboard Log 132
2.11.1.3 User Log 133
2.11.1.4 Fast Log 134
2.11.2 Sniffer 137
2.12 UI 139
2.12.1 Operation elements 139
2.12.2 Variables 140
2.13 HIL 142
2.14 Safety 145
2.14.1 Checklist 145
2.14.2 Config Manager 146
2.14.3 Safety bits 147

3 Integration Examples 155
3.1 Communication with PC 155
3.2 ArcTrim Button 156
3.3 CAN communication 159
3.4 Data transmission between BCS and 1x Autopilot 164
3.5 External devices 166
3.5.1 Altimeters 166
3.5.1.1 Lidar 166
3.5.1.1.1 I2C lidar 167
3.5.1.1.2 Using lidar readings 168
3.5.1.2 Radar 170
3.5.1.2.1 Smartmicro CAN Radar 170
3.5.2 Radios 175
3.5.2.1 Digi internal radio 175
3.5.2.1.1 Configuration 175
3.5.2.1.2 Operational range 180
3.5.2.2 External radios 180
3.5.3 Servos 182
3.5.3.1 Serial 182
3.5.3.1.1 Volz DA26 - RS485 183
3.5.4 Stick 186
3.5.4.1 PPM Stick 186
3.5.4.1.1 General case: BCS unit sends commands directly to the 1x air unit 186
3.5.4.1.2 Simulation case (HIL) 193
3.5.4.1.3 On-board PPM receiver case 193
3.5.4.2 USB joystick 199
3.5.4.3 Virtual Stick 199
3.5.5 Veronte products 200
3.5.5.1 CEX/MEX 200
3.5.5.2 MC01 210
3.5.5.3 VSE (Veronte Stick Expander) 215

4 Troubleshooting 217
4.1 Communication lost with intenal Digi radio 217
4.2 Debug serial messages transmission 218
4.3 Maintenance mode 220
4.4 Maintenance mode (loaded with errors) 221
4.5 Radios paired but 1x air unit not showing 222
4.6 Reduce GNC Task frequency 222

BCS | PDI BUILDER

BCS PDI Builder is the configuration tool for the Veronte control station **BCS**, so it can be adapted to a specific flight control system.

CHAPTER

ONE

QUICK START

BCS PDI Builder is the main configuration tool to adapt a **Veronte BCS** to a specific flight control system, including user-defined commnication protocols. **BCS PDI Builder** includes:

- Telemetry: real-time reception of UAV metrics, such as sensors, actuators and control states.
- Configuration: edit communication settings according to the station control requirements.
- Automations: actions that are automatically executed when a set of configured conditions are accomplished.
- Block Programs: BCS can be programmed with a friendly-user programming language.

Once **BCS** has been detected on **Veronte Link**, install **BCS PDI Builder**.

1.1 Download

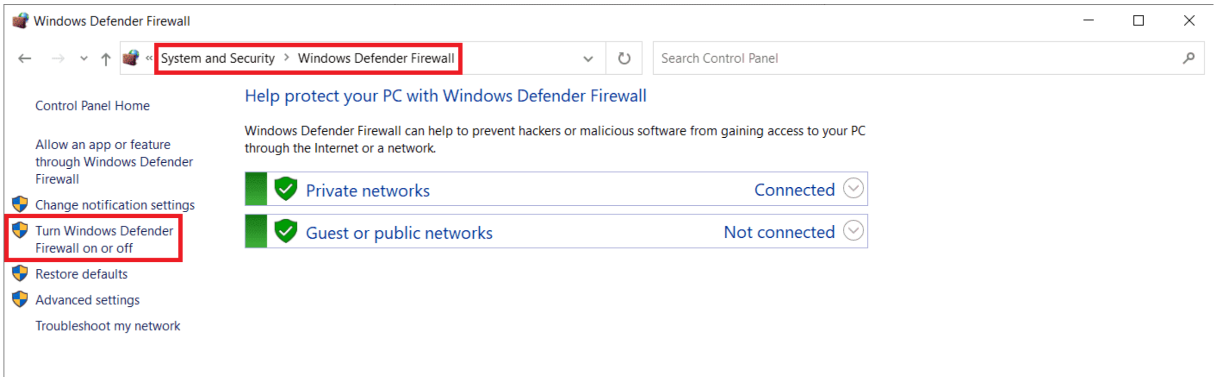
Once the **Veronte BCS** has been purchased, a GitHub release should be created for the customer with the application.

To access to the release and download the software, read the Releases section of the **Joint Collaboration Framework** manual.

1.2 Installation

To install **BCS PDI Builder** on Windows just execute "VeronteBCSPDIBuilder.exe" and follow the instructions of the setup wizard.

Warning: If users have any problems with the installation, please disable the antivirus and the Windows firewall. Disabling the antivirus depends on the antivirus software. To disable the firewall, go to "Control Panel" → "System and Security" → "Windows Defender Firewall" and then, click on "Turn windows Defender Firewall on or off".



Windows Defender Firewall

System and Security > Windows Defender Firewall

Help protect your PC with Windows Defender Firewall

Windows Defender Firewall can help to prevent hackers or malicious software from gaining access to your PC through the Internet or a network.

Private networks — Connected

Guest or public networks — Not connected

Search Control Panel
Control Panel Home
Allow an app or feature through Windows Defender Firewall
Change notification settings
Turn Windows Defender Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network

Fig. 1: **Windows Defender Firewall**

CHAPTER

TWO

CONFIGURATION

2.1 Veronte

2.1.1 Unit name

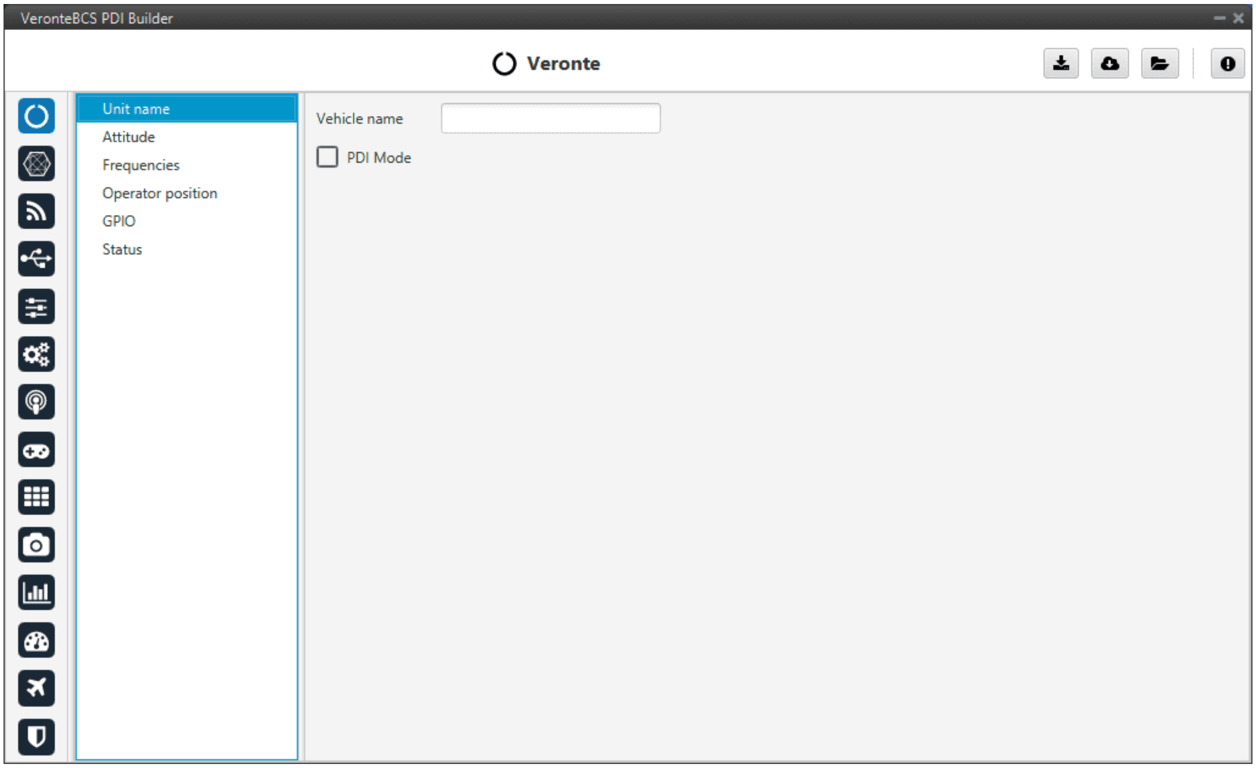


Fig. 1: **Unit name section**

- **Vehicle name**: The user can define the name of the configuration.
- **PDI Mode**: It can be enabled or disabled. PDI mode allows the user to change the setup if **BCS** is not in INI phase.

Warning:

- **Not** being in **PDI mode**, the user cannot do out of the INI phase:
  - Reboot **BCS**
  - Change **BCS** setup (i.e. save to SD card)
  - Enter manually in maintenance mode
- The variable '**System error**' prevents operation in normal mode (**not PDI mode**). A list of all errors that can cause this bit to be set can be found in the Activation system error bits section of the **1x Software Manual**.
- If **BCS** has '**sensor errors**' and it is in normal mode (**not PDI mode**), the user will not be able to switch to another flight phase, it will remain in **INI phase**.

Danger: PDI mode is intended for development purposes since, as detailed above, it **allows flight phase changes with system, sensor and PDI errors**.

It is highly recommended to limit its use to simulation and ground testing of peripherals during the development phase.

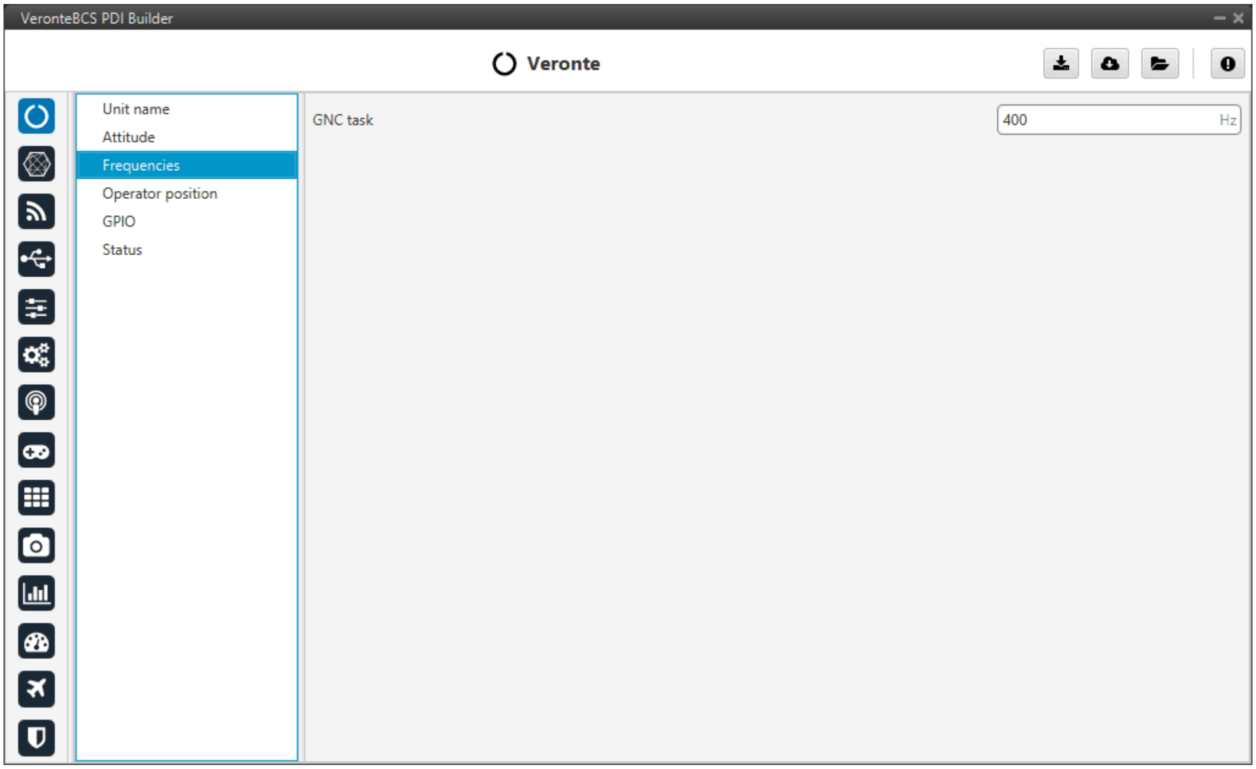
Therefore, as **it is not advisable to operate in PDI mode**, please disable it once the configuration is finished and intended to be used in flight.

2.1.2 Attitude

Deprecated

2.1.3 Frequencies

The frequency of the GNC task refers to the maximum working frequency of the core. In this case, **400 Hz**, which is the **maximum possible**.



GNC task 400 Hz

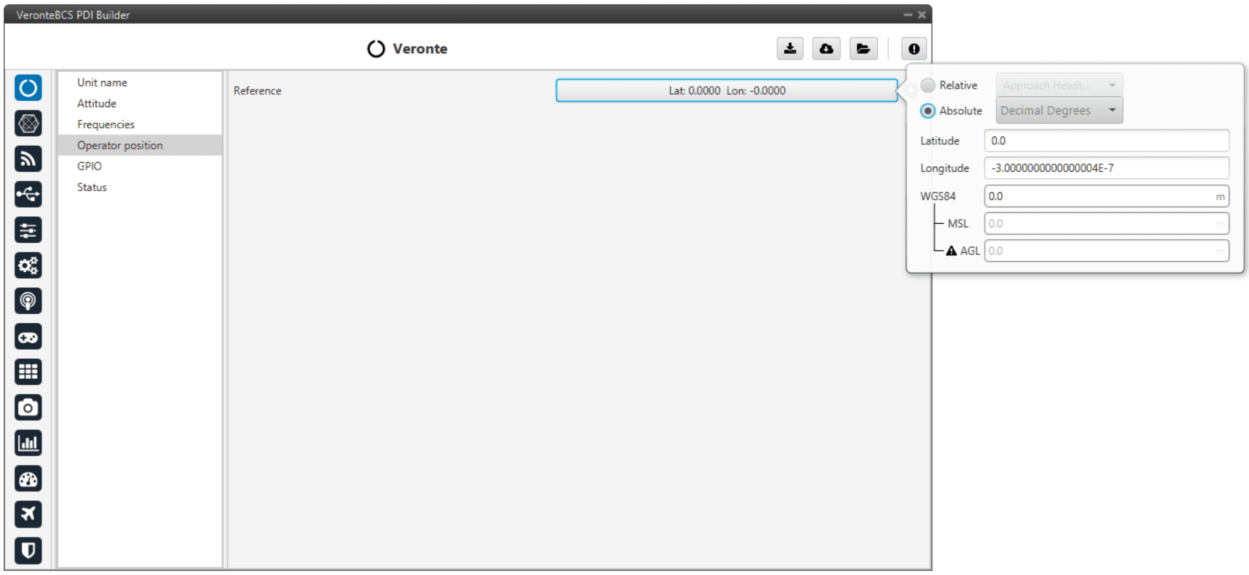
Fig. 2: **Frequencies section**

Warning: Only 400 Hz can be used for simple configurations, so it is often necessary to reduce the frequency to 250-300 Hz. To find out why the user should reduce the GNC Task frequency, see *Reduce GNC Task frequency -> Troubleshooting section* of this manual.

2.1.4 Operator position

The operator position is the reference for the aircraft, which is used to calculate the distance allowed by the license.

Press 🔗 or ⚙ to switch between reference modes: *valid ID* and *invalid ID*.



Reference: Lat: 0.0000 Lon: -0.0000

○ Relative Approach Headi...

● Absolute Decimal Degrees

Latitude 0.0

Longitude -3.0000000000000004E-7

WGS84 0.0 m

MSL 0.0 m

AGL 0.0 m

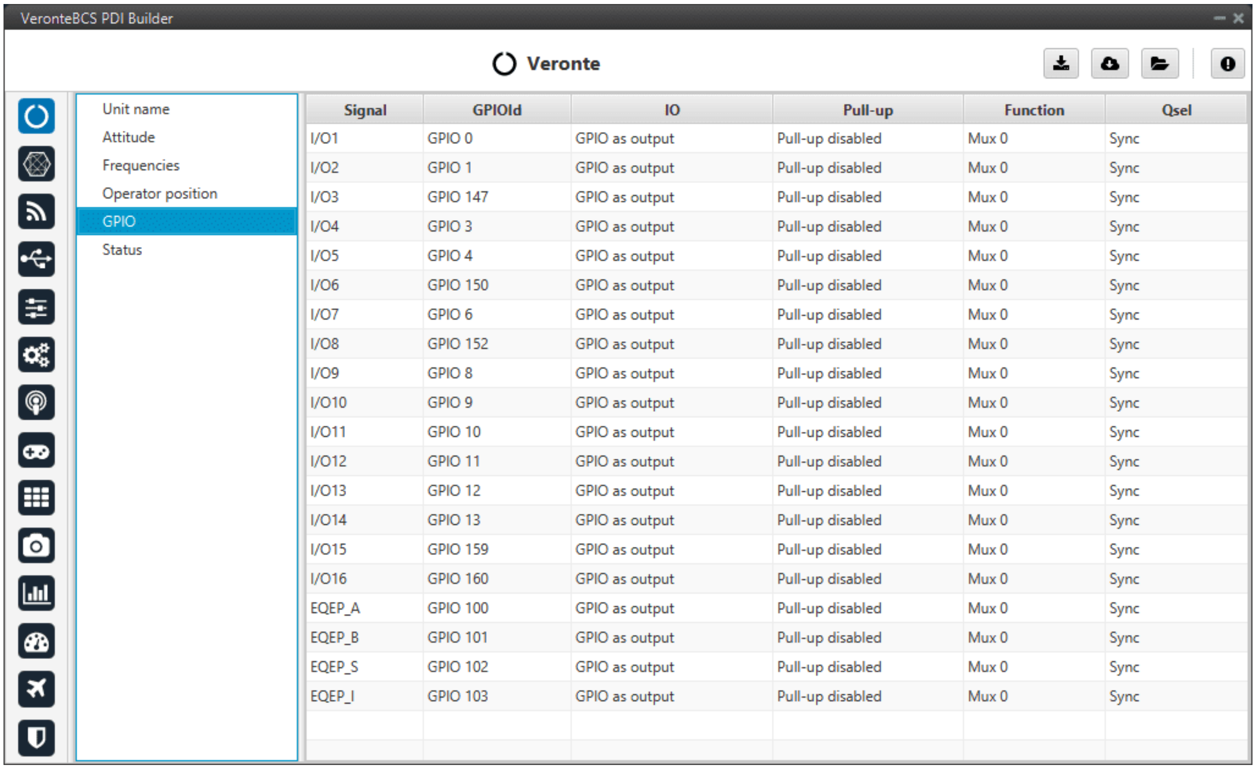
Fig. 3: **Operator position - Valid ID mode**

The absolute position is a specific point defined. The relative position is defined as a **Custom point** in **Veronte Ops**, to know more read the Custom points section of **Veronte Ops manual**.

Tip: In operation, if the air unit does not have the license activated, it is recommended to set in the air unit (**1x**) the position of the ground unit (**BCS**). Remember to use **1x PDI Builder** to configure the **Autopilot 1x** and the **BCS PDI Builder** to configure the control station **BCS**.

2.1.5 GPIO

In this tab each individual GPIO behavior can be configured:



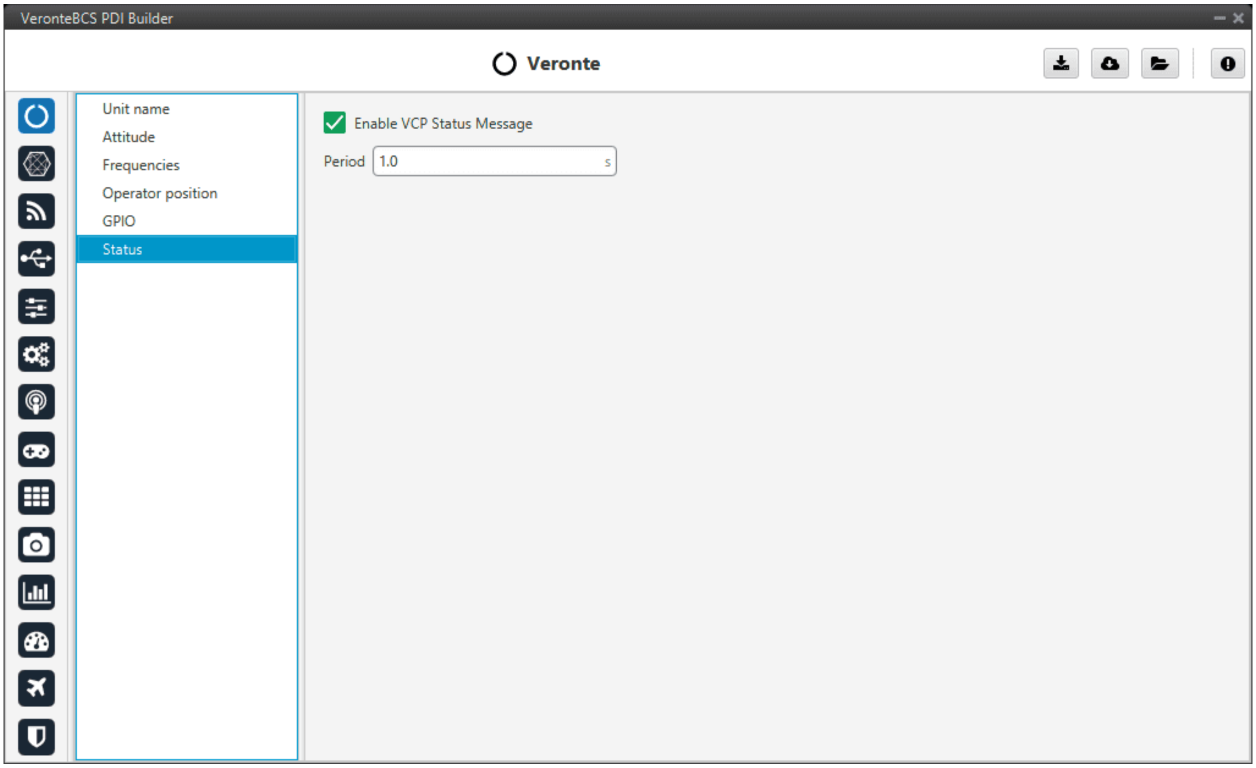
Signal	GPIOId	IO	Pull-up	Function	Qsel
I/O1	GPIO 0	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O2	GPIO 1	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O3	GPIO 147	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O4	GPIO 3	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O5	GPIO 4	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O6	GPIO 150	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O7	GPIO 6	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O8	GPIO 152	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O9	GPIO 8	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O10	GPIO 9	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O11	GPIO 10	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O12	GPIO 11	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O13	GPIO 12	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O14	GPIO 13	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O15	GPIO 159	GPIO as output	Pull-up disabled	Mux 0	Sync
I/O16	GPIO 160	GPIO as output	Pull-up disabled	Mux 0	Sync
EQEP_A	GPIO 100	GPIO as output	Pull-up disabled	Mux 0	Sync
EQEP_B	GPIO 101	GPIO as output	Pull-up disabled	Mux 0	Sync
EQEP_S	GPIO 102	GPIO as output	Pull-up disabled	Mux 0	Sync
EQEP_I	GPIO 103	GPIO as output	Pull-up disabled	Mux 0	Sync

Fig. 4: **GPIO section**

1. **Signal**: Pin ID as described in the Pinout section of the **BCS** user manual.
2. **GPIOId**: GPIO ID of the microcontroller.
3. **IO**: Define GPIO as an input or ouput.
4. **Pull-up**: Enable or disable the pull-up resistance.
5. **Function**: Mux 0: GPIO, Mux 1: PWM, Mux 2, Mux 3, etc. These are the different functionalities that the GPIO can have, this depends on the multiplexer.
6. **Qsel**: This is the "input qualification", it is used to control how the value of a GPIO is evaluated. The available options are:
   - **Sync**: The value is taken as whatever is present at the time it is checked (synchronously). This is the default mode of all GPIO pins.
   - **3 Samples**: The value is checked 3 times and the value is only changed when the 3 times are the same.
   - **6 Samples**: Same as the previous one, but checking 6 times instead of 3.
   - **ASync**: No checks are performed. It is used when it is not used as GPIO.

2.1.6 Status

This option enables the periodic sending of the status message that **Veronte Link** uses to recognise the **BCS**.



☑ Enable VCP Status Message

Period 1.0 s

Fig. 5: **Status section**

- **Period**: Enter a desired period to send repeatedly the status message.

Note: VCP is the Veronte Communication Protocol. To know more, read the VCP user manual.

2.2 Connections

Here the Input/Output ports of the control station can be configured. Depending on the configurable port selected the user will need to provide different parameters.

Each connection is associated with a specific pin number. For more details see the Pinout section of the **BCS** hardware manual.

2.2.1 Arbiter

Pins 45 and 46 are dedicated pins to allow the UART communication with the Safety micro Controller (SuC). This microcrollers is in charge of monitoring the state of the main microcontroller and providing the Flight Termination Signals (FTS).

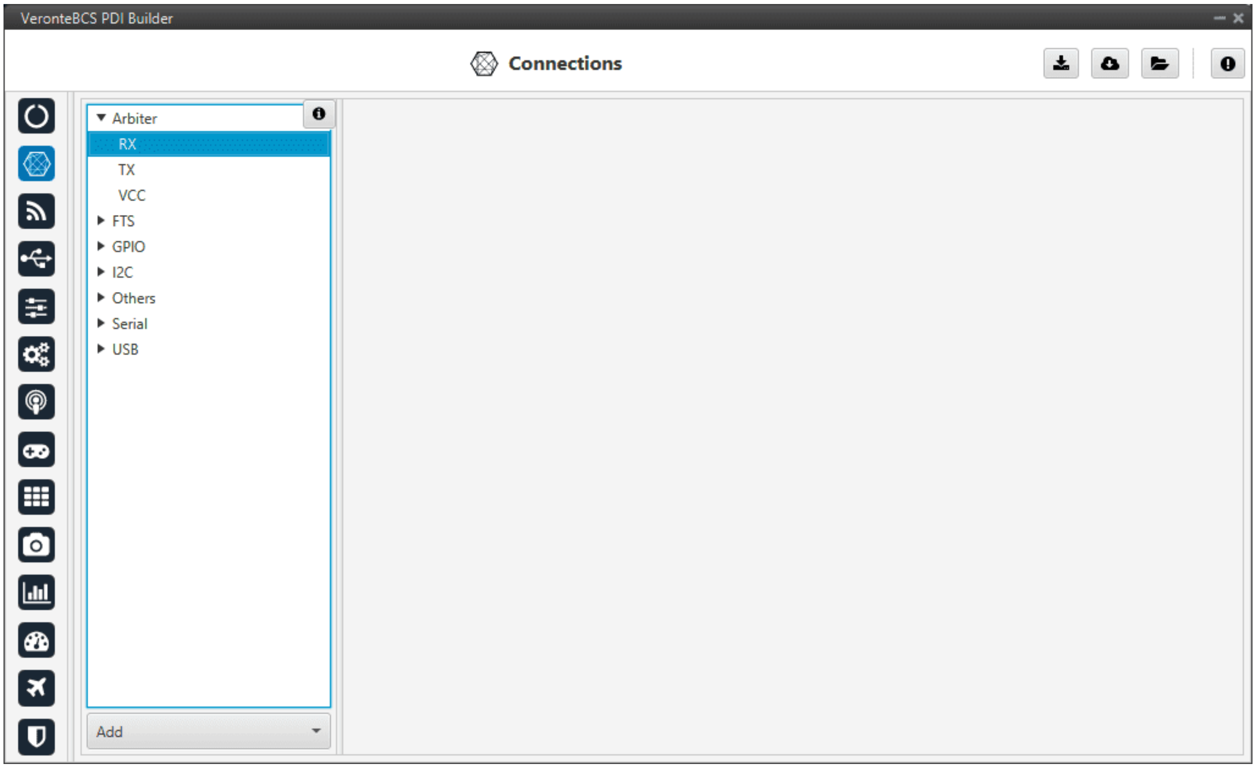


Fig. 6: **Arbiter menu**

2.2.2 FTS

The FTS (Flight Termination System) is a signal that is activated when a sytem error occurs (System Error bit is False).

There’s a group of bits that when failed cause the system error, to see more about the condition that make the system error happen, go to Activation system error bits section of the **1x Software Manual**.

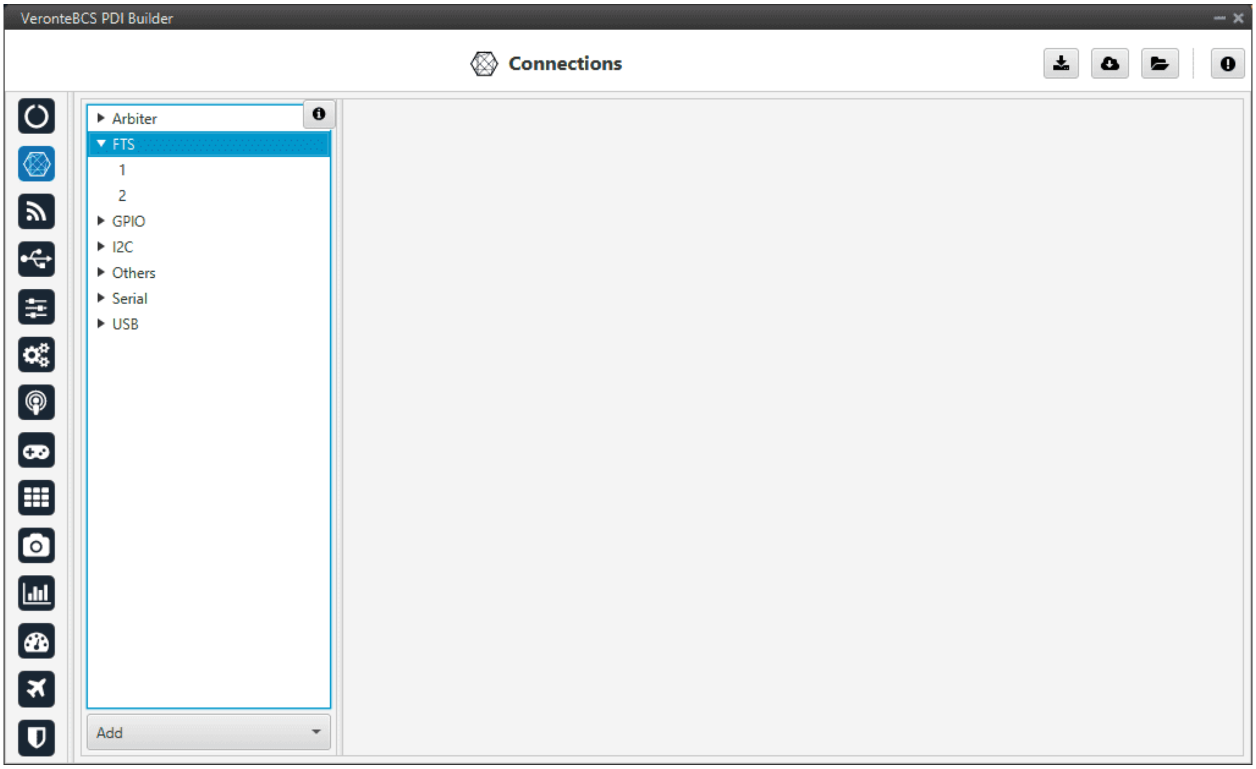


Fig. 7: **FTS menu**

2.2.3 GPIO

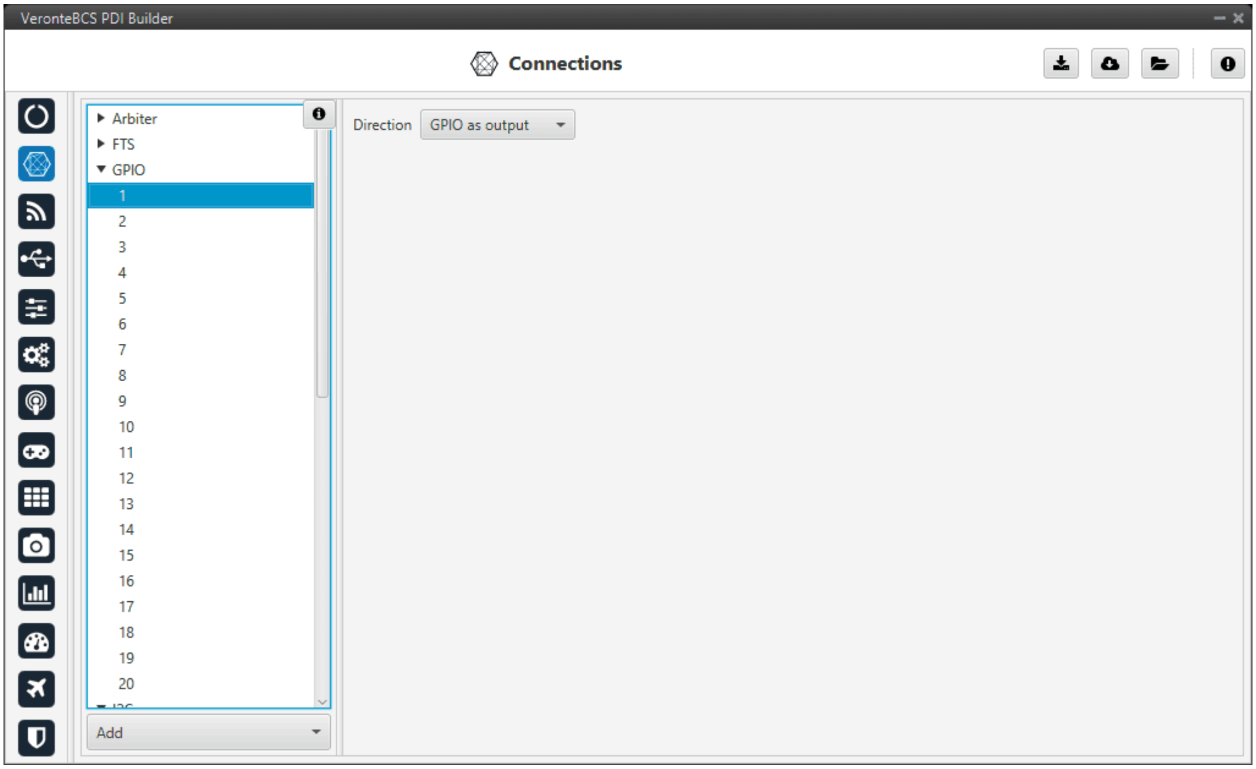
Output pins produce PWM or GPIO signals that are used to move different servos and actuators of the platform.

A GPIO (General Purpose Input/Output) is a generic pin that can be configured as an input or output pin. When this option is configured as an output pin, the value sent will be different from the one sent if it was a PWM.

GPIO pins admit up to 4 different states:

- **ON**: A continuous signal of value 1, made by 3.3V.
- **OFF**: A continuous signal of value 0, made by Ground.
- **PULSE ON**: A single pulse of value 1, with a width specified in seconds.
- **PULSE OFF**: A single pulse of value 0, with a width specified in seconds.

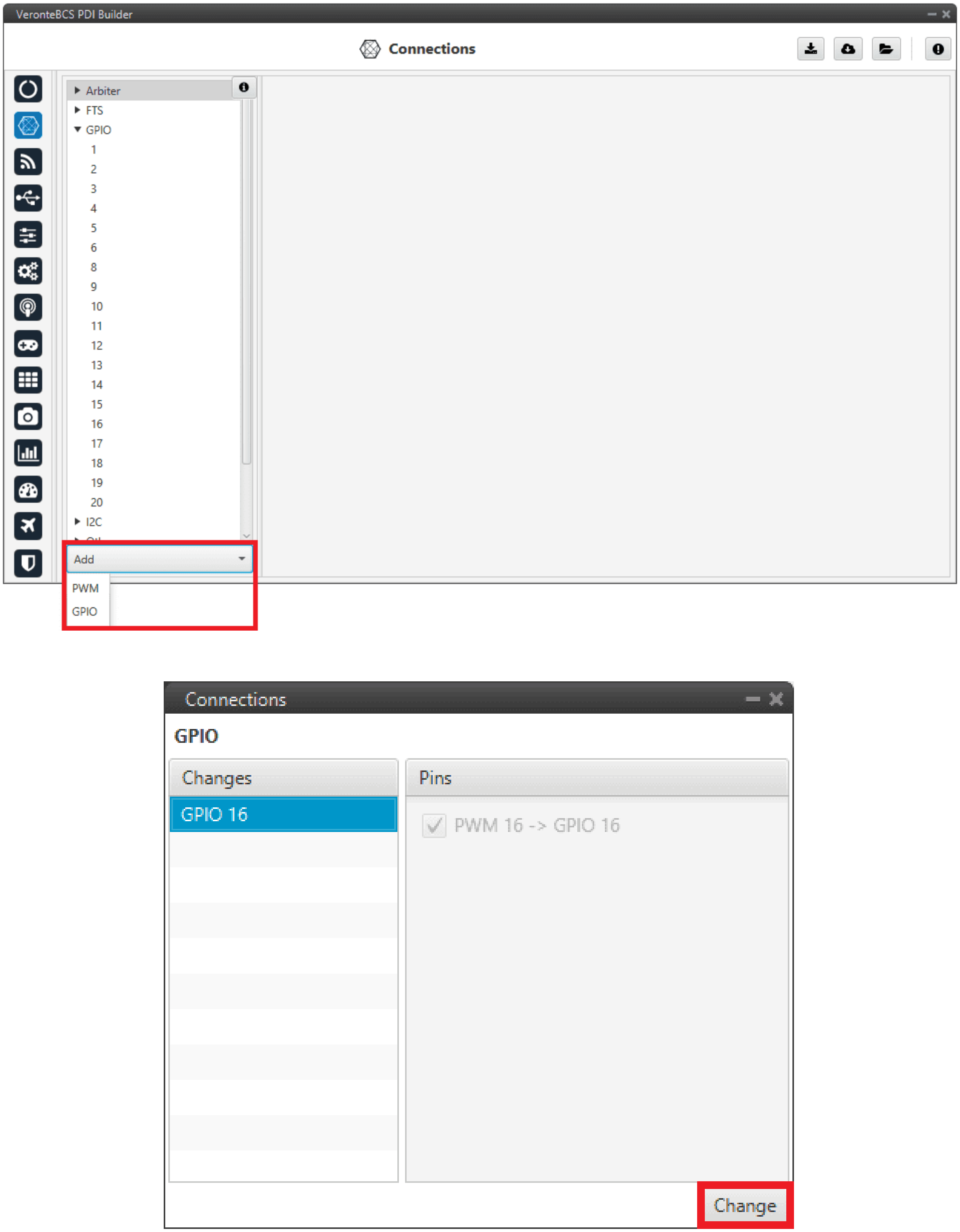
The configuration of the pin output value is done with an *Output* action in the Automations menu.



Direction: GPIO as output

Fig. 8: **GPIO window**

BCS admits up to 20 I/O PWM/GPIO signals. To configure a pin as GPIO after it has been changed to PWM, click *Add* and select GPIO:



☑ PWM 16 -> GPIO 16

Fig. 9: **Add GPIO**

2.2.4 I2C

I2C stands for Inter-Integrated Circuit bus. It is a bus interface connection protocol incorporated into devices for serial communication. It operates in 2 modes: master and slave.

I2C uses only 2 bi-directional open-drain lines for data communication called **SDA** and **SCL**. Both these lines are pulled high.

- **Pin 31 - SCL**: Clock line for I2C bus (0.3V to 3.3V), it carries the clock signal.
- **Pin 32 - SDA**: Data line for I2C bus (0.3V to 3.3V), transfer of data takes place through this pin.

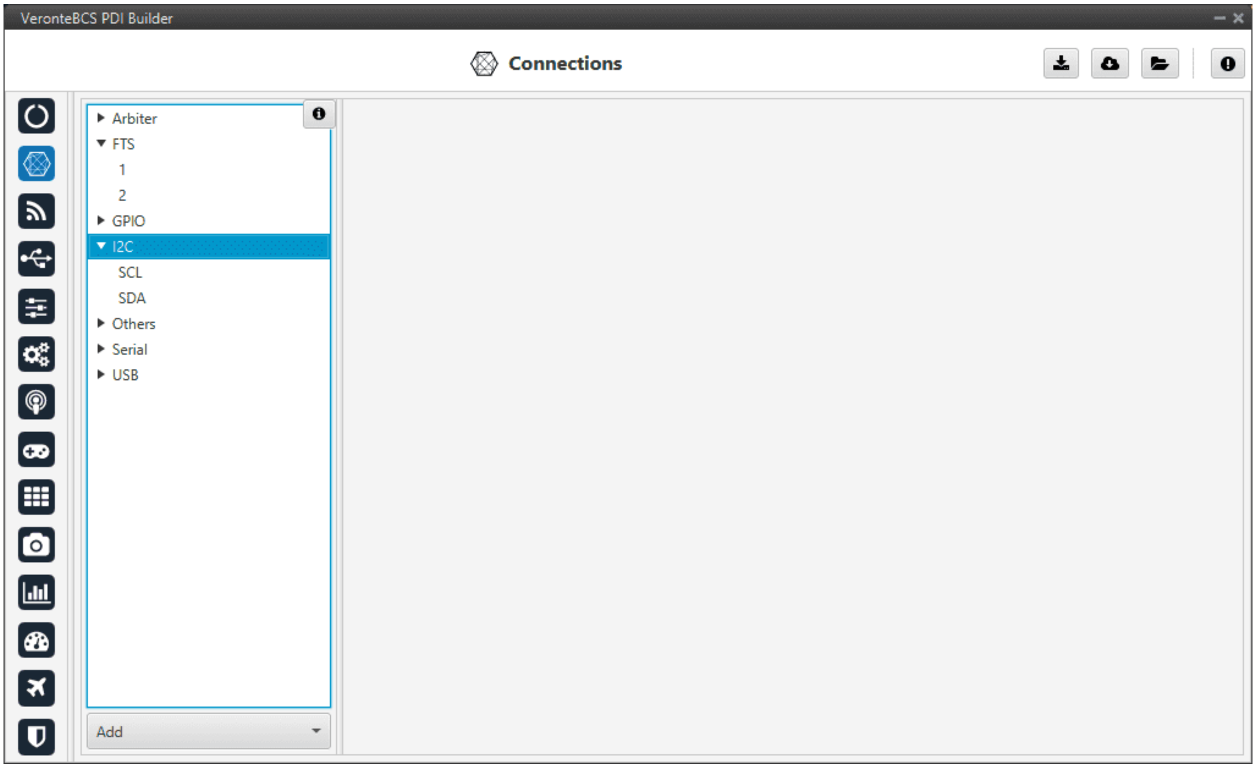


Fig. 10: **I2C menu**

2.2.5 Others

Deprecated

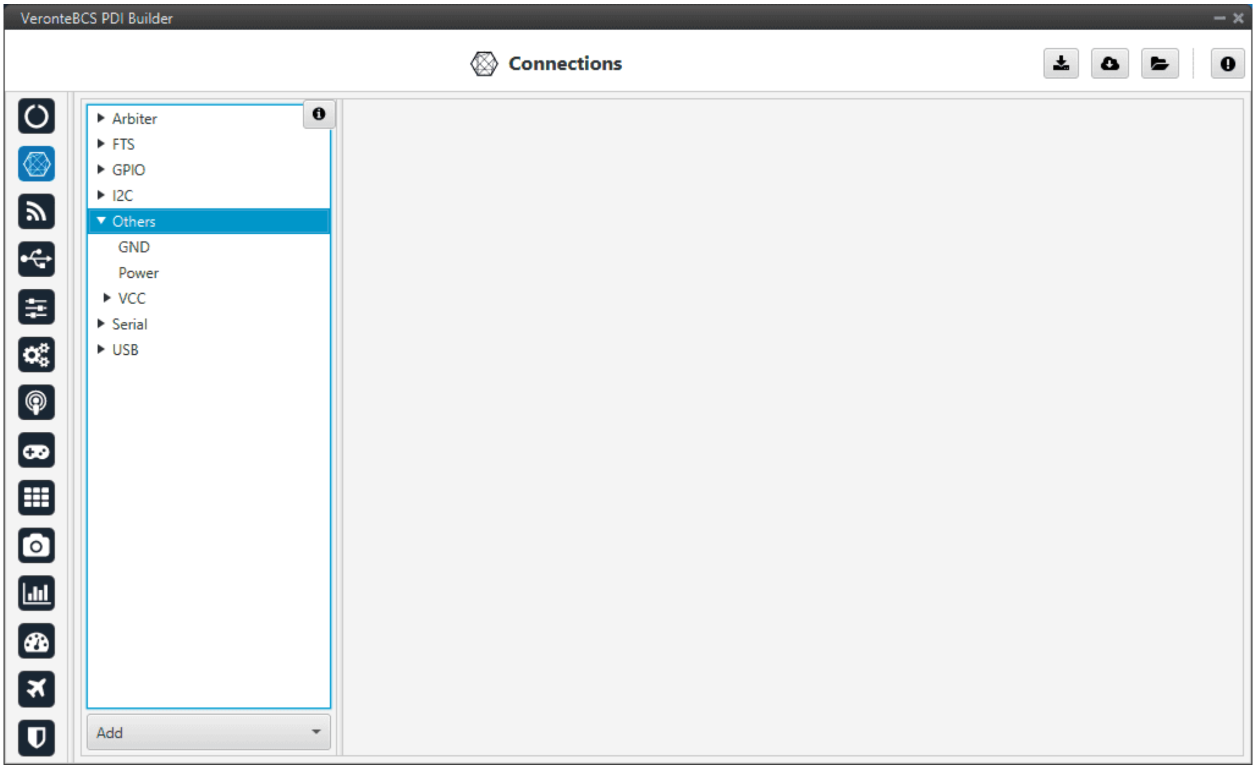


Fig. 11: **Others menu**

2.2.6 PWM

Output pins produce PWM or GPIO signals that are used to move the different servos and actuators of the platform.

The acronym PWM corresponds to Pulse Width Modulation. **BCS** sends a pulse with a certain width that is received by the servo/actuator, and according to the width of such pulse, it changes its behaviour. A wide pulse will correspond to a big movement and a narrow one to a small movement.

By default, all PWM/GPIO pins are configured as GPIO output. So, to configure them as PWM, it is necessary to click *Add*:

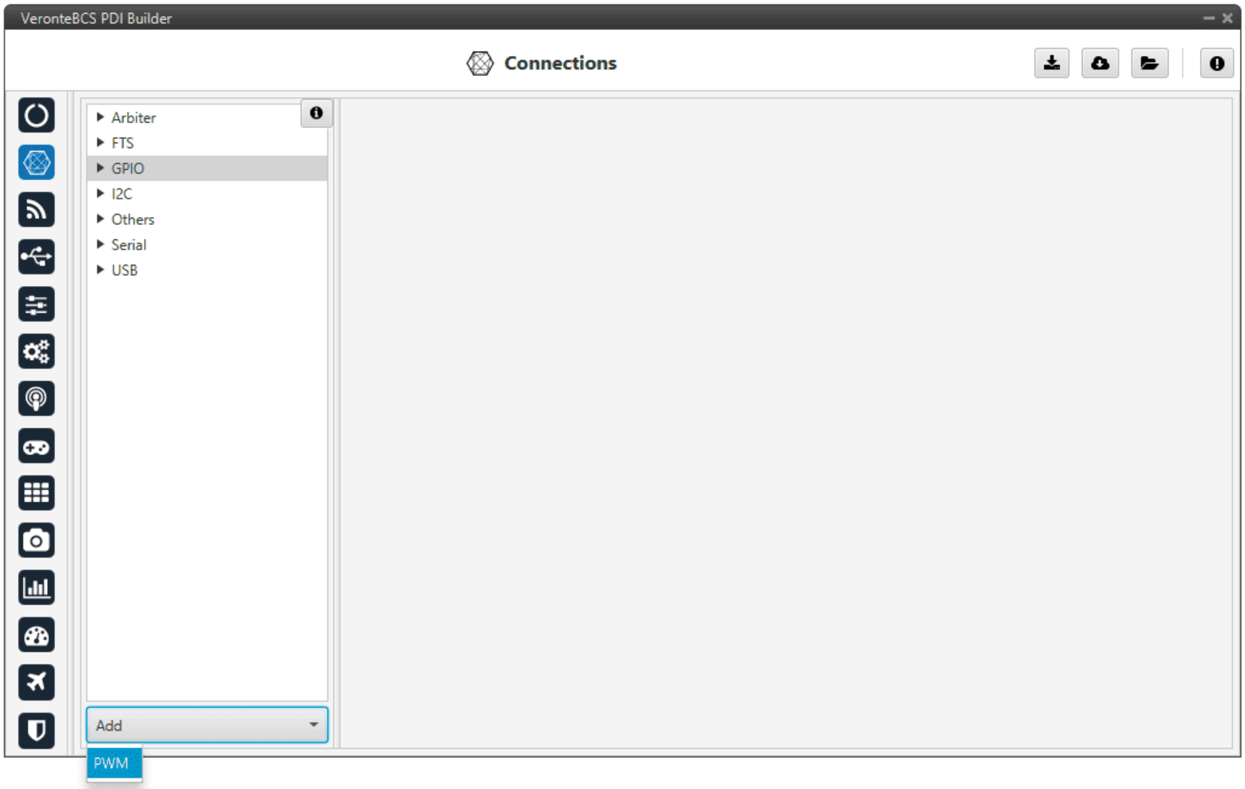


Fig. 12: **Add PWM**

Then, select the GPIO pin the user want to change to PWM. As can be seen, pins are interchangeable.

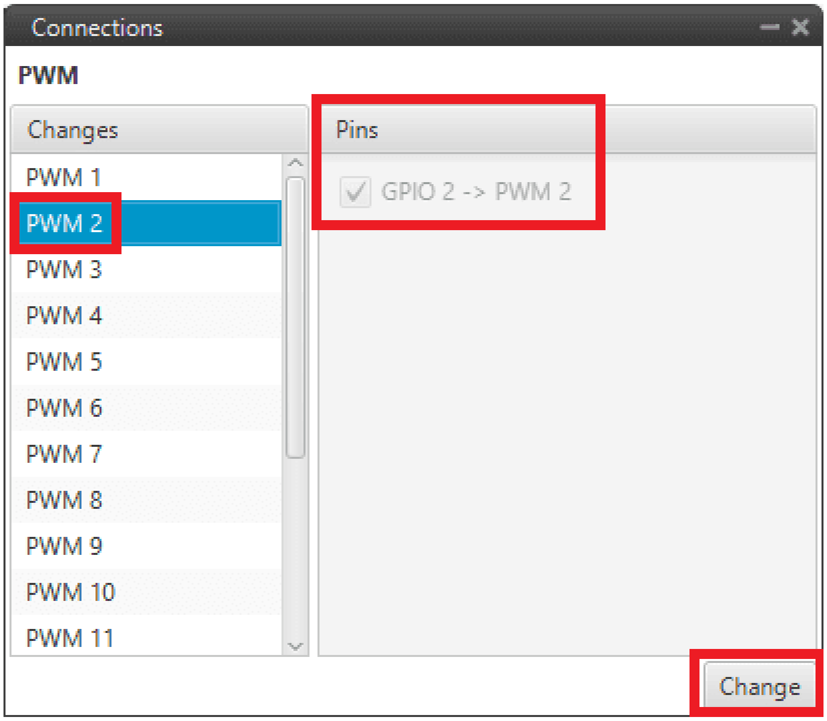


Fig. 13: **PWM change**

As shown in the image below, the GPIO 2 output is now missing as it has been changed to a PWM output.

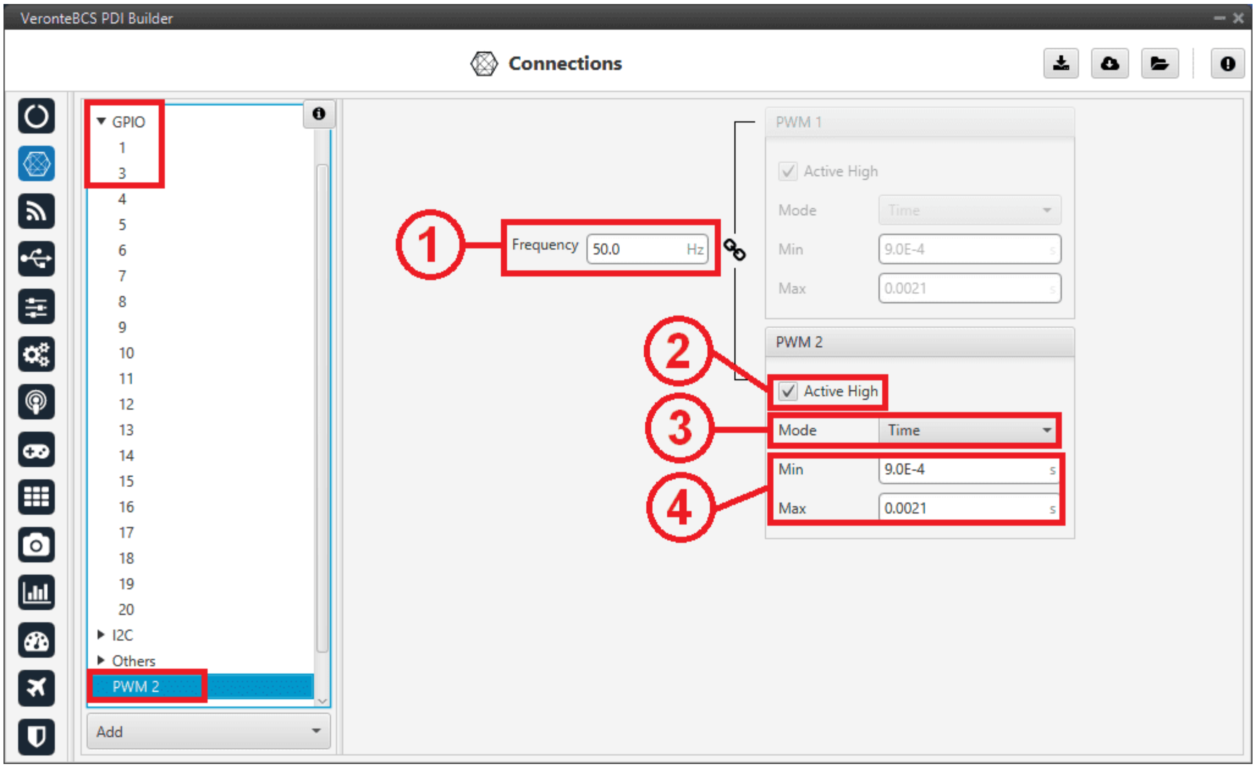


Fig. 14: **PWM menu**

In this menu the following parameters can be configured:

1. **Frequency**: This option determines the period of the pulses sent by the **BCS**. The PWM is built in pairs inside the control station, and that is why the frequency is indicated in pairs, i.e when the frequency of PWM 1 is changed, the one of PWM 2 also changes. The following table shows the PMW pairs as configured.

PWM Pairs	
PWM 1	PWM 2
PWM 3	PWM 4
PWM 5	PWM 6
PWM 7	PWM 8
PWM 9	PWM 10
PWM 11	PWM 12
PWM 13	PWM 14
PWM 15	PWM 16
PWM 17	PWM 18
PWM 19	PWM 20

2. **Active High**: Enable/disable. Polarity high or low.

3. **Mode**: The options available are **Time** and **Duty cycle**. The second option is a a different way of indicating the pulse width. Now the value indicated is a percentage which corresponds to the relationship between the pulse width over the total period of the sent signal. So a 100% duty cycle will correspond to a signal with a constant value of 1, while a 0 % duty cycle implies a constant signal with value 0. Between this two extremes, the pulse width can vary as in the examples shown in the following figure.

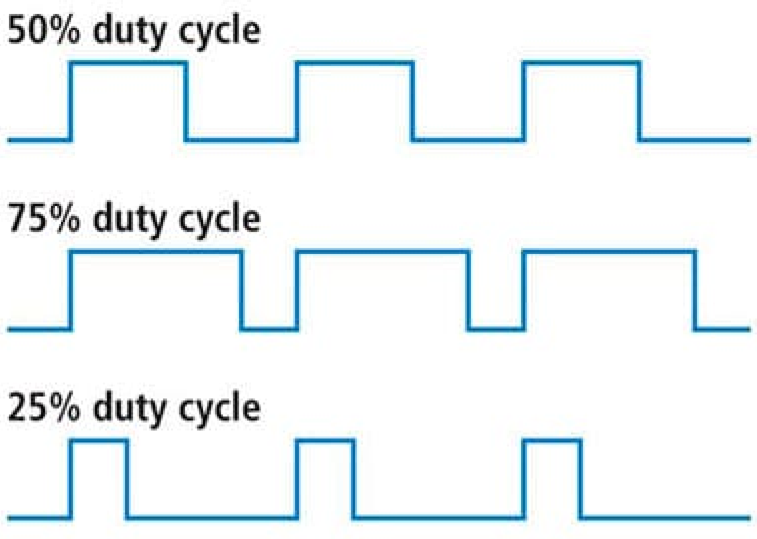


Fig. 15: **Duty Cycle**

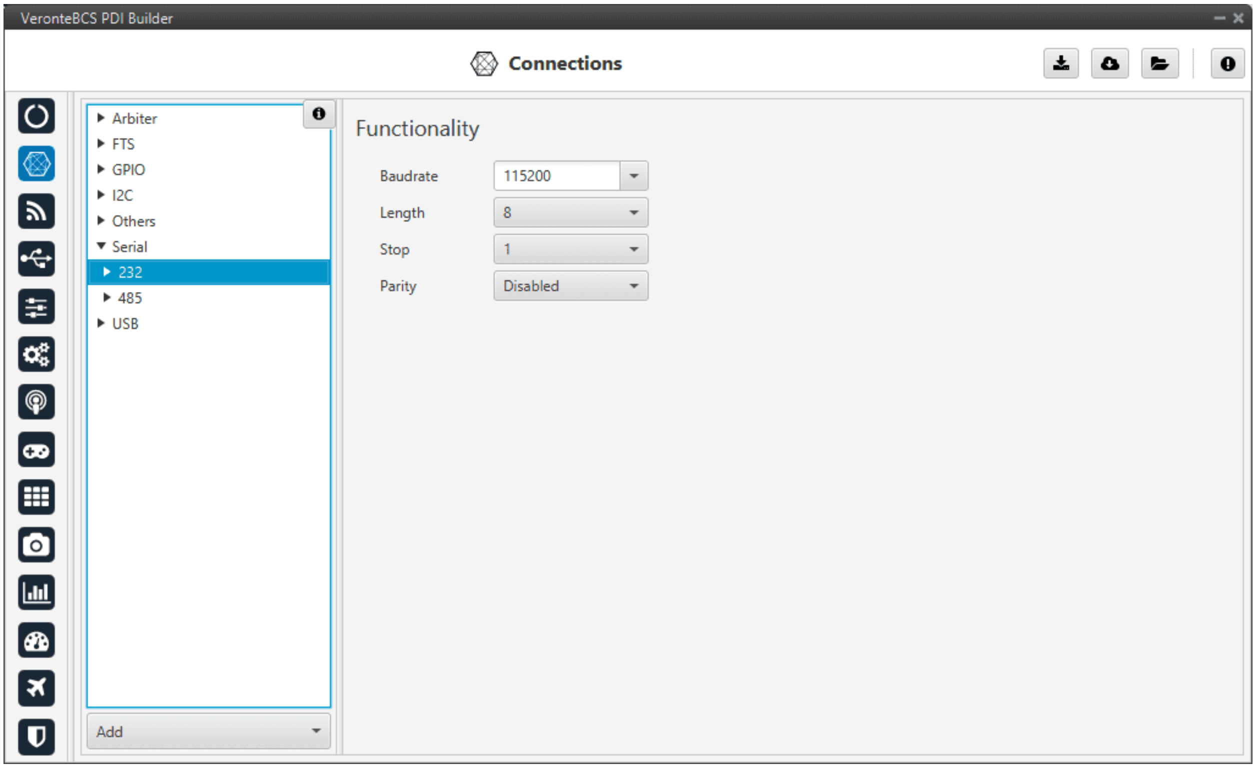
4. **Min/Max**: These parameters are the pulse width values that will make the servo/actuator go to its lowest and highest position. As an example let's consider the servo of an aircraft elevator, a pulse sent by **BCS** of 0.9 ms will correspond with the lowest point of the servo range (-30 degrees for example). On the other hand, a pulse of 2.1 ms will make the servo go to its top position (for example 30 degrees).

Summary

A PWM is a signal which consists of a series of pulses having a width determined by a percentage over a range specified by the parameters Min and Max. The GPIO is a signal with a constant value (1,0) or with a single pulse (1,0).

2.2.7 Serial

Two serial interfaces are available with **BCS**, 1 port RS-232 and 1 port RS-485, however more can be added by using a CEX. Each one of the serial interfaces is associated with a set of pins.



Functionality

Baudrate: 115200
Length: 8
Stop: 1
Parity: Disabled

Fig. 16: **Serial menu**

The following fields can be configured for RS-232 and RS-485:

- **Baudrate**: This specifies how fast data is sent over a serial line.
- **Length**: This defines the number of data bits in each character: 4 to 8.
- **Stop**: Stop bits sent at the end of every character: 1, 1.5, 2.
- **Parity**: Is a method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit. **Disabled**, **odd** or **even**.

Note: All these settings are already specified for a given device, therefore, **BCS** should match with them in order to be able to communicate.

Serial information table:

Port name	RS-232	RS-422 / RS-485
Transfer type	Full duplex	Full duplex / Half Full duplex
Maximum distance	15 meters at 9600 bps	1200 meters at 9600 bps
Topology	Point to point	Point to point / Multi point
Max number of devices	1	1-10 in receive mode / 32

2.2.8 USB

Deprecated.

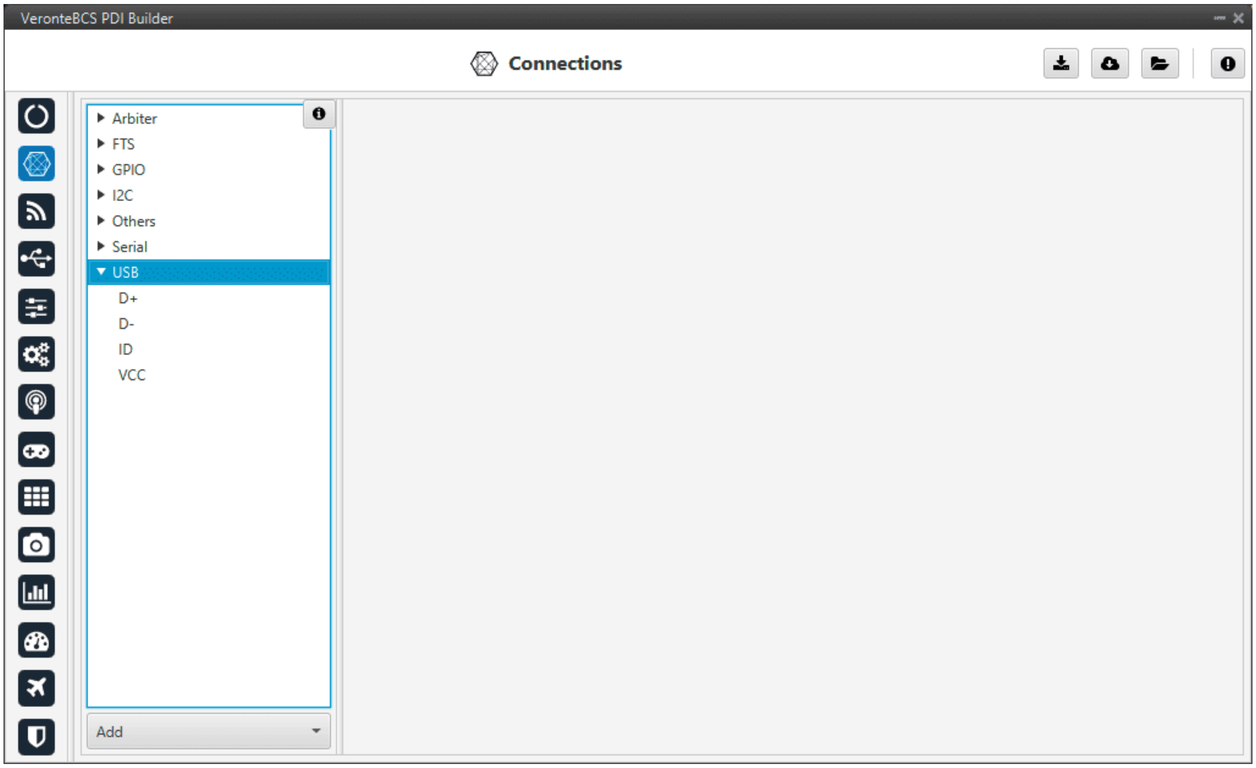
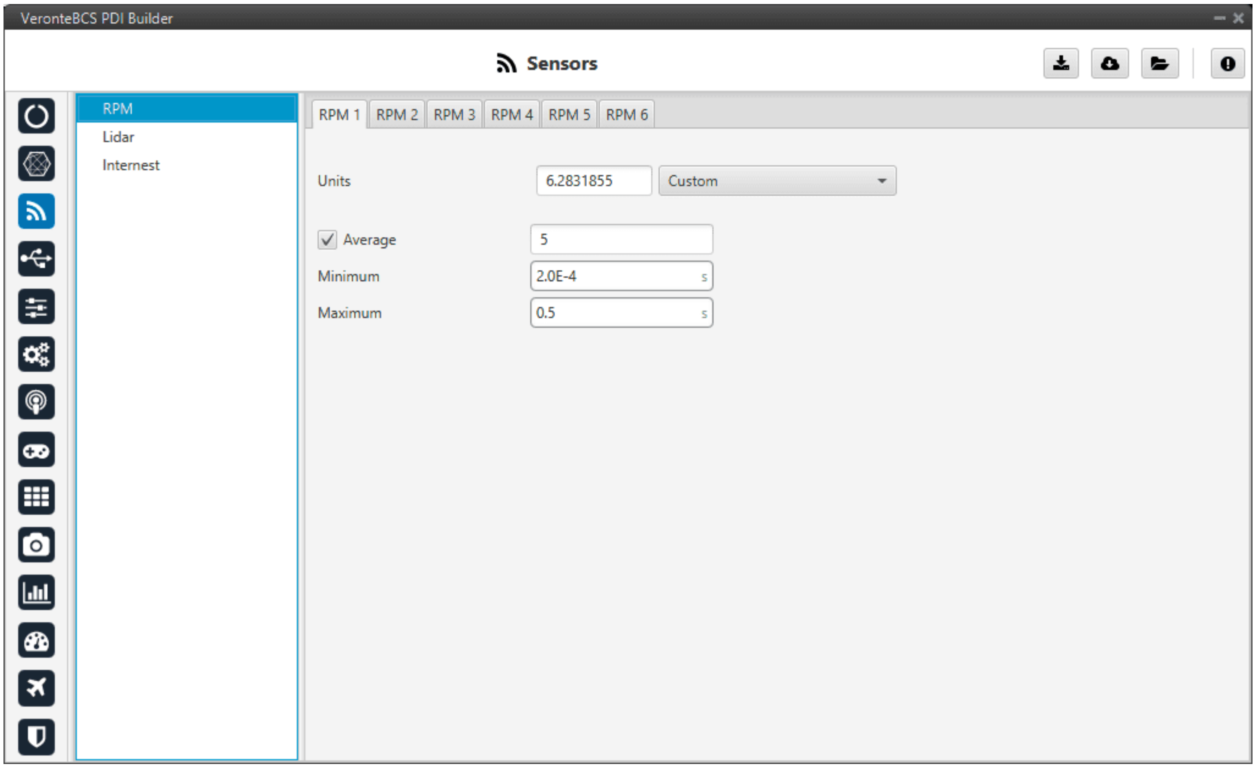


Fig. 17: **USB menu**

2.3 Sensors

2.3.1 RPM

BCS can measure RPMs by measuring up to 6 input sources:



Units 6.2831855 Custom

☑ Average 5

Minimum 2.0E-4 s

Maximum 0.5 s

Fig. 18: **RPM menu**

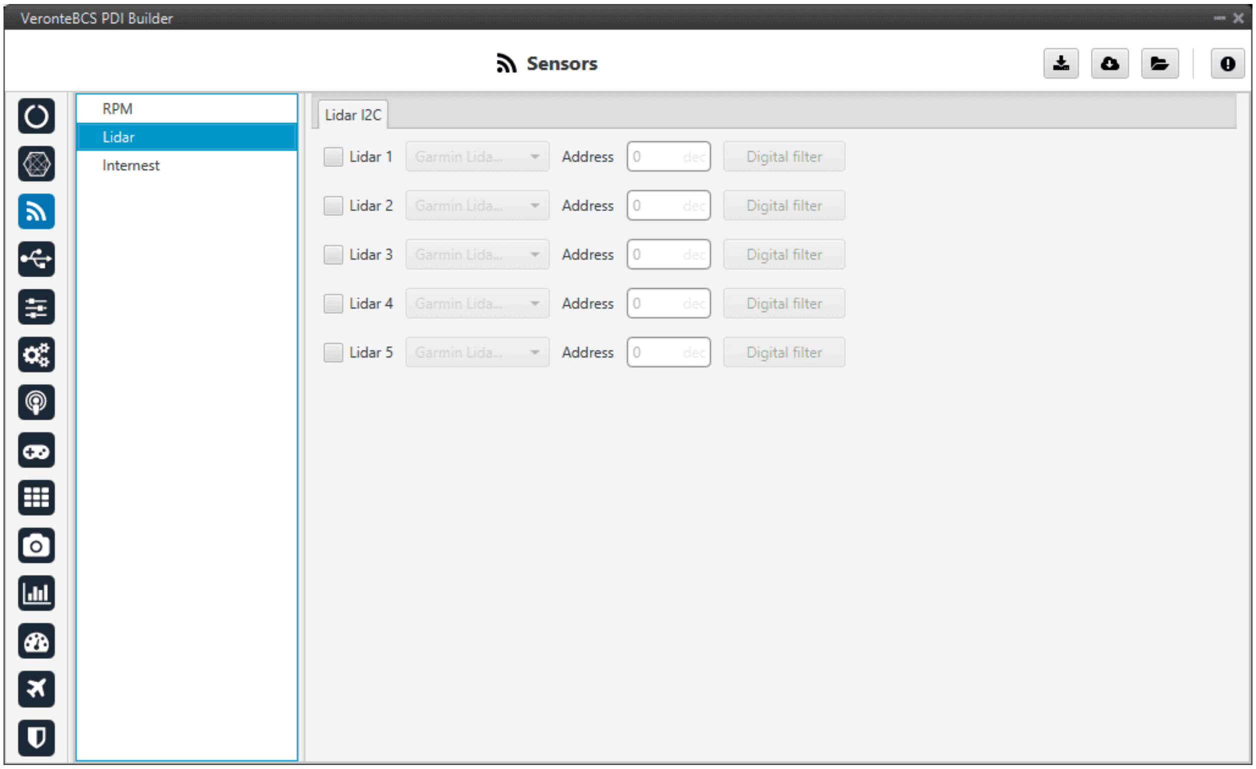
- **Units**: Sensor conversion factor. It can be *Custom*, *Radians per pulse* or *Pulse per cycle*.
- **Average**: Filter to prevent voltaje spikes. The readout of the pulse can be filtered as an average output. The amount of measurements to do the average needs to be specified.
- **Minimum**: Here the minimum expected pulse period needs to be specified. This will discard spurius pulses (e.g. induced by EMI) which are smaller than this minimum pulse.
- **Maximum**: This is the maximum period of time allowed without capturing. If no incoming pulse is received for more than this time, the output RPMs will be 0.

2.3.2 Lidar

The I2C bus allows the connection of several devices with different addresses to the same line via master-slave communication. At this moment, **BCS** supports the following Lidar devices:

- **Garmin LIDAR-Lite v3**: Optical distance measurement sensor with a range from 5 cm to 40 m.
- **SF11 Lidar**: Long range laser altimeter. Supported SF11/B and SF11/C with a range from 0 to 50 m and 0.2 m to 120 m respectively.
- **SF20 Lidar**: OEM laser altimeter module. Supported SF20/C with a range from 0.2 m to 100 m.

BCS allows up to 5 lidar devices to be connected to the system at the same time. The configuration menu can be seen below:



□ Lidar 1 Garmin Lida... Address 0 dec Digital filter

□ Lidar 2 Garmin Lida... Address 0 dec Digital filter

□ Lidar 3 Garmin Lida... Address 0 dec Digital filter

□ Lidar 4 Garmin Lida... Address 0 dec Digital filter

□ Lidar 5 Garmin Lida... Address 0 dec Digital filter

Fig. 19: **Lidar devices**

After enabling the needed number of lidar devices, configurable parameters are:

- **Type of lidar**.
- **Address**: With an accepted value between 16 and 239, this is the origin address from the lidar being configured.
- **Digital filter**: Enables a low pass filter which its cutoff frequency is configured manually, allowing the user to input any desired value in Hz. It is a **software filter**.

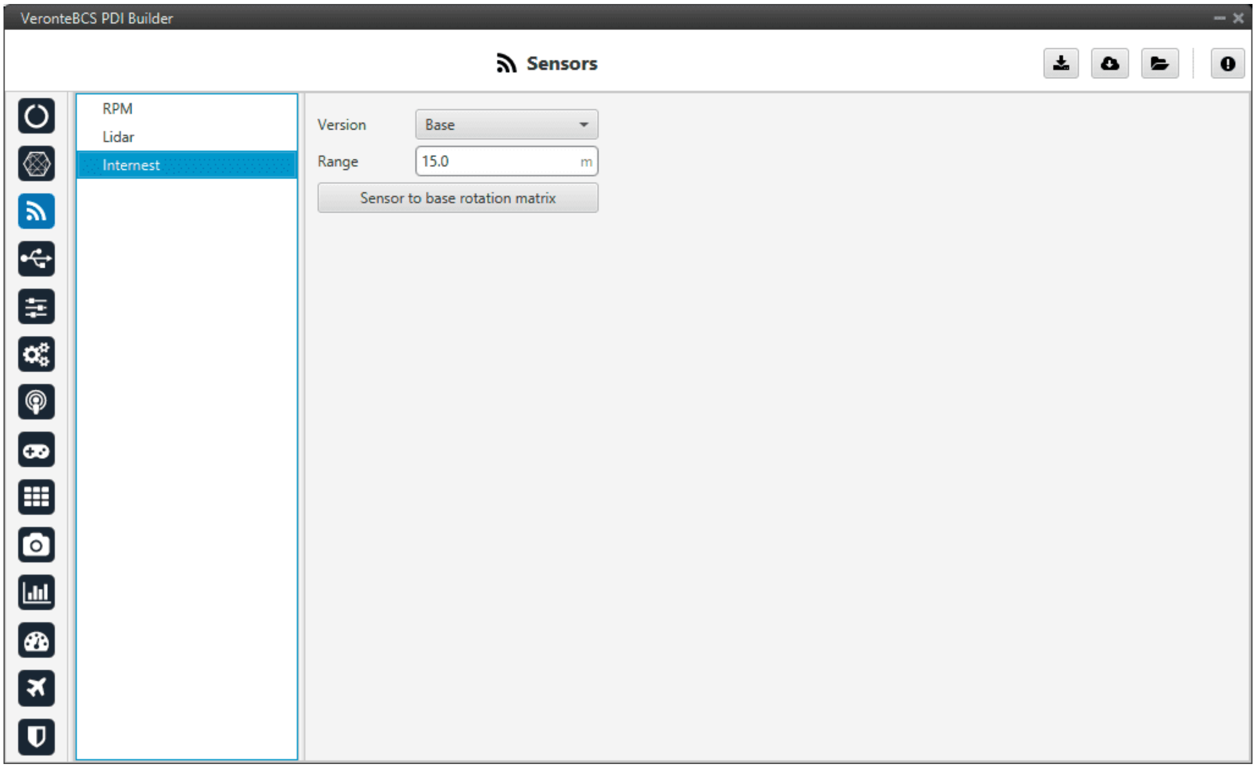
Warning: I2C address will be different for different devices make sure to define it properly by checking the manufacturer documentation.

Note: The lidar number (lidar 1/5) needs to be kept in order to properly configure the altimeter later.

2.3.3 Internest

An **ultrasound sensor** computes **BCS** position by measuring the time that signal sent out takes to return. The following panel together with *Relative position sensor block* allows the user to configure an **Internest system** with **BCS**.

This menu allows the user to choose the version of Internest to be used, its range and the rotation matrix:

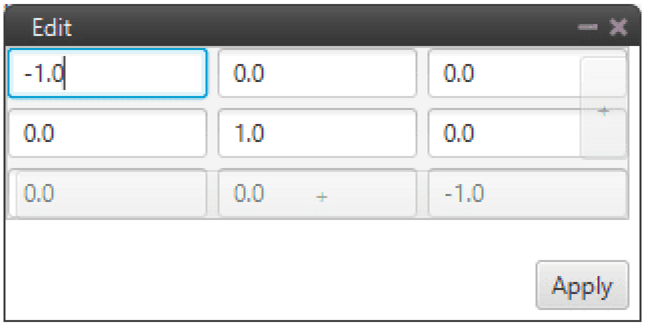


Version: Base

Range: 15.0 m

Fig. 20: **Internest menu**

- **Version**: Users must choose the version of the Internest system, the available options are *Base* and *Explore*.
- **Range**: Defines the distance at which Internest values will start to be valid.
- **Sensor to base**: Matrix to rotate the system to match the **BCS** coordinate system.



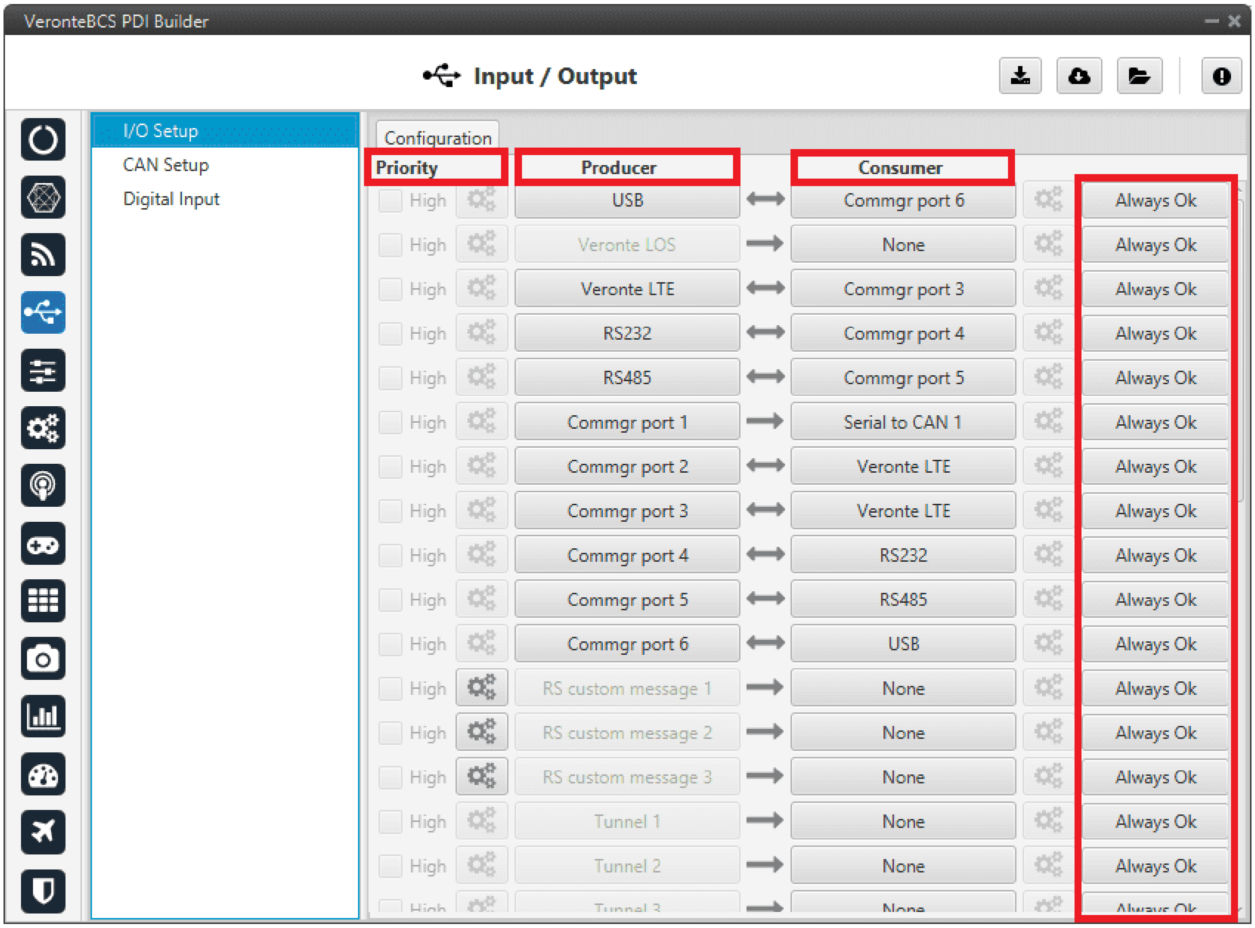
-1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	-1.0

Fig. 21: **Internest - Rotation matrix**

2.4 Input/Output

2.4.1 I/O Setup

In this panel the user can stablish the relationship between a determined signal with a I/O port. This allows users to configure external sensors, messages between **BCS** units (Tunnel) and custom messages.

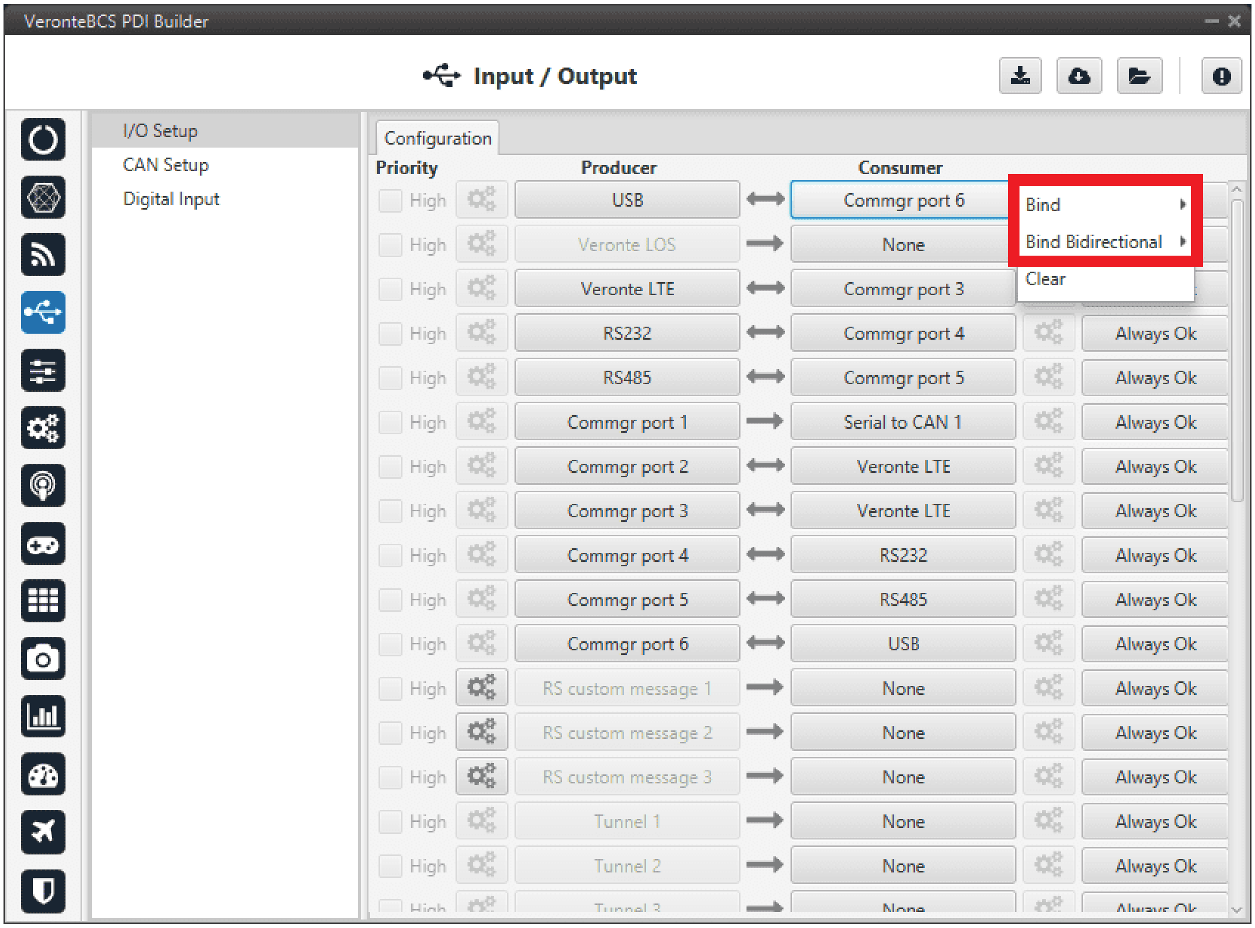


Priority	Producer		Consumer	
□ High	USB	⟷	Commgr port 6	Always Ok
□ High	Veronte LOS	→	None	Always Ok
□ High	Veronte LTE	⟷	Commgr port 3	Always Ok
□ High	RS232	⟷	Commgr port 4	Always Ok
□ High	RS485	⟷	Commgr port 5	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	⟷	Veronte LTE	Always Ok
□ High	Commgr port 3	⟷	Veronte LTE	Always Ok
□ High	Commgr port 4	⟷	RS232	Always Ok
□ High	Commgr port 5	⟷	RS485	Always Ok
□ High	Commgr port 6	⟷	USB	Always Ok
□ High	RS custom message 1	→	None	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	→	None	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 22: **I/O Setup menu**

- **Priority**: Connections between I/O ports can be marked with **high priority** with this checkbox. If enabled, they will run at high frequency: **1000 Hz**.
- **Producer**: Functions for creating and sending messages.
- **Consumer**: Functions for receiving and parsing messages
- **Bit**: This assigns each connection a bit in a way that allows this connection to be activated/deactivated depending on the status of the selected bit.

  By default, the ‘Always Ok’ bit is set to all connections so that they are always active.



Priority	Producer		Consumer	
□ High	USB	↔	Commgr port 6	
□ High	Veronte LOS	→	None	
□ High	Veronte LTE	↔	Commgr port 3	
□ High	RS232	↔	Commgr port 4	Always Ok
□ High	RS485	↔	Commgr port 5	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	↔	Veronte LTE	Always Ok
□ High	Commgr port 3	↔	Veronte LTE	Always Ok
□ High	Commgr port 4	↔	RS232	Always Ok
□ High	Commgr port 5	↔	RS485	Always Ok
□ High	Commgr port 6	↔	USB	Always Ok
□ High	RS custom message 1	→	None	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	→	None	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 23: **I/O Setup consumer options**

Firstly, users have to configure the **Producer** selecting the I/O port or information to use. Later, users have to configure the **Consumer** by clicking on an element, a new window will be displayed to select an item. The relationship between them can be unidirectional (Bind →) or bidirectional (Bind Bidirectional ↔), the last enables a port to receive or send information.

The following I/O ports are available:

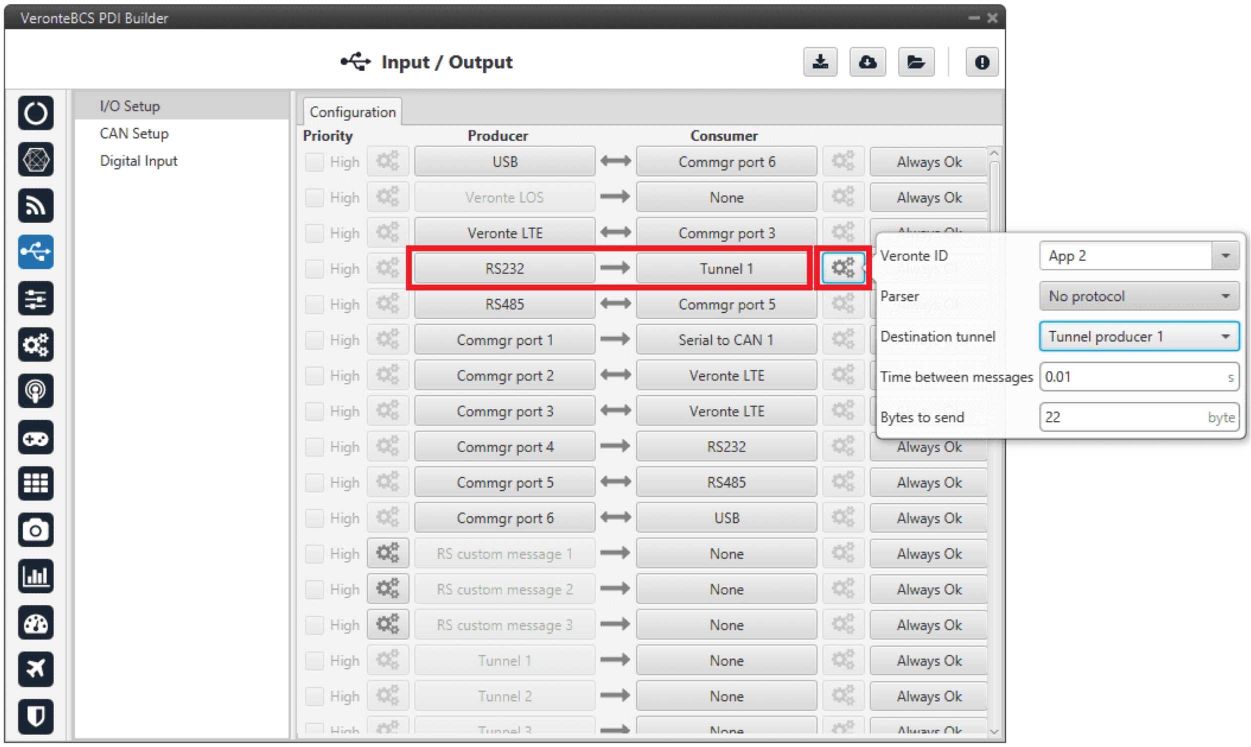
Field	Description
USB	USB Port
Veronte LOS	Radio
Veronte LTE	4G Connection
Veronte LTE Auxiliary	Reserved port
RS232	Serial Port 232
RS485	Serial Port 485
Commgr port	COM Manager ports send and receive VCP messages. This is the protocol used by Veronte products to communicate. For more information on VCP, read the VCP user manual
Tunnel	Creates a bidirectional brigde between two devices, see *Tunnel*
RS Custom Message	This allows user to send/receive a serial custom message, see *Serial Custom Messages*
GPS RTCM	This allows the user to send/receive RTK information from GND unit to AIR unit
Iridium	Iridium communication, see *Iridium section*
Y Splitter	Used to split a signal into 2
NMEA Parser	NMEA 0183 messages parser, see *NMEA Parser*
Unescape port	This allows user to reconstruct a byte stream with an escape logic, see *Unescape port*
CAN to serial / Serial to CAN	**Serial to CAN** sends serial streams over a CAN Bus **/ CAN to serial** undoes the transformation 'Serial to CAN'
CAN wrapper / CAN unwrapper	**CAN wrapper** sends CAN streams over a serial Bus **/ CAN unwrapper** undoes this transformation
External ultrasound	External ultrasound sensor, see *Internest section*

More information about some elements can be found in the following sections.

Tunnel

It is possible to configure a Tunnel which is a bidirectional bridge between **BCS** units that communicate to each other sharing information about an external device connected to the Serial or Digital port.

Let's consider the following image.



Priority	Producer		Consumer	
High	USB	↔	Commgr port 6	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	↔	Commgr port 3	Always Ok
High	RS232	→	Tunnel 1	
High	RS485	↔	Commgr port 5	
High	Commgr port 1	→	Serial to CAN 1	
High	Commgr port 2	↔	Veronte LTE	
High	Commgr port 3	↔	Veronte LTE	
High	Commgr port 4	→	RS232	Always Ok
High	Commgr port 5	↔	RS485	Always Ok
High	Commgr port 6	↔	USB	Always Ok
High	RS custom message 1	→	None	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Veronte ID	App 2
Parser	No protocol
Destination tunnel	Tunnel producer 1
Time between messages	0.01 s
Bytes to send	22 byte

Fig. 24: **Tunnel configuration**

In the image above there is a device connected to the **RS232 (Producer)** and there is a **Tunnel (Consumer)** which sends that information to **Veronte Ops**. In that case the **Producer** will be **Tunnel** and **Consumer** will be the **port or destination tunnel where the device is connected**.

The options available when configuring **Tunnel as Consumer** are:

- **Veronte ID**: Select the address that will receive the information.
  - **App 2**: Veronte Ops address.
  - **Broadcast**: All units on the network. Select this option for a generic configuration.
  - **Veronte v4.X XXXX**: Address of a specific unit, it can be a 1x, a 4x, a CEX, etc.
- **Parser**: The user can choose protocol to parse message data. The options available are:
  - No protocol
  - RTCM3
  - CANserial
- **Destination tunnel**: Number of port is used to avoid mistakes and identify a Tunnel when using more than one, *Tunnel 1, 2 and 3* are available.
- **Time between messages**.
- **Bytes to send**: Sets the message size to send.

When configuring **Tunnel as Producer** (i.e. on the unit that receives the information), no configuration is required. It is only necessary to connect it to a Consumer, usually to a serial port.

Serial Custom Messages

Warning:

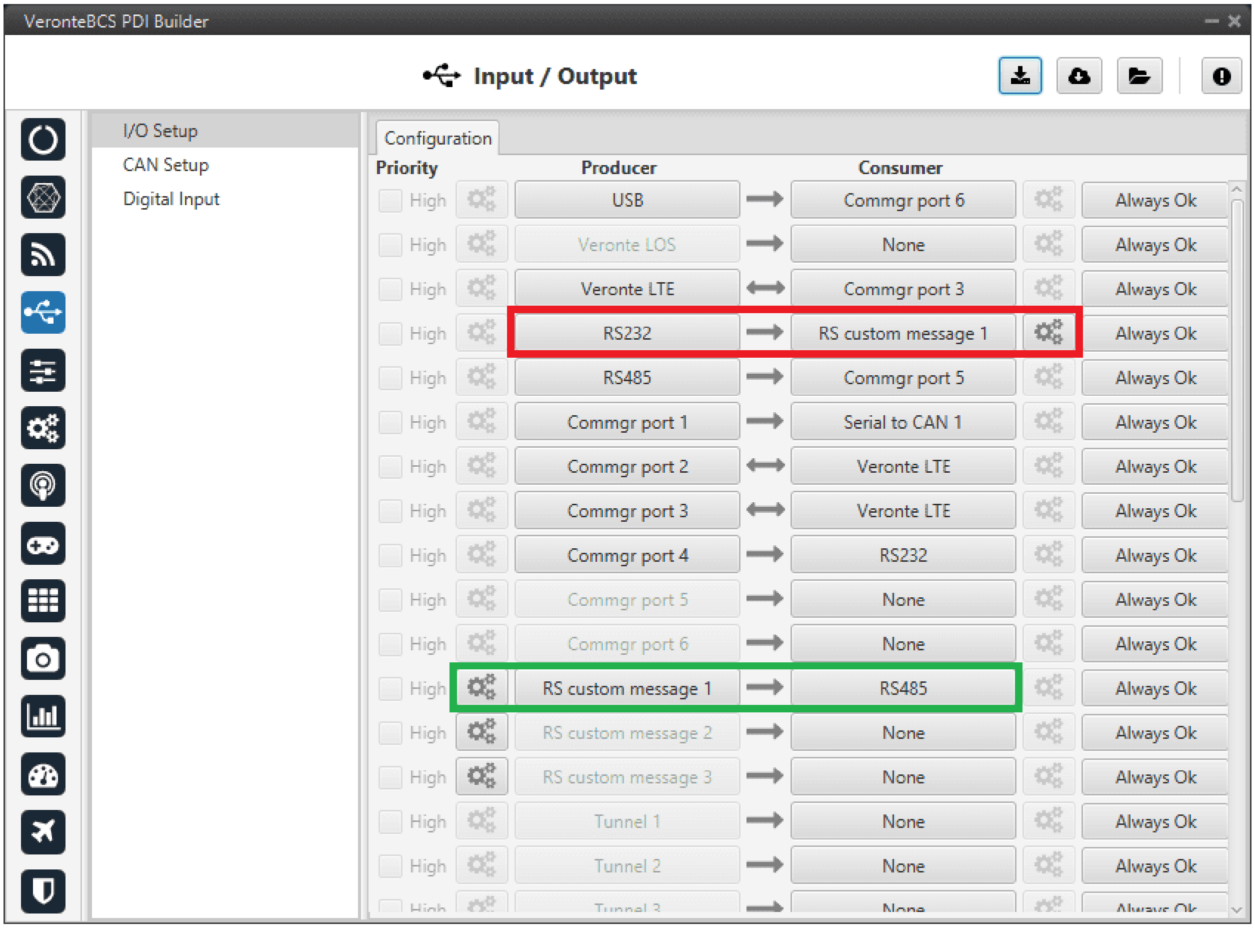
- **BCS** autopilot has a **serial limitation** of **64 vectors** (fieldset) per Custom.

  In addition, there is a limit shared with all Customs, including **CAN Custom Messages**:

  – Maximum number of **vectors** (fieldset): **104**

  – Maximum number of **fields**: **2000**

It is possible to configure the messages sent/received through the serial port and its conversion to system variables by selecting the option **RS Custom message** and configuring the I/O port.



Priority	Producer		Consumer	
□ High	USB	→	Commgr port 6	Always Ok
□ High	Veronte LOS	→	None	Always Ok
□ High	Veronte LTE	↔	Commgr port 3	Always Ok
□ High	RS232	→	RS custom message 1	Always Ok
□ High	RS485	→	Commgr port 5	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	↔	Veronte LTE	Always Ok
□ High	Commgr port 3	↔	Veronte LTE	Always Ok
□ High	Commgr port 4	→	RS232	Always Ok
□ High	Commgr port 5	→	None	Always Ok
□ High	Commgr port 6	→	None	Always Ok
□ High	RS custom message 1	→	RS485	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	→	None	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 25: **Serial Custom Messages**

In the image above can be seen two possible configurations using a RS Custom Message. The 'red' one is configured to receive a determined message from a RS-232 serial port and the 'green' is used to send a RS Custom Message through a RS-485 serial port. It is also possible to use the same RS Custom Message for both tasks if the bidirectional relationship is selected (the arrow indicates this (Bind Bidirectional ↔)).

To configure a RS Custom message, the user must follow the next steps:

1. Press the **configuration button** (⚙ icon) and another window will be displayed. In this window press the "+" icon to add a custom message, the user can choose between **System variables**, **ADSB Vehicle** or **External Sensor**.

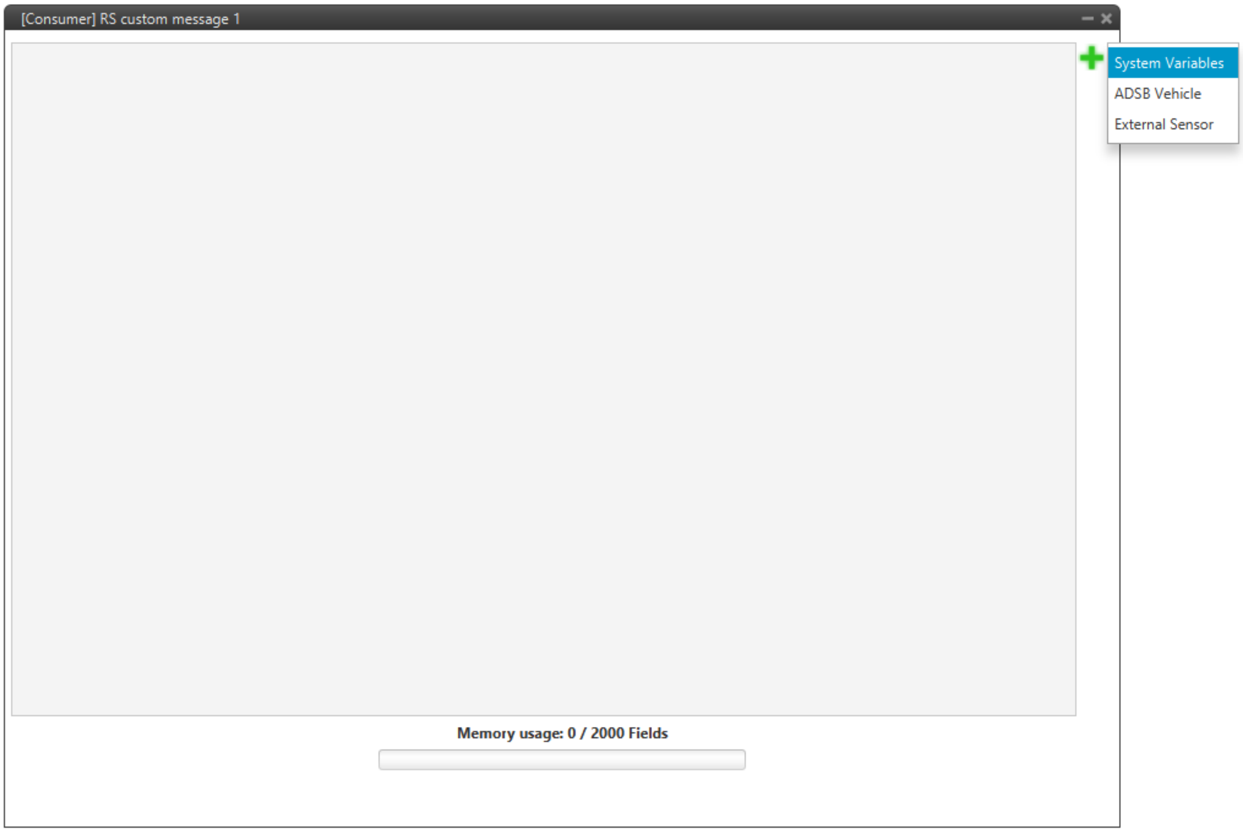


Fig. 26: **Serial Custom Message configuration**

Note: The difference between choosing **System Variables**, **ADSB Vehicle** or **External Sensor** is that when the user selects **Variable** as the custom message type, only system variables will appear when *System Variables* is selected, only ADSB variables when *ADSB Vehicle* is selected and only variables related to external sensors if *External Sensor* is selected.

2. When it is already added, the following options are available to configure a custom message:

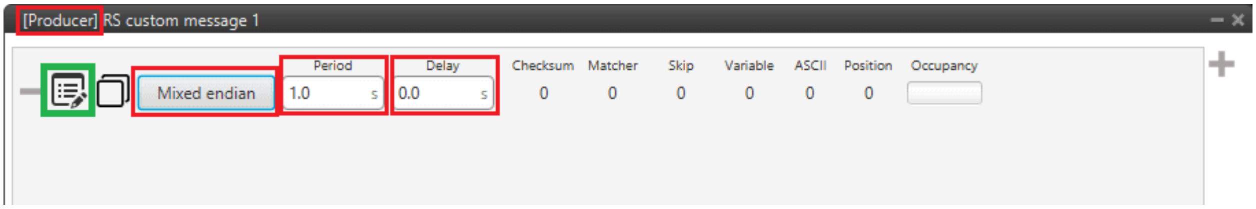
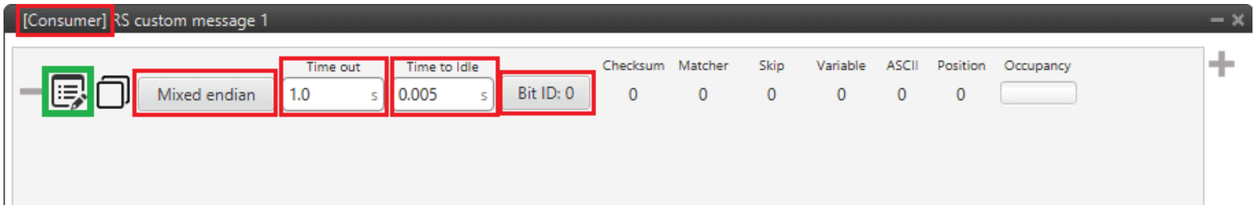


Fig. 27: **Producer RS Custom Message configuration**



[Consumer] RS custom message 1

Mixed endian

Time out: 1.0 s

Time to Idle: 0.005 s

Bit ID: 0

Checksum 0, Matcher 0, Skip 0, Variable 0, ASCII 0, Position 0, Occupancy

Fig. 28: **Consumer RS Custom Message configuration**

- **Endianness**: Depending on the order in which the device issue the message, it is possible to select:
  - **Big endian**: Set the value from left to right.
  - **Little endian**: Set the value from right to left.
  - **Mixed endian**: Some devices use this format. If users need to configure it, please contact the support team (create a ticket in the customer's **Joint Collaboration Framework**; for more information, see Tickets section of the JCF manual).
- **Period/Time out**: This option has a dual role depending on if it is used to transmit or receive data.
  - **Period - Producer**: It is the inverse of the send frequency.
  - **Time out - Consumer**: This is the threshold time between receptions to consider that the message is not being received correctly.
- **Delay/Time to Idle**: This option has a dual role depending on if it is used to transmit or receive data.
  - **Delay - Producer**: It is a delay applied before sending the message. This serves to send messages with the same period without overloading the Serial bus.
  - **Time to Idle - Consumer**: This is the time that **BCS** waits before discarding partially parsed bytes.
- **Bit ID**: This option is only available when a message is configured as **Consumer**. The user bit selected in Bit ID box will be true if the message is being received correctly.

> **Warning:** Pay attention that the user bit selected in **Bit ID** is not in use for another task.

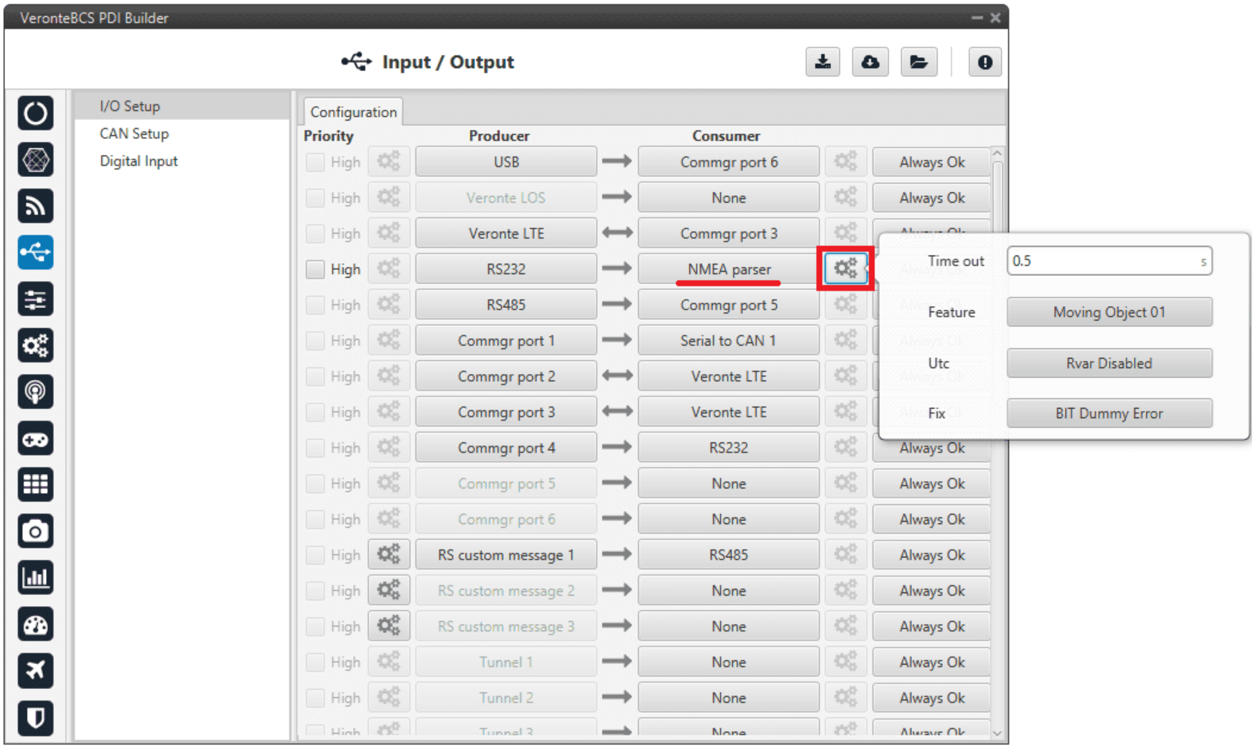
3. To create the structure of the message, click on the **edit message button** (icon) and then press the "**+**" icon to add fields to it. The following type of messages are available to configure a structure: **Variable**, **Checksum**, **Matcher**, **Skip**, **Parse ASCII** and **Position**. The configuration of each structure is covered in *Custom Messages section*.

> **Warning:** Before configuring any message, user has to know the structure it has to have according to the device that is connected to the port. Each device may have a different message structure when it sends or receives information.

To check serial messages transmission, see the *Debug serial messages transmission -> Troubleshooting section* of this manual.

NMEA Parser

NMEA Parser is another way to add an axternal GNSS device. This consumer allows to receive NMEA 0183 messages and parses them directly. The NMEA Parser configuration menu includes the following parameters:



Priority	Producer		Consumer	
□ High	USB	→	Commgr port 6	Always Ok
□ High	Veronte LOS	→	None	Always Ok
□ High	Veronte LTE	↔	Commgr port 3	Always Ok
□ High	RS232	→	NMEA parser	Always Ok
□ High	RS485	→	Commgr port 5	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	↔	Veronte LTE	Always Ok
□ High	Commgr port 3	↔	Veronte LTE	Always Ok
□ High	Commgr port 4	→	RS232	Always Ok
□ High	Commgr port 5	→	None	Always Ok
□ High	Commgr port 6	→	None	Always Ok
□ High	RS custom message 1	→	RS485	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	→	None	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Time out: 0.5 s

Feature: Moving Object 01

Utc: Rvar Disabled

Fix: BIT Dummy Error

Fig. 29: **NMEA Parser configuration**

- **Time out**: Defines the period of incoming information from the external system.
- **Feature**: Variable extracted from the message defining the GNSS position. Usually Moving Object variables are used in **BCS PDI Builder**.
- **Utc**: Variable extracted from the message defining the UTC.
- **Fix**: Data provided by the external device which is important to know the status of the positioning.

Once the NMEA message has been parsed, the variables used for **Fix** and **Feature** can be selected in the GPS External configuration of the GNSS block as **Fix Bit** and **GPS Position**.

Unescape port

To understand what unescape is, the user must first understand what an **escape byte** is.

Let's consider that there is a protocol that defines a '**Flag**' as the start and end byte of the frame. In case the flag or an escape value appears in the frame data, and in order not to misinterpreted the message, an escape byte or the same value repeated will be added before them so that, at the time of parsing, it will be reconstructed with the original byte.

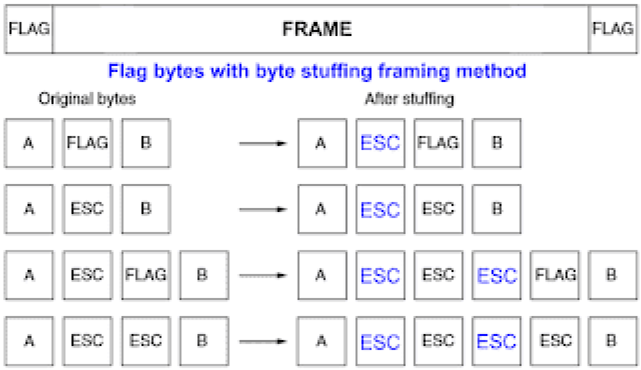
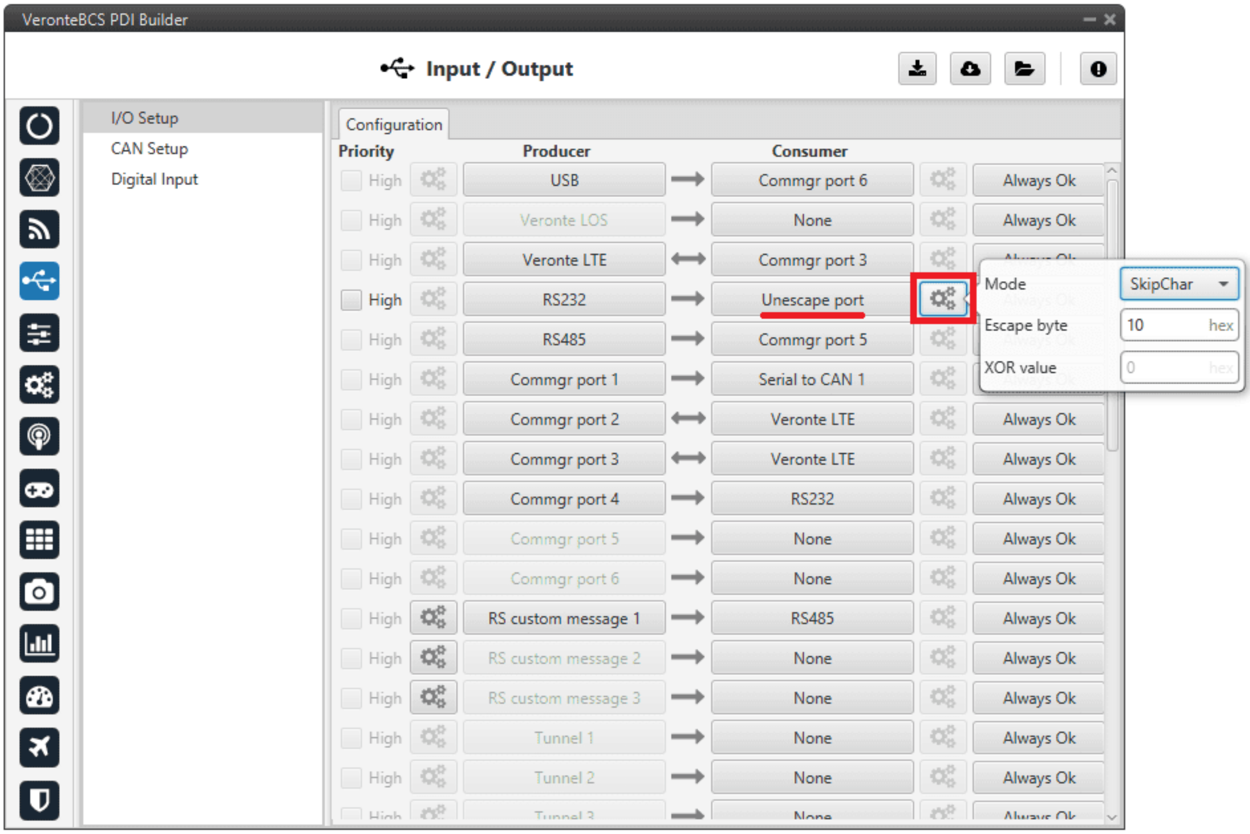

FLAG
FRAME
FLAG
Flag bytes with byte stuffing framing method
Original bytes
After stuffing
A FLAG B → A ESC FLAG B
A ESC B → A ESC ESC B
A ESC FLAG B → A ESC ESC ESC FLAG B
A ESC ESC B → A ESC ESC ESC ESC B

Fig. 30: **Escape byte**

In **BCS PDI Builder**, an Unescape port has been implemented to allows to reconstruct a byte stream with an escape logic.



Priority	Producer		Consumer	
High	USB	→	Commgr port 6	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	↔	Commgr port 3	Always Ok
High	RS232	→	Unescape port	Always Ok
High	RS485	→	Commgr port 5	Always Ok
High	Commgr port 1	→	Serial to CAN 1	Always Ok
High	Commgr port 2	↔	Veronte LTE	Always Ok
High	Commgr port 3	↔	Veronte LTE	Always Ok
High	Commgr port 4	→	RS232	Always Ok
High	Commgr port 5	→	None	Always Ok
High	Commgr port 6	→	None	Always Ok
High	RS custom message 1	→	RS485	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Mode: SkipChar
Escape byte: 10 hex
XOR value: 0 hex

Fig. 31: **Unescape port configuration**

Two modes of escapes are supported:

- **SkipChar**: The unfolding of the value to be escaped, the byte to escape has been repeated in the message.
- **SkipAndXOR**: In this case, the escape byte is entered first and then the value to escape (i.e. the value to escape XOR escape byte).

In addition to this, two more options are available in this pop-up window:

- **Escape byte**: Escape byte added.
- **XOR value**: Only available when '**SkipAndXOR**'option is selected.

2.4.2 CAN Setup

A CAN (Controller Area Network) Bus is a robust vehicle bus standard widely used in the aviation sector. **BCS** is fitted with two CAN buses that can be configured independently.

The structure of a CAN message can be seen in the following image:

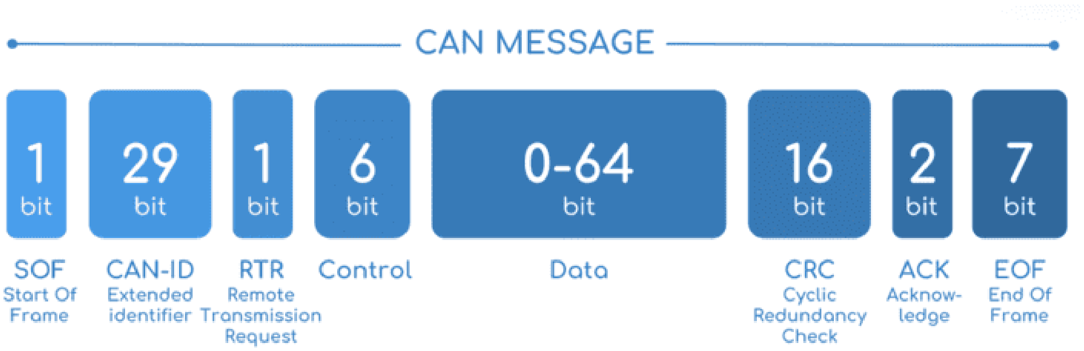


Fig. 32: **CAN message structure**

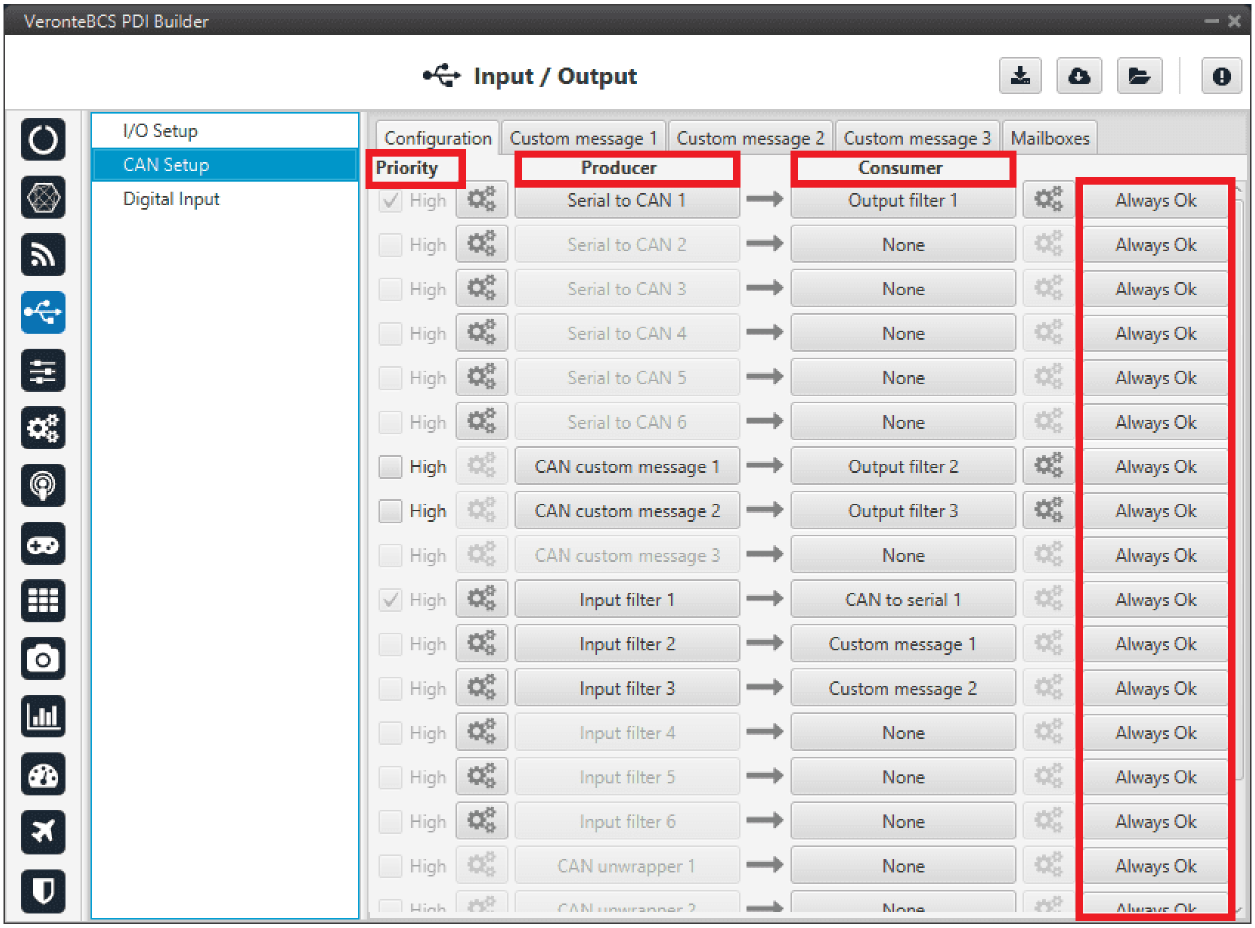
Only the ID is introduced in the system, the rest of the message layout is already coded. The data field is the one build by the user to send, and parsed when received.

The baud rate of both CAN buses can be configured in the *Mailboxes section*.

The steps to be followed from the moment a CAN message arrives at or is sent from the **BCS** are described in the *CAN communication -> Integration examples* section.

2.4.2.1 Configuration

This menu allows the configuration of communications between different devices.



Priority	Producer	Consumer	
☑ High	Serial to CAN 1	Output filter 1	Always Ok
□ High	Serial to CAN 2	None	Always Ok
□ High	Serial to CAN 3	None	Always Ok
□ High	Serial to CAN 4	None	Always Ok
□ High	Serial to CAN 5	None	Always Ok
□ High	Serial to CAN 6	None	Always Ok
□ High	CAN custom message 1	Output filter 2	Always Ok
□ High	CAN custom message 2	Output filter 3	Always Ok
□ High	CAN custom message 3	None	Always Ok
☑ High	Input filter 1	CAN to serial 1	Always Ok
□ High	Input filter 2	Custom message 1	Always Ok
□ High	Input filter 3	Custom message 2	Always Ok
□ High	Input filter 4	None	Always Ok
□ High	Input filter 5	None	Always Ok
□ High	Input filter 6	None	Always Ok
□ High	CAN unwrapper 1	None	Always Ok
□ High	CAN unwrapper 2	None	Always Ok

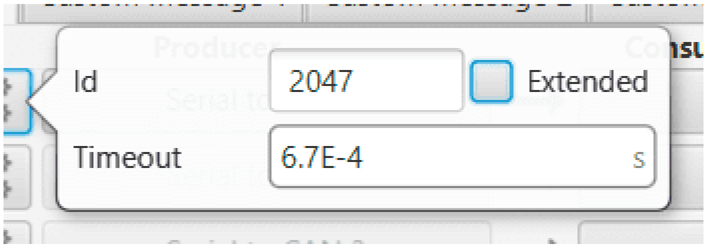
Fig. 33: **CAN configuration section**

In this menu, the user can find the same 'columns' (**Priority**, **Producer**, **Consumer** and **Bit**) as in the *I/O Setup menu*.

> **Warning:** In CAN, in Low state the specified period is not guaranteed but in High state it is.
>
> However, only those **messages that are critical for external devices** should be set as **high priority**, as this may disrupt the proper functioning of **Veronte BCS**.

On the one hand, **BCS** has the **producers** shown below:

- **Serial to CAN**: Serial messages over CAN output, it has to be connected to *I/O Setup* **consumer**. It can be configured in the **configuration button** (icon), a pop-up window will appear:



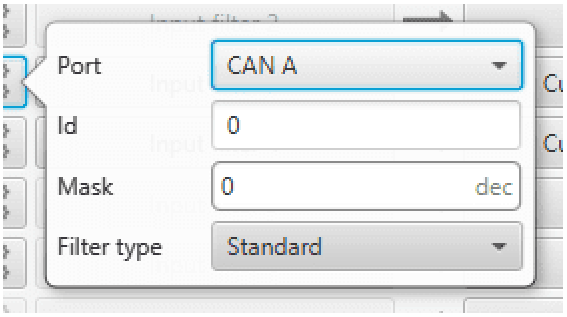
Id 2047 □ Extended

Timeout 6.7E-4 s

Fig. 34: **Serial to CAN configuration**

- **Id**: CAN Id must be set and it is used to identify messages. The value set has to be **decimal format**.
- **Extended**: If enabled, the frame format will be this, 'Extended', i.e. with a 29-bit identifier. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
- **Time out**: This is the threshold time between receptions to consider that it is not being received correctly.

- **CAN custom message**: CAN custom messages transmission. They are configured in the next section, see *Custom Messages*.
- **Input filter**: CAN input filters. Those CAN messages received in one filter can no longer be received in subsequent filters. Input filter must be configured in the **configuration button** (⚙ icon), a pop-up window will appear:



Port CAN A

Id 0

Mask 0 dec

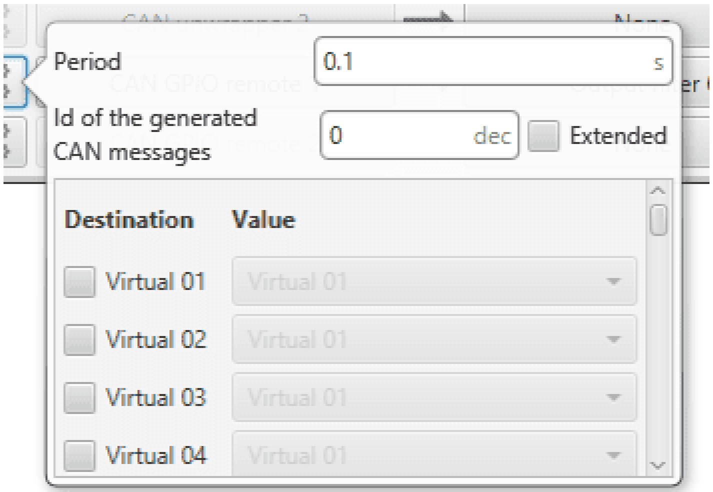
Filter type Standard

Fig. 35: **Input filter configuration**

- **Port**: It is required to configure the CAN bus from which it listens, the user can choose between *CAN A*, *CAN B* or *BOTH*.
- **Id**: CAN Id must be set and it is used to identify messages. The value set has to be **decimal format**.
- **Mask**: Here a CAN Id mask can be set to filter messages. The mask defines the bits that should match.
- **Filter type**: The options available are *Standard*, *Extended* and *Both*.

Attention: Make sure that the mask is set properly to be able to receive the desired CAN messages.

- **CAN unwrapper**: This undoes the 'CAN wrapper' action, it has to be connected to *I/O Setup* **consumer**.
- **CAN GPIO remote**: CAN messages to GPIO peripherals such as CEX. It can be configured in the **configuration button** (⚙ icon), a pop-up window will appear:



Period 0.1 s

Id of the generated CAN messages 0 dec □ Extended

Destination	Value
□ Virtual 01	Virtual 01
□ Virtual 02	Virtual 01
□ Virtual 03	Virtual 01
□ Virtual 04	Virtual 01

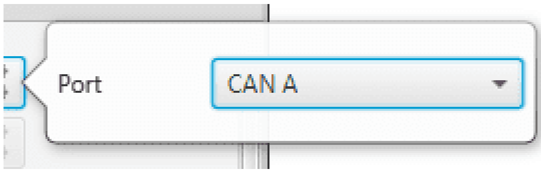
Fig. 36: **CAN GPIO remote configuration**

- **Period**: It is the period of sending messages.
- **Id of the generated CAN message**: CAN Id must be set and it is used to identify messages. The value set has to be **decimal format**.
- **Extended**: If enabled, the frame format will be this, 'Extended', i.e. with a 29-bit identifier. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
- **Destination**: Here the user select the destination CEX pins.
- **Value**: The user must select the **BCS** pin to be connected to the CEX pin.

- **CAN 4x**: CAN message transmission already configured for correct **communication between the 1x autopilots within the Veronte Autopilot 4x**. If the user has any questions, please contact the support team following the Joint Collaboration Framework.

On the other hand, the **consumers** are the following:

- **CAN to serial**: This undoes the 'Serial to CAN' action, it has to be connected to *I/O Setup* **producer**.
- **Custom message**: CAN custom messages reception. They are configured in the next section, see *Custom Messages*.
- **Output filter**: CAN output filters. The user can choose between *CAN A*, *CAN B* or *BOTH* in the **configuration button** (⚙ icon).



Port CAN A

Fig. 37: **Output filter configuration**

- **CAN wrapper**: CAN messages over serial output, it has to be connected to *I/O Setup* **producer**.
- **CAN 4x**: CAN message reception already configured for correct **communication between the 1x autopilots within the Veronte Autopilot 4x**. If the user has any questions, please contact the support team following the Joint Collaboration Framework.

2.4.2.2 Custom Messages

In the custom message tabs (there are 3 available), the user chooses the variables to be sent/received over the CAN buses. The following elements can be configured:

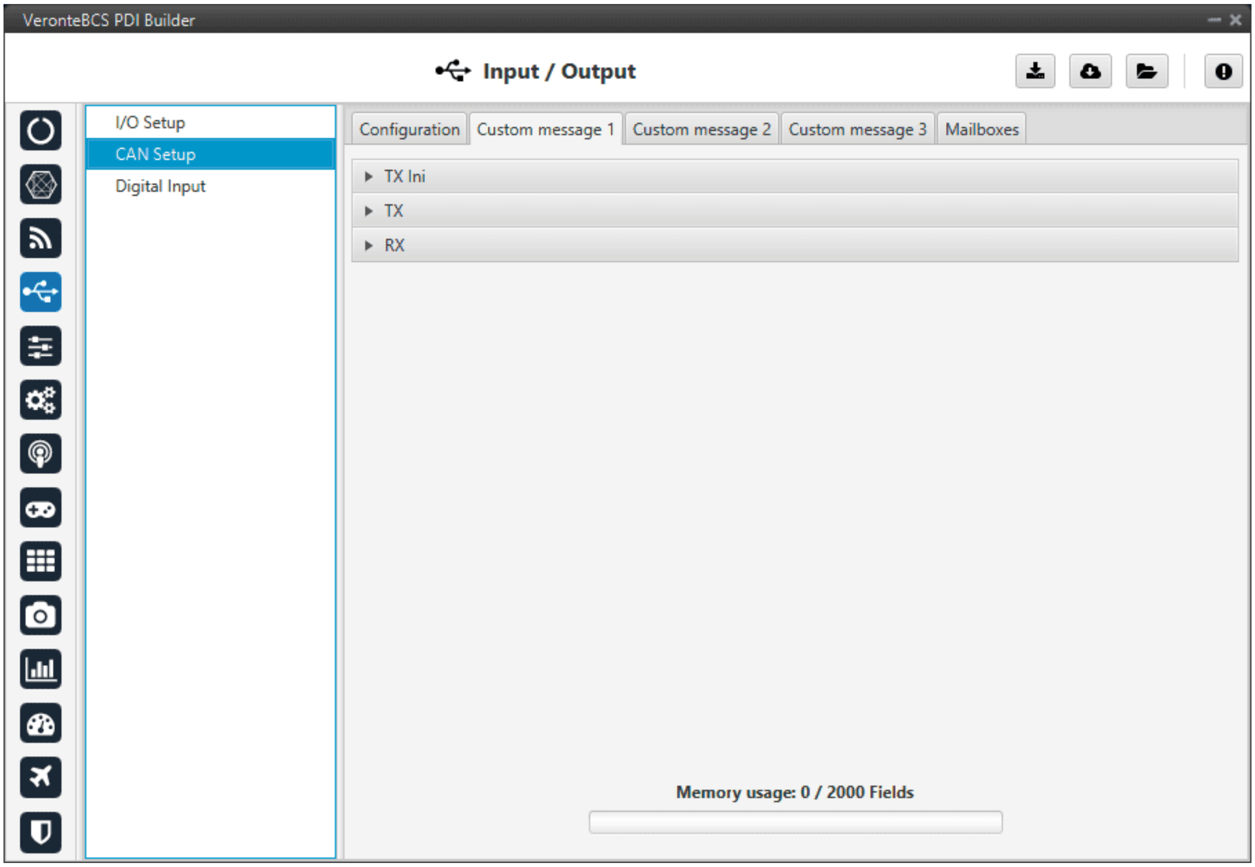
- **TX Ini**: Used to configure transmitted messages that are only sent once at the beginning of the operation (sent when the autopilot boots up). They can be used to initialize some devices.
- **TX**: Used to configure transmitted messages.
- **RX**: Used to configure the reception messages (where they are stored).

Warning:

- The **maximum capacity** of a **CAN message** is **64 bits** (8 bytes), so to send more information it must be divided into several messages.
- **BCS** has a **CAN limitation** of **40 TX** messages per Custom, **40 TX Ini** messages per Custom and **80 RX** messages per Custom.

  In addition, there is a limit shared with all Customs, including **RS Custom Messages**:

  - Maximum number of **vectors** (fieldset): **104**
  - Maximum number of **fields**: **2000**



Memory usage: 0 / 2000 Fields

Fig. 38: **CAN Custom Message section**

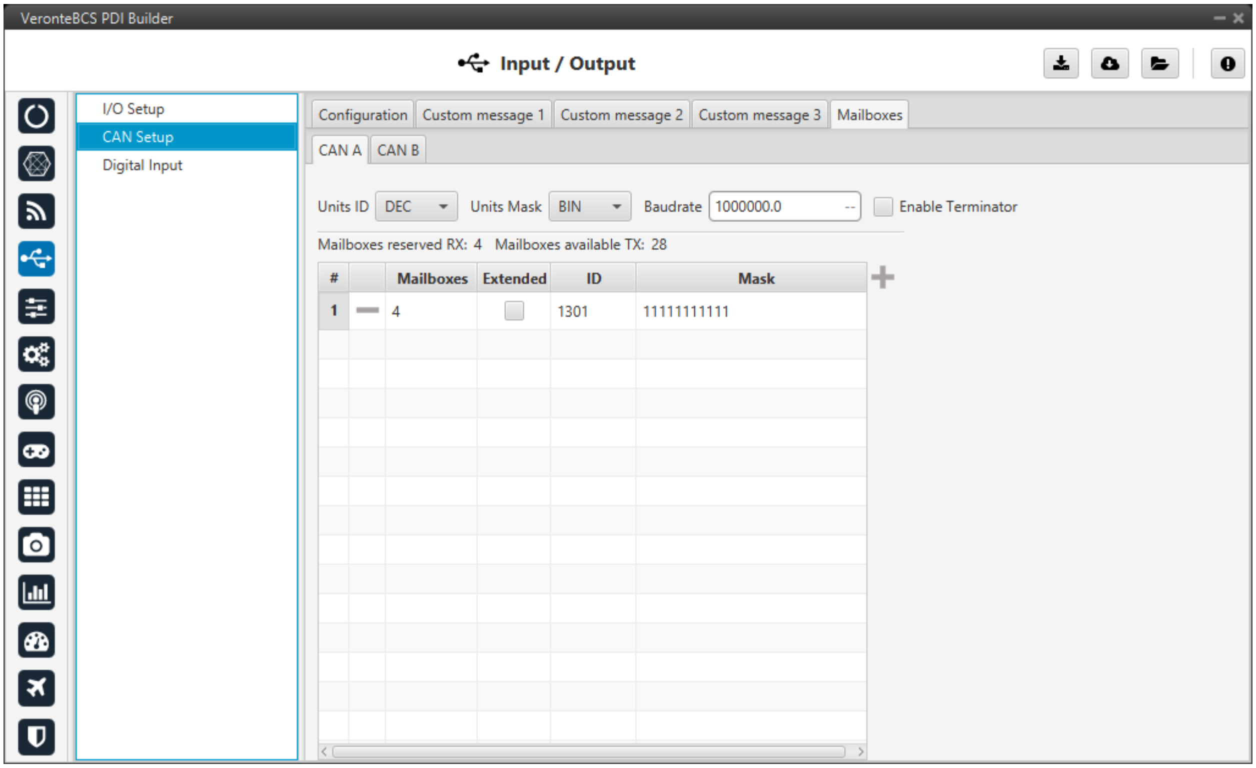
Since this section works in a similar way to the CAN Custom Message configuration in the 1x PDI Builder software, the explanation to configure the telemetry messages via CAN can be found in the CAN Setup -> Input/Output section of the **1x PDI Builder user manual**.

2.4.2.3 Mailboxes

Here the user can modify the mailboxes from the selected CAN bus (CAN A or CAN B).

When **BCS** is going to receive data on the CAN Bus, it is mandatory to configure a certain number of mailboxes. However, it is also necessary to have at least 1 mailbox for transmission (TX).

In order to add a mailbox, press the “**+**” icon.



Units ID DEC Units Mask BIN Baudrate 1000000.0 ☐ Enable Terminator

Mailboxes reserved RX: 4 Mailboxes available TX: 28

#		Mailboxes	Extended	ID	Mask
1	—	4	☐	1301	11111111111

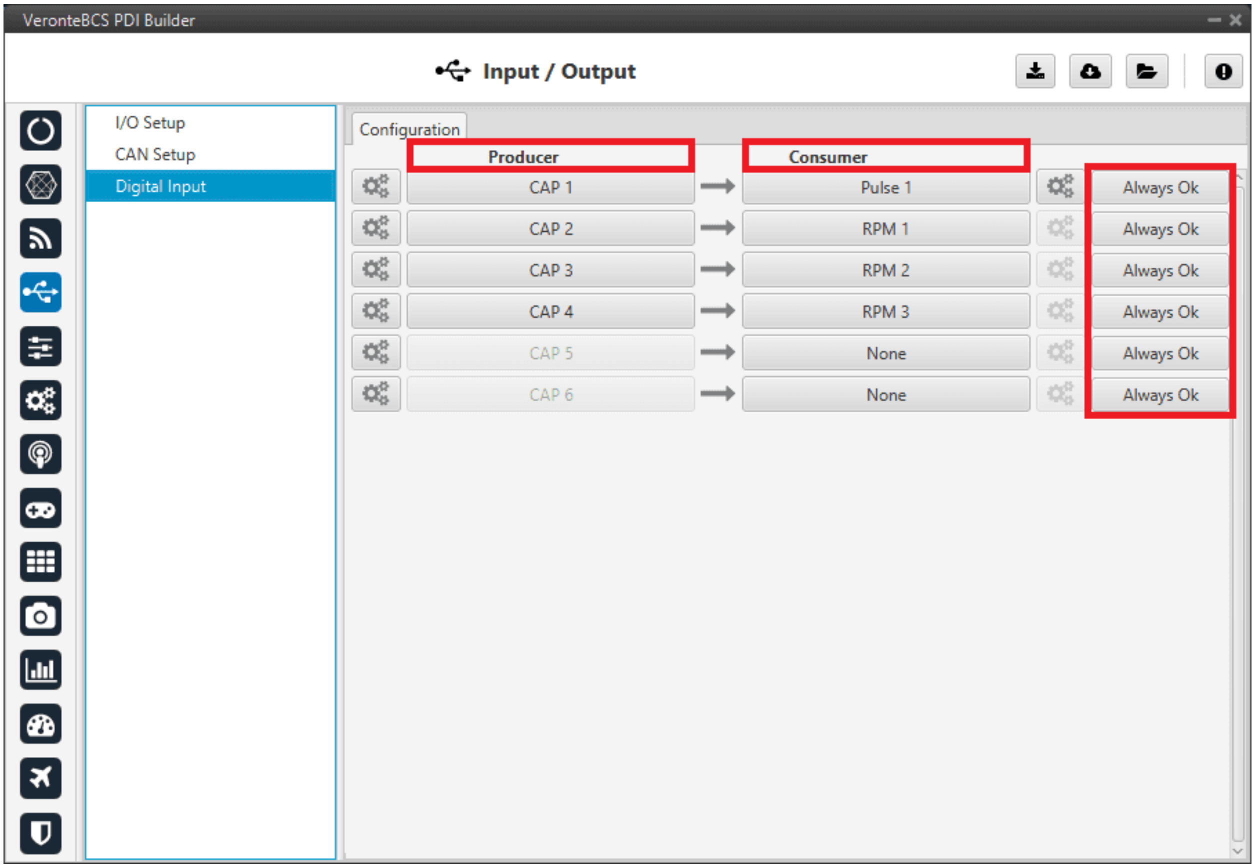
Fig. 39: **Mailboxes section**

More information about Mailboxes can be found in the Mailboxes section of the **1x PDI Builder** user manual.

2.4.3 Digital Input

GPIO pins can work as Digital Input or Output as well as PWM. In order to configure some custom sensors such as a Stick PPM, Pulse sensor or RPM sensor pins are reserved, which correspond to pins 55-58. These pins can also be used as Digital I/O.

Sensors using a Digital Input are configured in this menu.



Producer		Consumer	
CAP 1	→	Pulse 1	Always Ok
CAP 2	→	RPM 1	Always Ok
CAP 3	→	RPM 2	Always Ok
CAP 4	→	RPM 3	Always Ok
CAP 5	→	None	Always Ok
CAP 6	→	None	Always Ok

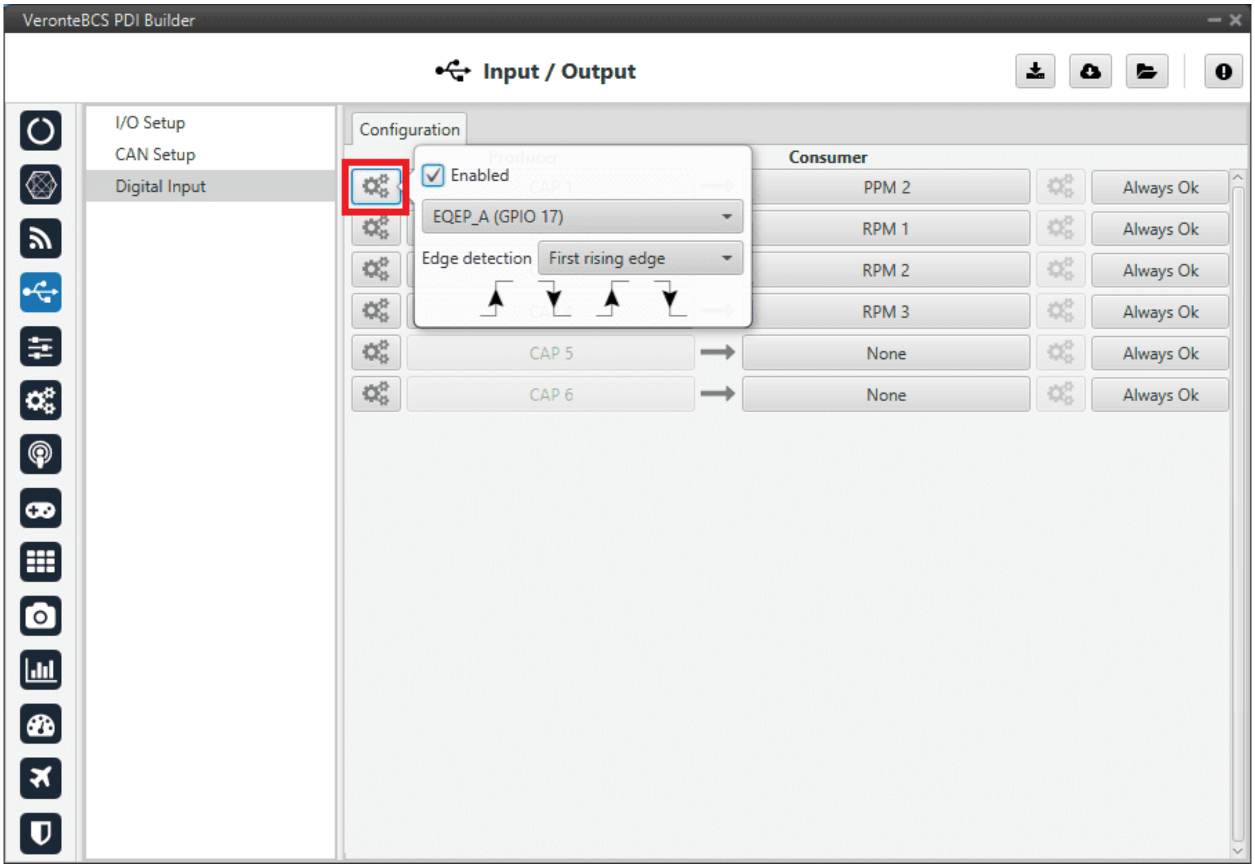
Fig. 40: **Digital Input section**

In addition, in this menu the user can also find the same ‘columns’ (Producer, Consumer and Bit) as in the *I/O Setup menu*.

The process to configure a device can be done as follows:

1. Select and configure a **Producer**. There are 6 possible producers: CAP 1 - 6.

   Press on the configuration button (⚙ icon) and a new pop-up window will show.



☑ Enabled

EQEP_A (GPIO 17)

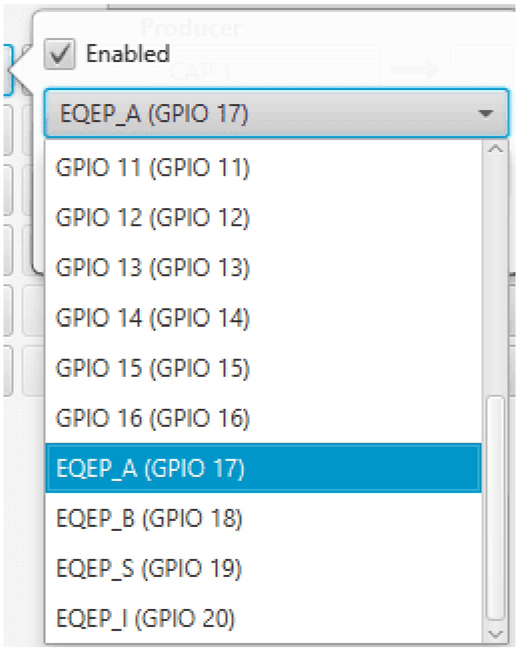
Edge detection: First rising edge

Producer		Consumer	
CAP 1	→	PPM 2	Always Ok
		RPM 1	Always Ok
		RPM 2	Always Ok
		RPM 3	Always Ok
CAP 5	→	None	Always Ok
CAP 6	→	None	Always Ok

Fig. 41: **Digital Input - Producer**

In the pop-up window, users can check:

- That this producer is **enabled**.
- Which **pin** this CAP is associated and, therefore, to which device is connected. It is possible to select these pins. Pins available are GPIO 1 to 16, and EQEP A, B, S and I. When using the harness provided by Embention the transmitter Digital Input is connected to the pin 55 (EQEP_A) with pin 49 as Ground.



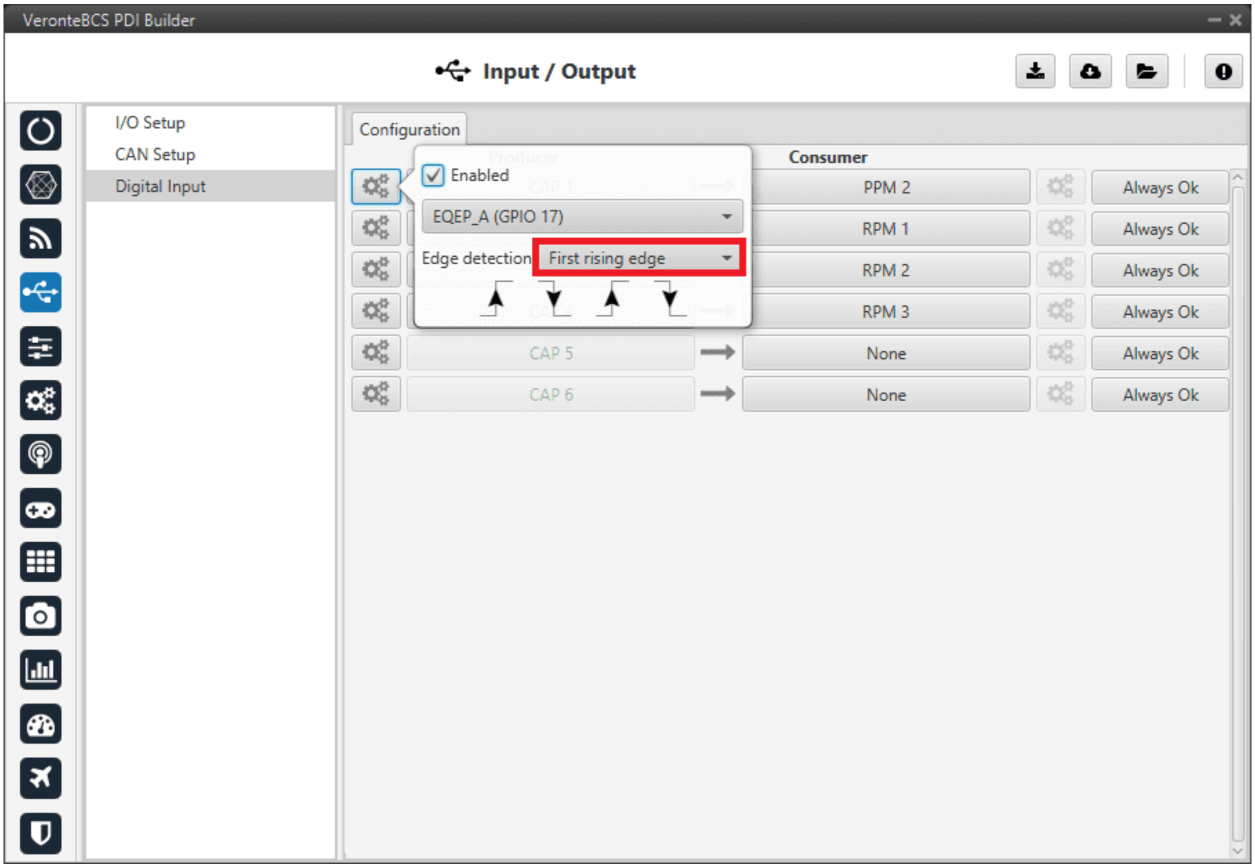
☑ Enabled

EQEP_A (GPIO 17)

Fig. 42: **Digital Input - CAP**

- How the pulses are read and transformed into a digital signal (how they are processed). That can be configured with the **Edge detection** option.

  By clicking on the drop-down menu, the following options can be selected:



☑ Enabled

EQEP_A (GPIO 17)

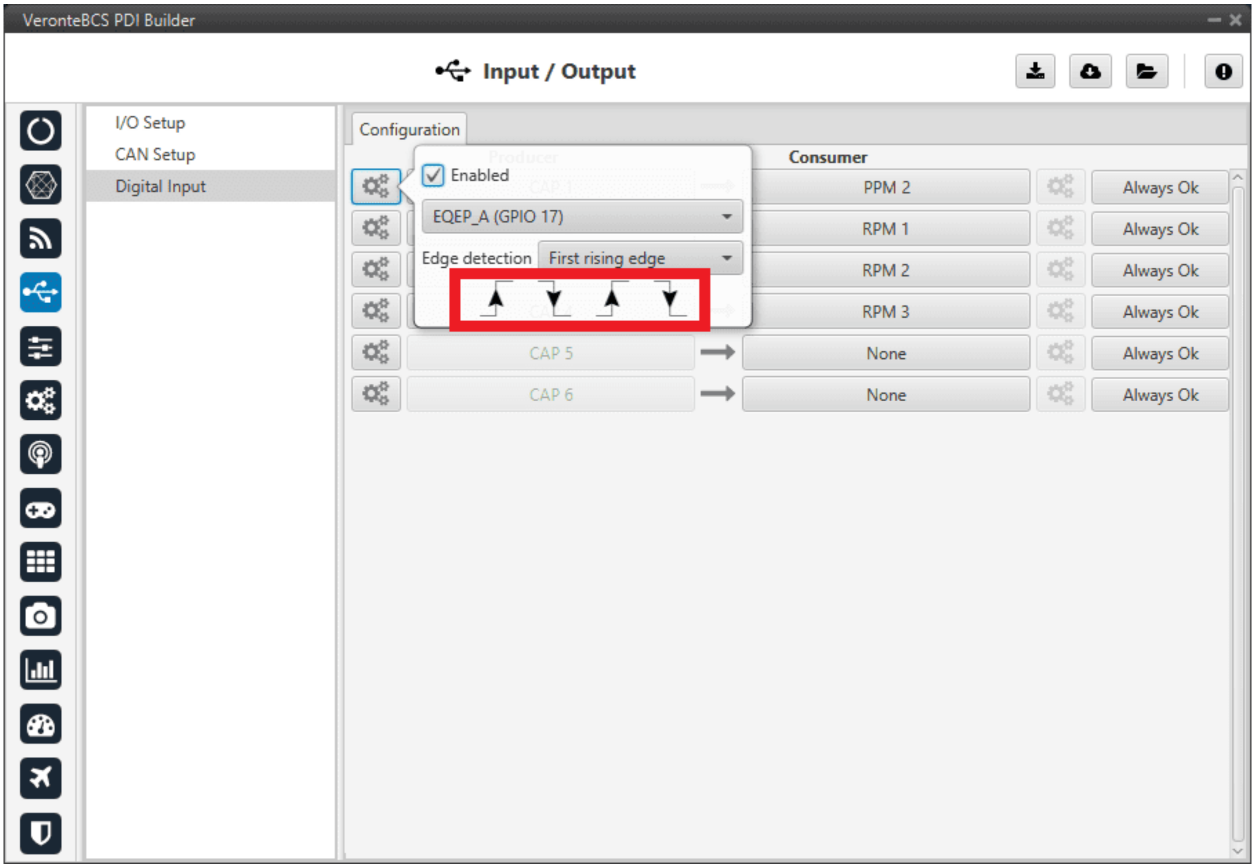
Edge detection: First rising edge

Producer	Consumer	
CAP 1	PPM 2	Always Ok
	RPM 1	Always Ok
	RPM 2	Always Ok
CAP 4	RPM 3	Always Ok
CAP 5	None	Always Ok
CAP 6	None	Always Ok

Fig. 43: **Digital Input - Edge detection option**

- **First rising edge**: With this option, when the **rise** of the pulse is detected, the data will start to be stored. Recommended when consumer is **PPM** or **Pulse**.
- **First falling edge**: With this option, when the **fall** of the pulse is detected, the data will start to be stored.

Note: By clicking on the arrows, it can also be configured in a specific way.



☑ Enabled

EQEP_A (GPIO 17)

Edge detection: First rising edge

Producer		Consumer		
CAP 1	→	PPM 2		Always Ok
		RPM 1		Always Ok
		RPM 2		Always Ok
		RPM 3		Always Ok
CAP 5	→	None		Always Ok
CAP 6	→	None		Always Ok

Fig. 44: **Digital Input - Edge detection option**

2. Click on the **Bind** button to select the type of **Consumer**, it is possible to choose among a PPM 1-4 (Stick PPM), RPM 1-6 (RPM sensor) or Pulse 1-4 (Pulse).

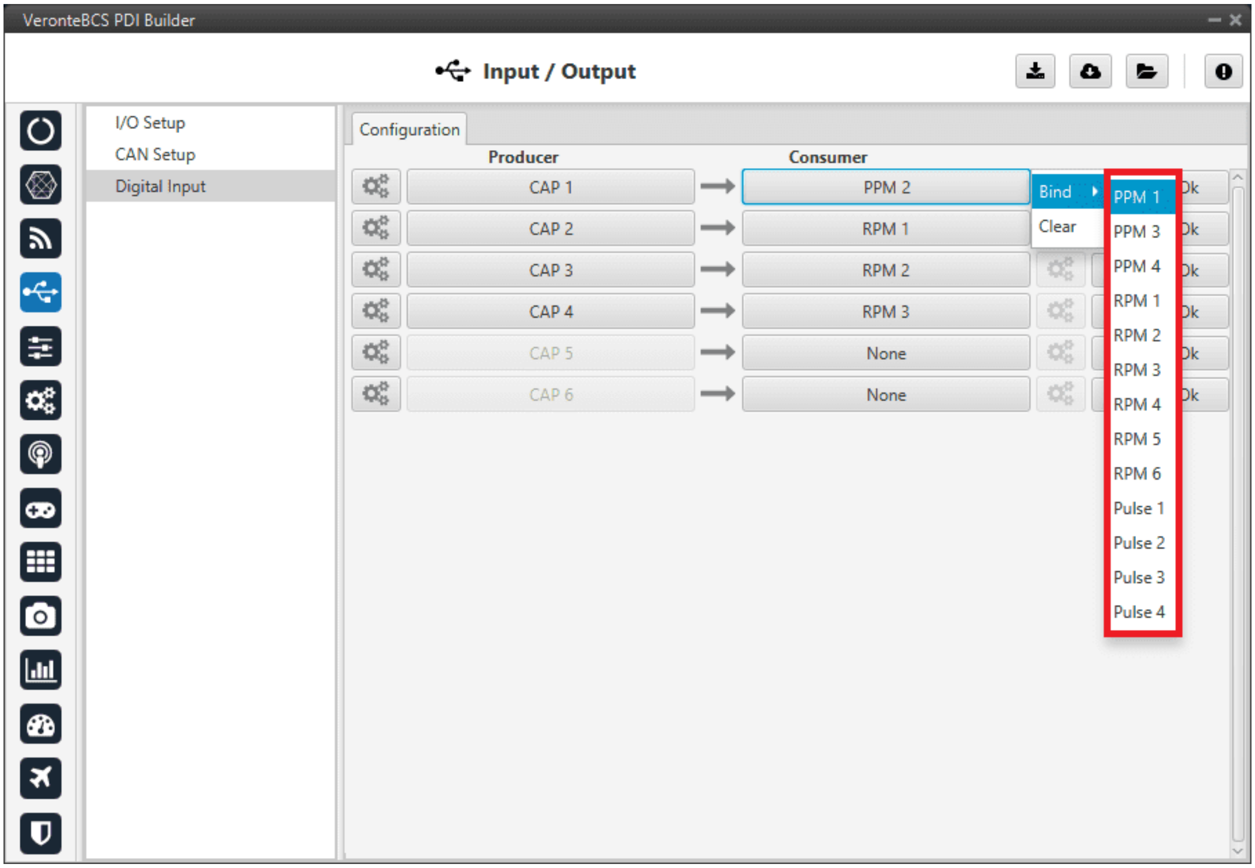
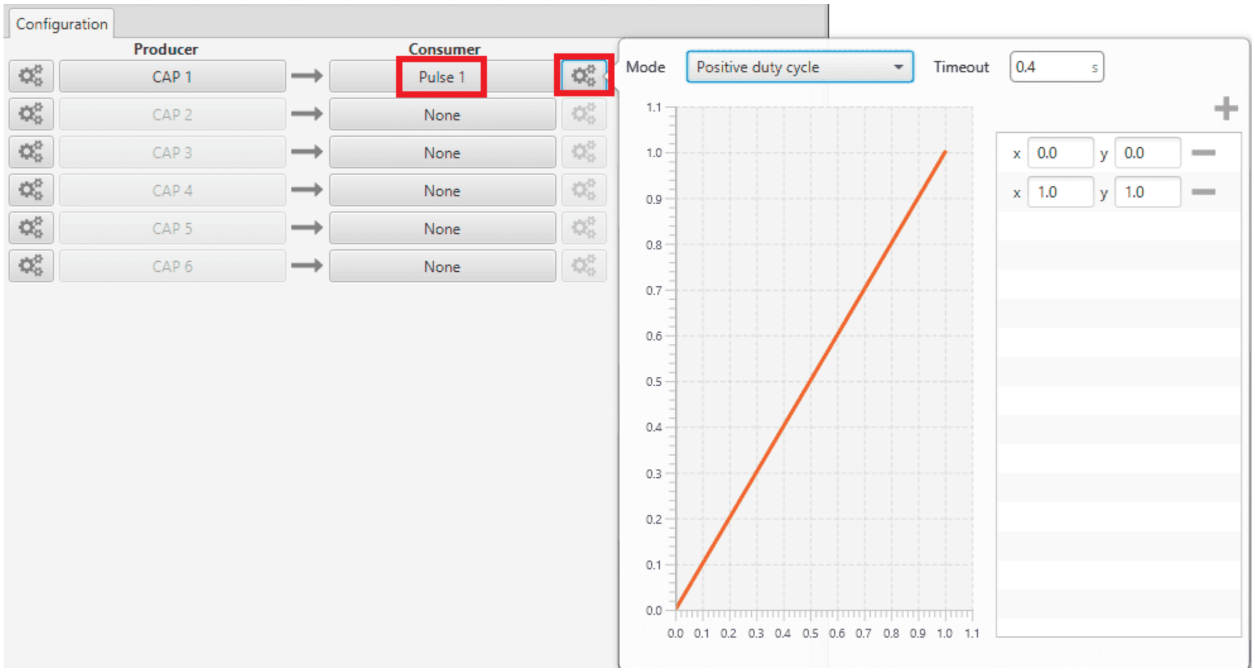


Fig. 45: **Digital Input - Consumer**

- **PPM 1-4** selected: PPM is configured in the *Stick menu*.
- **RPM 1-6** selected: The variables in which the information read here is stored are '**RPM 1-6**'. For more information on the configuration of RPM, see the *RPM section*.
- **Pulse 1-4** selected: The variables in which the information read here is stored are '**Captured pulse 1-4**'.

  It is possible to configure it clicking on the **configuration button** (⚙ icon):



Configuration

Producer		Consumer
CAP 1	→	Pulse 1
CAP 2	→	None
CAP 3	→	None
CAP 4	→	None
CAP 5	→	None
CAP 6	→	None

Mode: Positive duty cycle

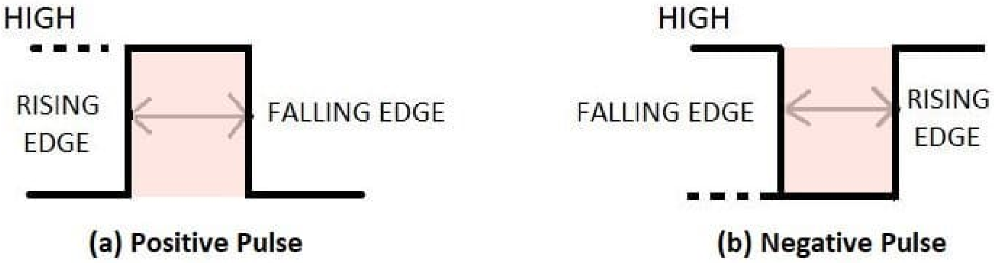
Timeout: 0.4 s

x	0.0	y	0.0
x	1.0	y	1.0

Fig. 46: **Digital Input - Pulse**

In the pop-up window, users will find the following options for configuration:

- **Mode**:
  - **Positive pulse duration**: The period of the pulse is obtained. It takes the time in ‘High’ state.
  - **Negative pulse duration**: The period of the pulse is obtained. It takes the time in ‘Low’ state.



(a) Positive Pulse

(b) Negative Pulse

Fig. 47: **Positive/Negative pulse duration**

  - **Positive duty cycle**: The duty cycle. It takes the time in ‘High’ state.
  - **Negative duty cycle**: The duty cycle. It takes the time in ‘Low’ state.

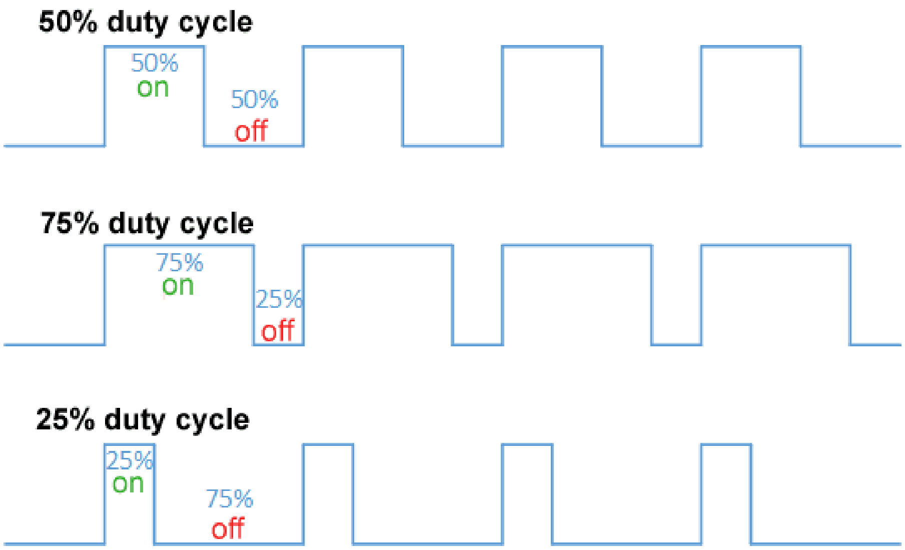


Fig. 48: **Positive/Negative duty cycle**

- **Time out**: This defines the time to consider that no signal is received.
- **Function**: Here the user can customise a function to handle the values. Normally, a function is set with the points [0,0] and [1,1], so no transformation is applied, input = output. However, the user can configure it as desired.

Example

Let's imagine that **First rising edge** has been selected as the edge detection option in Producer and the pulse that **BCS** has to read is a square signal with a period of 2 seconds and a duty cycle of 25% (see image below).

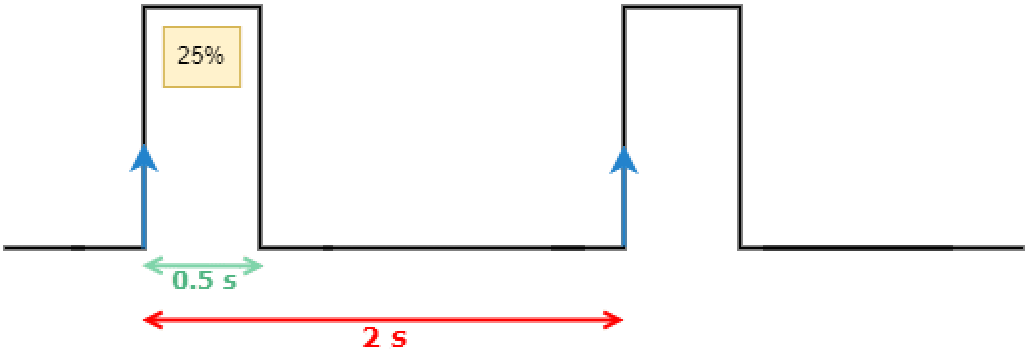


Fig. 49: **Signal generated**

On the other hand, if **Positive pulse duration** is selected as Consumer and it is configured as in the previous image (*Digital Input - Pulse*), the value obtained in the variable **Captured pulse** (*Captured pulse 1* in the following example) will be **0.50s**, this is because it is the period of the "Positive pulse" of that pulse.

However, if **Positive duty cycle** is selected as Consumer, the value obtained in the variable **Captured pulse** (*Captured pulse 2* in the following example) will be **0.25**, this is because it is the positive duty cycle of that pulse.

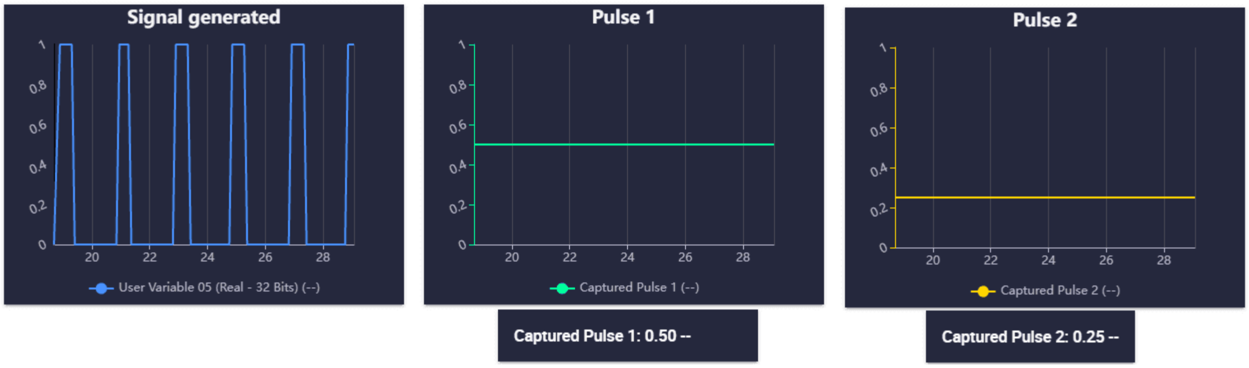


Fig. 50: **Digital Input example**

2.5 Control

In this panel all the parameters related to the control of the platform can be found.

2.5.1 Phases

This section creates (it defines but does not configures) the flight phases that will control the aircraft at different stages of the operation.

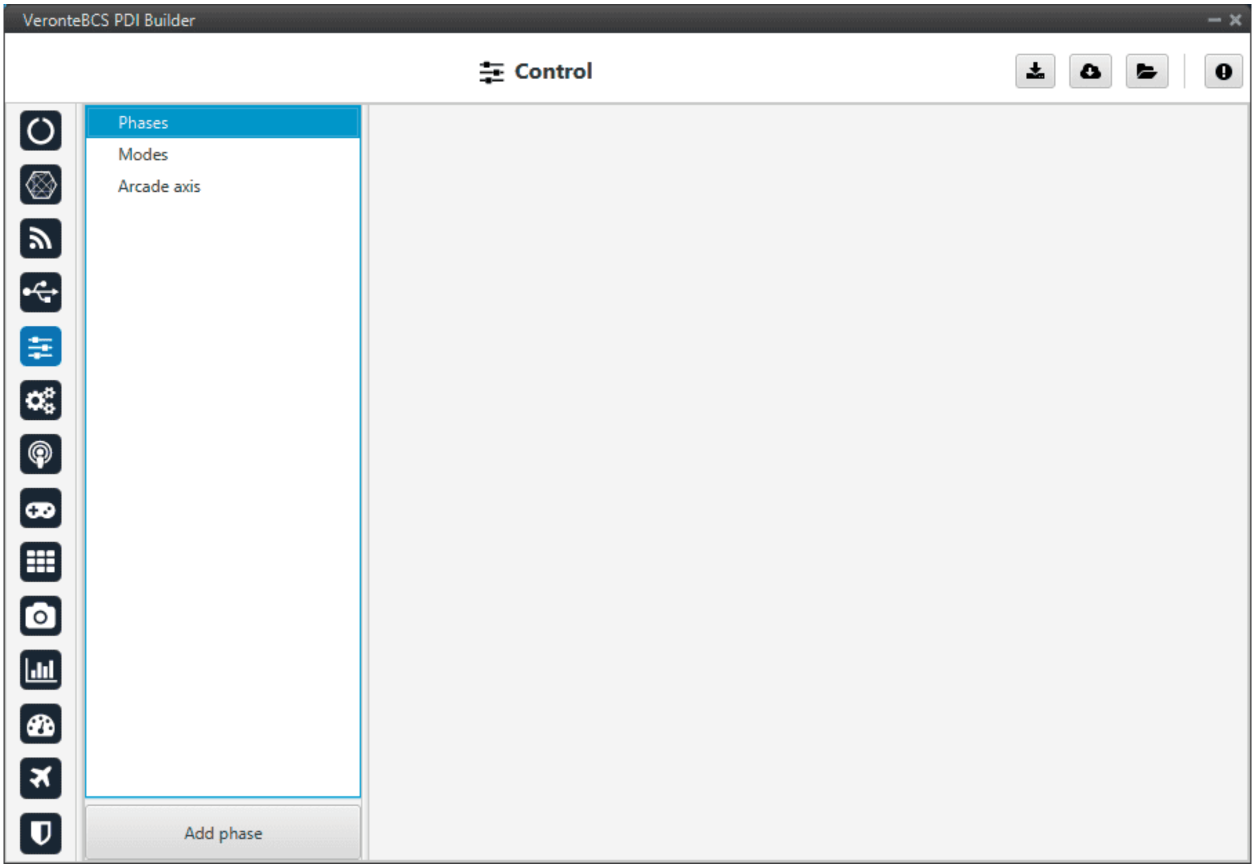


Fig. 51: **Phases menu**

To create a new phase click on **Add phase**, the user can then select a phase already created or create a new one.

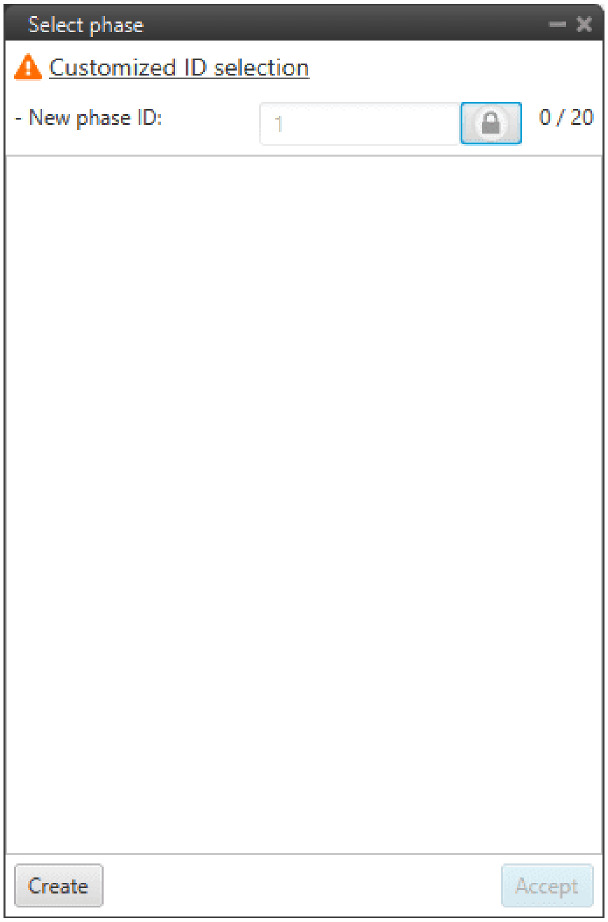


Fig. 52: **Create phase**

In addition, by *right clicking* on the phase, the user can rename, copy or remove it:

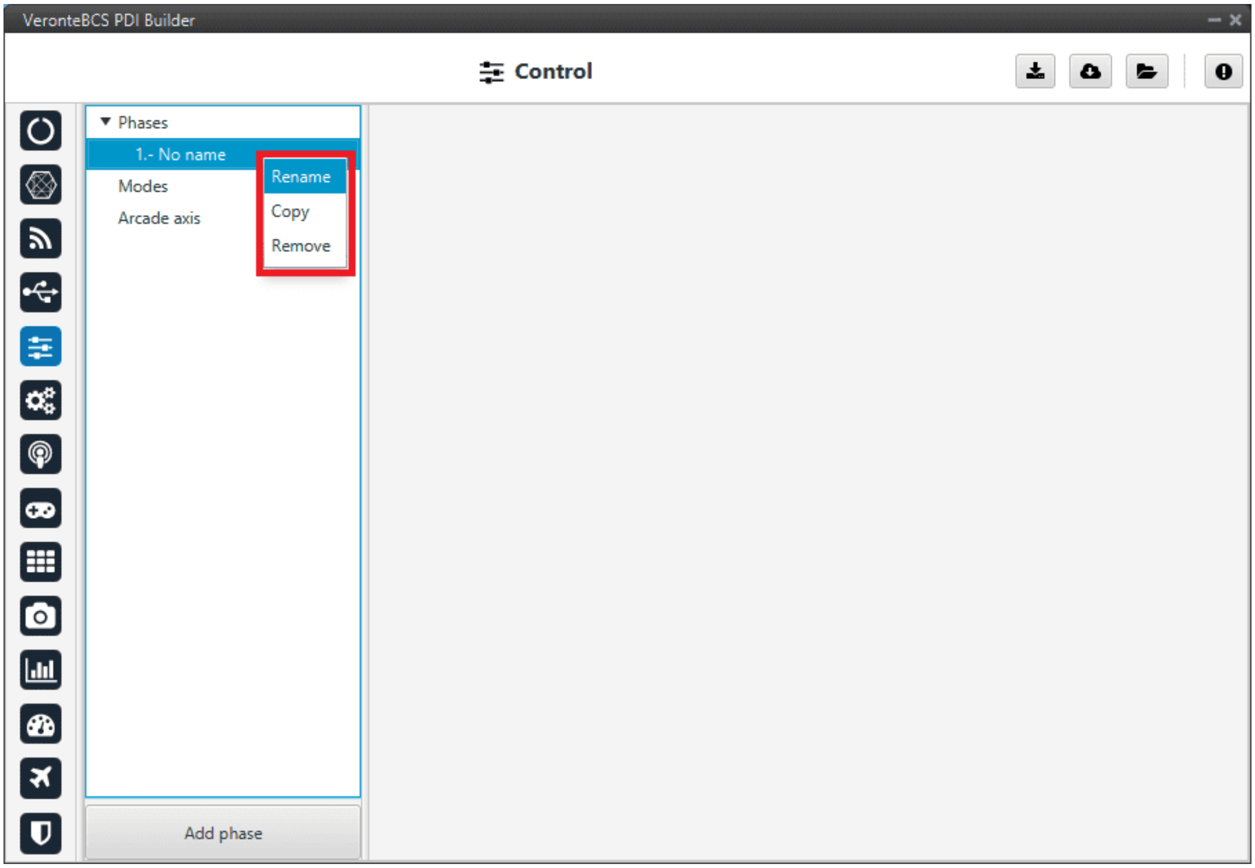


Fig. 53: **Phase options**

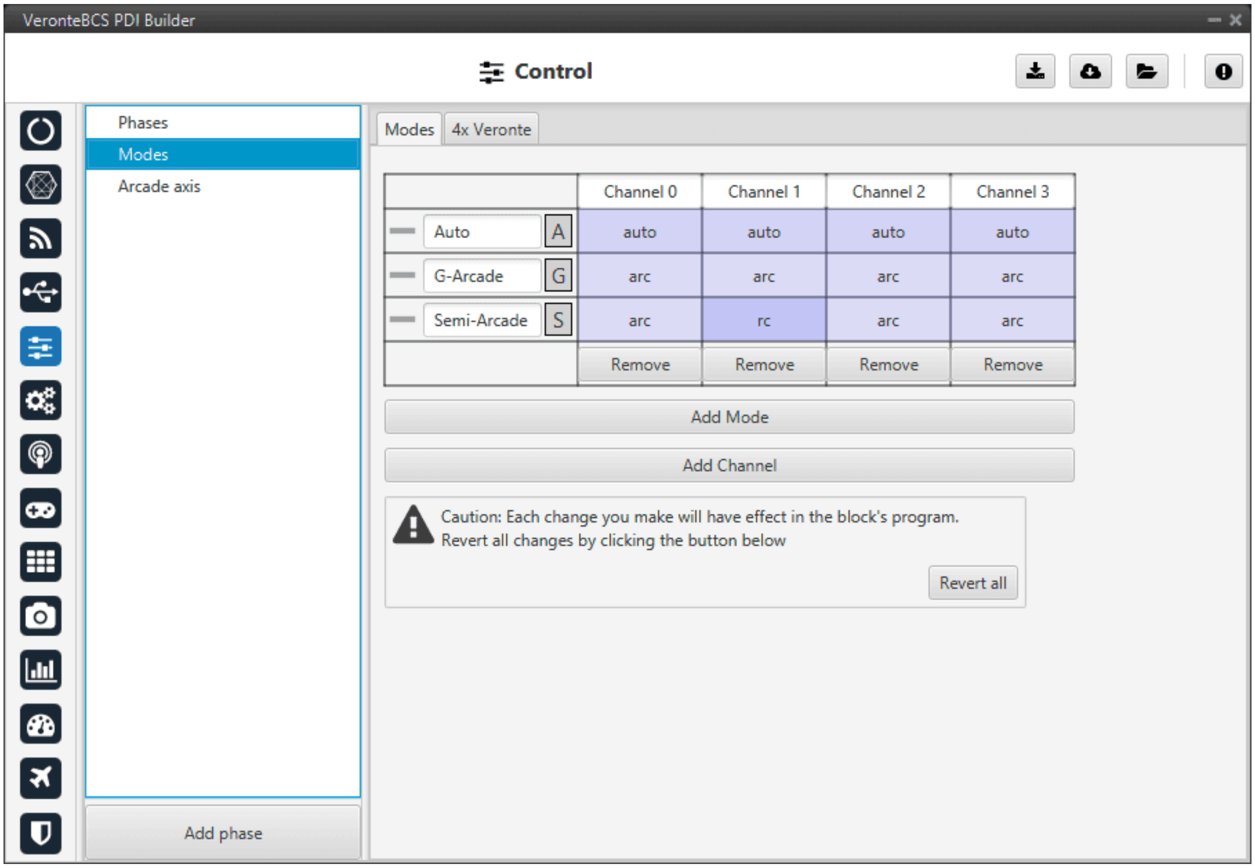
Note: The configuration of the phases (guidance and control commands) is done in the *Block Programs* section.

2.5.2 Modes

Modes

This menu allows the creation of custom flight modes. The flight modes determine who is in charge of controlling each one of the aircraft control channels.

There are 4 different control modes and it is possible to combine them to create custom flight modes.



	Channel 0	Channel 1	Channel 2	Channel 3
Auto A	auto	auto	auto	auto
G-Arcade G	arc	arc	arc	arc
Semi-Arcade S	arc	rc	arc	arc

Caution: Each change you make will have effect in the block's program. Revert all changes by clicking the button below

Fig. 54: **Modes menu**

The options available are:

- **auto**: The control channel is controlled totally by the autopilot.
- **rc**: The control is totally carried out manually. The movements on the pilot stick imply directly movements on the servo linked to that control channel.
- **arc**: The autopilot aids the radio controller during the flight, i.e it could be considered as a mix between automatic and manual. The movements on the pilot stick are the input values on the control system, so the pilot commands a desired pitch, roll, IAS, heading and so on, and is the control system who is in charge of making the platform follow those commands.
- **mix**: In this mode, it is possible to select in which step of the controller will enter the pilot command.

Example

For example, the pitching of an aircraft is commonly controlled with 3 PID being: flight path angle, pitch and pitch rate. In the arcade mode the pilot command will be a desired flight path angle that enters as input of the whole control system, but in the Mix mode is possible to select where we want the command to enter, so the pilot command could be pitch (entering in the second PID directly) or pitch rate (entering directly on the third PID).

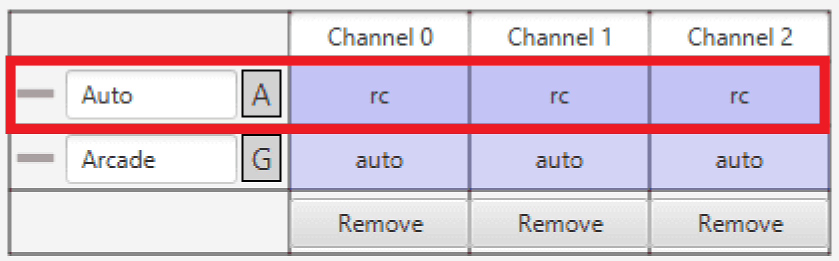
The control system will take this input as a disturbance that it wants to discard because the final objective is to match the input of the first PID (a desired flight path angle in this case), so the Mix mode can be used to make small corrections when the aircraft is following a route for example, where we want it to move slightly towards a certain direction by introducing a value directly on the roll PID.

To **change** any of this options, **click on the cell** the user would like to change and the next option will be set.

Warning:

- The name of the mode does not have to correspond to the configuration of the mode.

  For example, the user can **name** the mode as **Auto** but set the **channels** as **rc** (manual):



		Channel 0	Channel 1	Channel 2
Auto	A	rc	rc	rc
Arcade	G	auto	auto	auto
		Remove	Remove	Remove

Fig. 55: **Modes configuration**

- Moreover, although the mode is set "sensefully" here, in the block configuration (*Block Programs*) the control does not have to correspond to this.

  For example, if a channel is configured as manual (rc) here but then the control is configured so that the stick input does not control the channel, it will be auto control even though manual is specified. See the following example, where for consistency, the blocks in the 'True' and 'False' cases should be inverted:

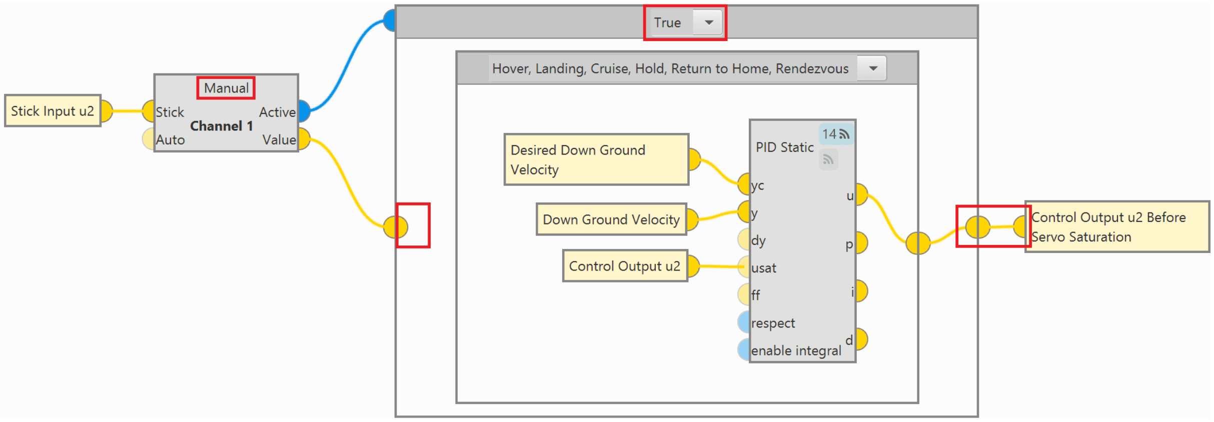


Fig. 56: **Modes configuration in blocks**

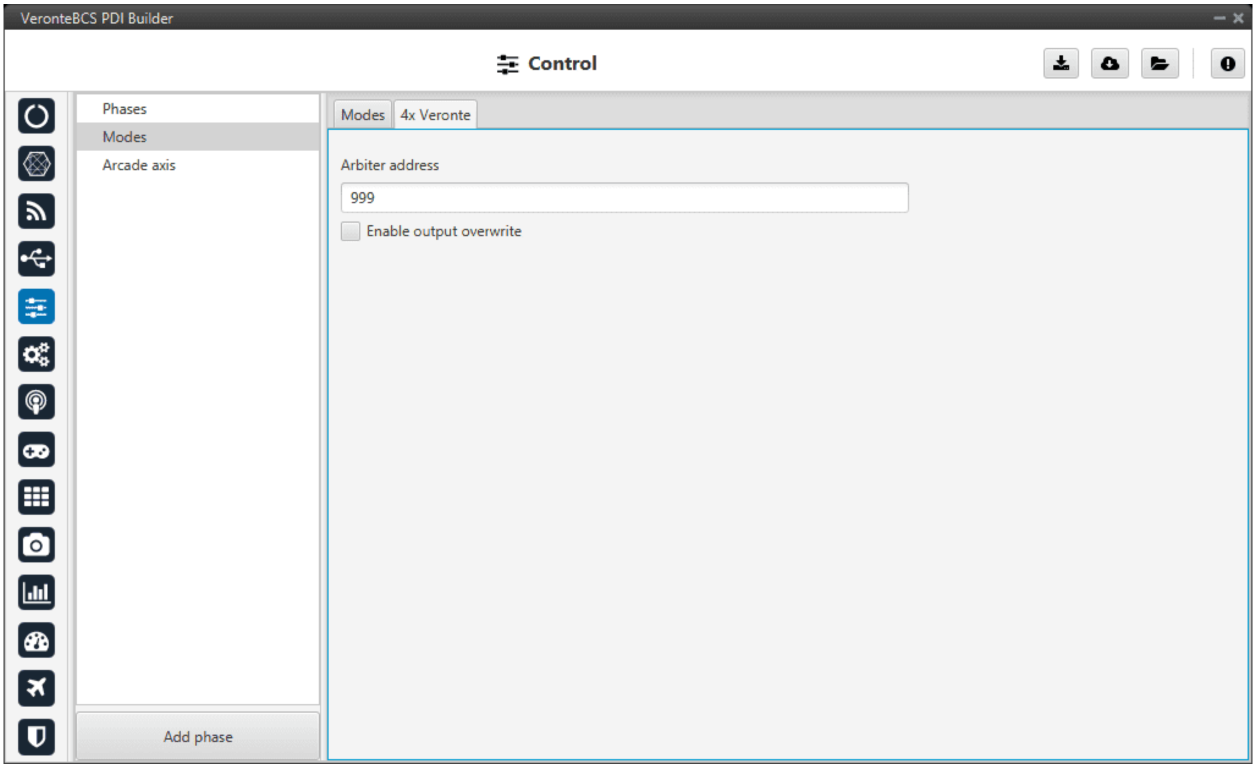
So, it is the user's responsibility to build the configuration correctly. In case of having any questions, the user should contact the support team (create a ticket in the customer's **Joint Collaboration Framework**; for more information, see Tickets section of the JCF manual).

4x Veronte

This tab allows the user to configure the **BCS** to operate in an **Autopilot 4x**.

By adding the arbiter address, **Veronte Ops** will recognise it as part of a **4x** group, and it will also be possible to do HIL simulations (with **Veronte HIL Simulator**) with this **4x**.

Note: If the arbiter address is set to 999, there is no arbiter.



Arbiter address

999

□ Enable output overwrite

Fig. 57: **4x Veronte tab**

To allow the **output to be overwritten**, the checkbox must be checked.

By enabling it, a table can be created in which columns correspond to each 1x Autopilot and the CanID row to the ID of the CAN message through which each Autopilot 1x communicates information with the other Autopilots 1x within the Veronte Autopilot 4x. Each CAN Id is associated to the **CAN 4x** producer/consumer of the AP in which it is configured. For more information on **CAN 4x**, see the *CAN Setup* section of this manual.

This option must work in conjunction with the *AP Selection block*.

2.5.3 Arcade axis

The Arcade Axis menu enables the option to change the center of the system axes. This option is used to create axis systems referred to a certain point or direction, for example, it is useful when the pilot wants to define all movement respect it (Ground axes). In this way, if the pilot command a turn right, the aircraft will turn to the right of the pilot, instead the right of the aircraft (Body axes).

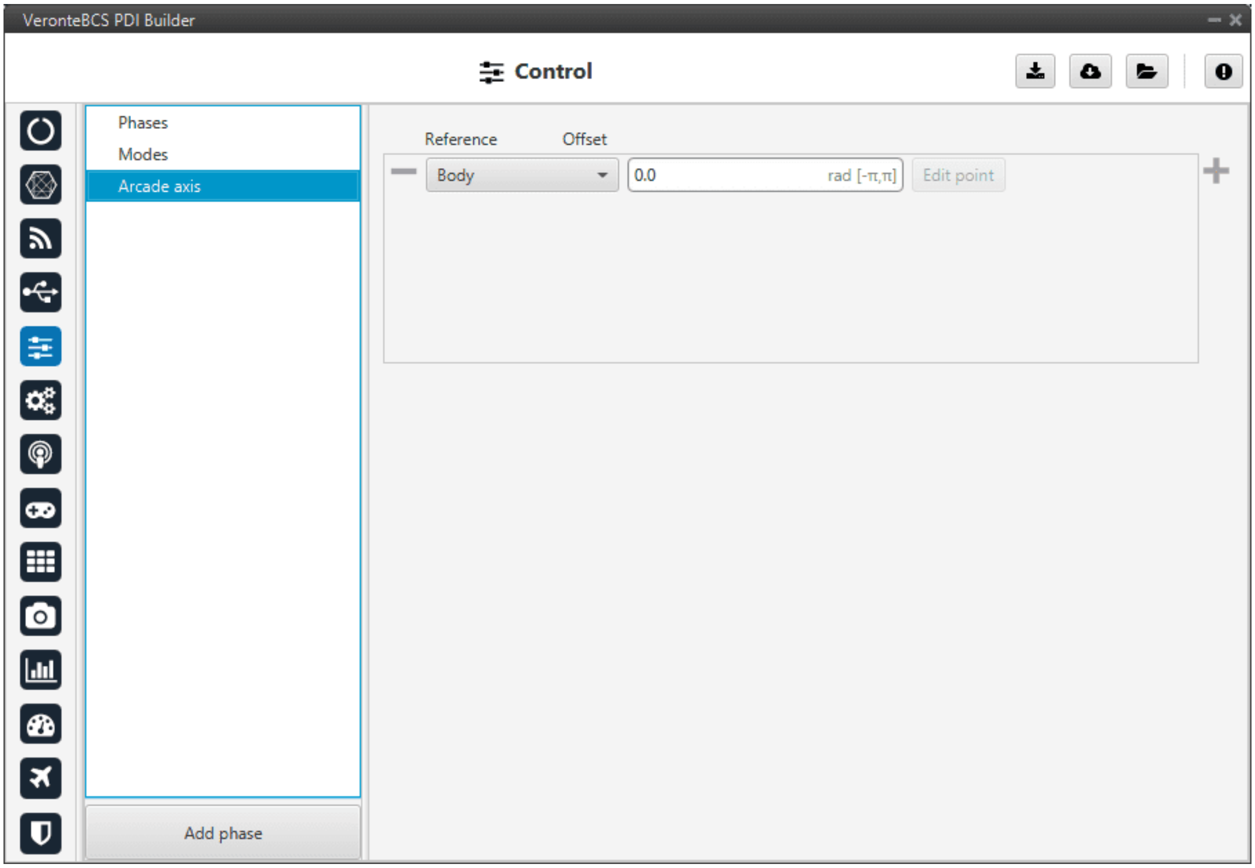


Fig. 58: **Arcade axis menu**

It is possible to add as many axes system as desired, being able to choose between the following types:

- **Body**: Fix the axes in the UAV. It is standard for the pilot.
- **Ground**: Fix the axes in the 1x GND unit.
- **Point**: Fix the axes in a point that user defines.
- **Heading**: Fix the axes in the the heading defined.
- **Desired heading**: Fix the axes in the desired heading.
- **Tangent direction**: Fix the axes in the tangent direction of the designed path.
- **Desired yaw**: Fix the axes in the desired yaw.

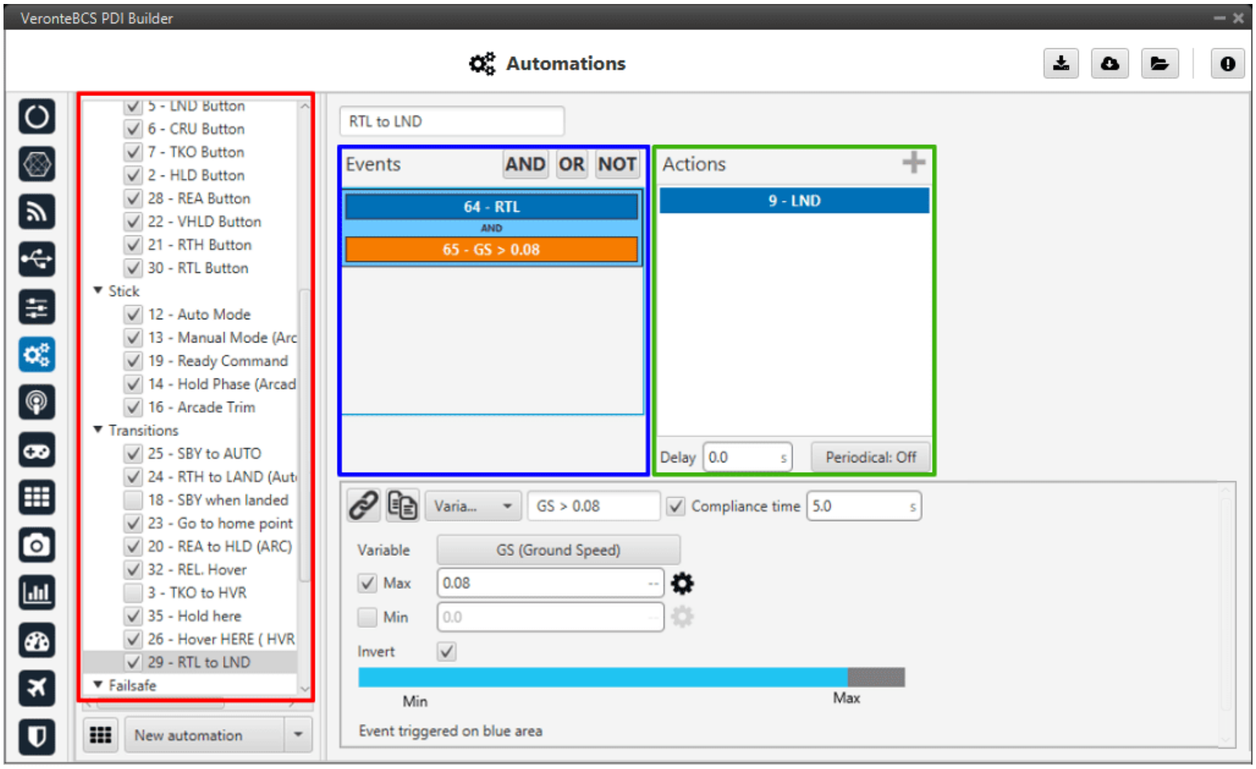
An automation can be used to select an Arcade Axis in flight, see *Actions* in Automations section.

2.6 Automations

Automations are actions that are executed when a combination of events happen, i.e when the events are accomplished the action is done. An example of what an automation could be a change of phase when reaching a certain altitude and speed, moving a servo when a button is clicked and many other possible combinations.

In this section all the possible events and actions will be explained in detail, so the user can combine them to create the automations that best suit their needs.

The following figure shows the layout of the automations menu, with a column for the events and another for the actions linked to these events.



RTL to LND

Events: 64 - RTL AND 65 - GS > 0.08

Actions: 9 - LND

Delay 0.0 s

Varia... GS > 0.08 ☑ Compliance time 5.0 s

Variable GS (Ground Speed)

☑ Max 0.08

☐ Min 0.0

Invert ☑

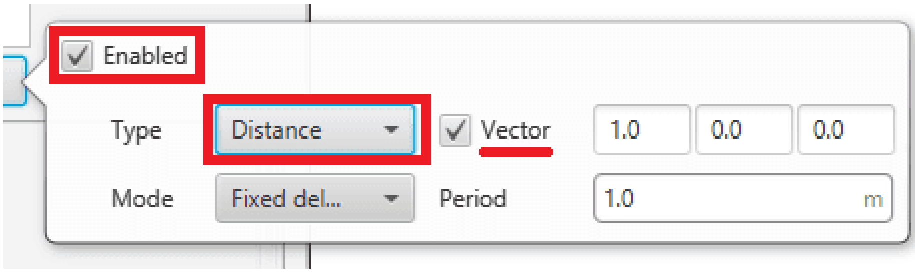
Fig. 59: **Automations menu**

All the automations that have been created (red) are a combination of events (blue) and actions (green). All actions will be performed on event or an event combination triggering.

There are some parameters that can be configured in the events and actions menu and which are applicable independently of the type of event/action configured.

- **Compliance time**: It is a value related to the automations. It indicates how much time the event has to be accomplished in order to trigger the action.
- **Delay**: It is the time between the triggering of the event and the beginning of the action.
- **Periodical**: This menu is used to configure actions to take place periodically during the time that the events are active. The action can be configured to take place each certain:

- **Distance**: When using distance, the option Vector allows to measure that distance along a direction specified by that vector.



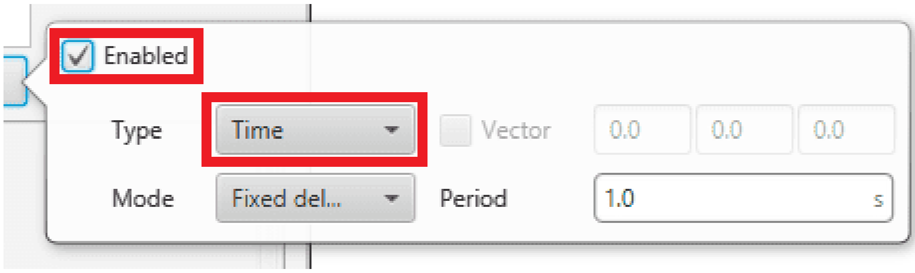
☑ Enabled

Type Distance ☑ Vector 1.0 0.0 0.0

Mode Fixed del... Period 1.0 m

Fig. 60: **Periodical distance menu**

- **Time**



☑ Enabled

Type Time □ Vector 0.0 0.0 0.0

Mode Fixed del... Period 1.0 s

Fig. 61: **Periodical time menu**

The two **Modes** available for both time and distance are **fixed delay** and **fixed period**. In order to explain the difference between them, the following figure is presented as an aid to the user.

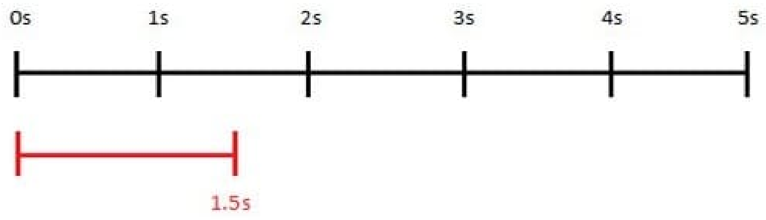


Fig. 62: **Periodical modes**

Let's consider that the system evaluates the automations each second (black line), and the automation that contains the periodical option is wanted to execute each 1.5 seconds (red line). In that case, the **first action** will be **triggered at the second 1.5** but will be **evaluated at second 2**. The **second time** that the action is evaluated will depend on the mode, if it is selected:

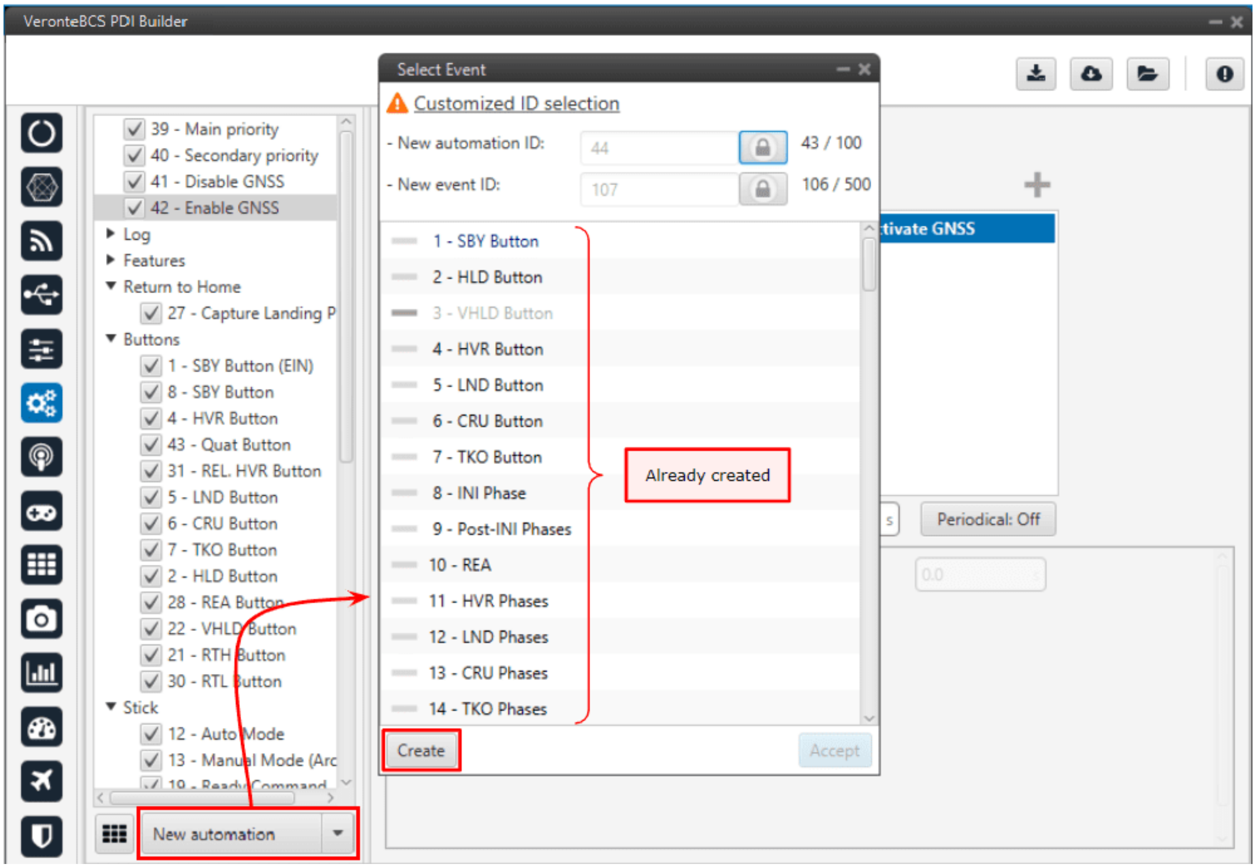
- **Fixed delay**, the evaluation of the action will be done **1.5 seconds after it was evaluated** the first time, so that will be at **second 3.5**.
- **Fixed period**, the action will be evaluated **1.5 seconds after the first triggering** (not evaluation) so that would be at the **second 3**.

In the real praxis, the evaluation time for the automation is much lower than 1 second so the difference between the modes is much smaller.

2.6.1 New automation

Warning: It is important to know that there is a limit of **500 events**, **120 actions** and **100 automations**.

To create a new automation press **New Automation** and a new window will be displayed. Users can select a **previous one** (if exists) or **Create** new.



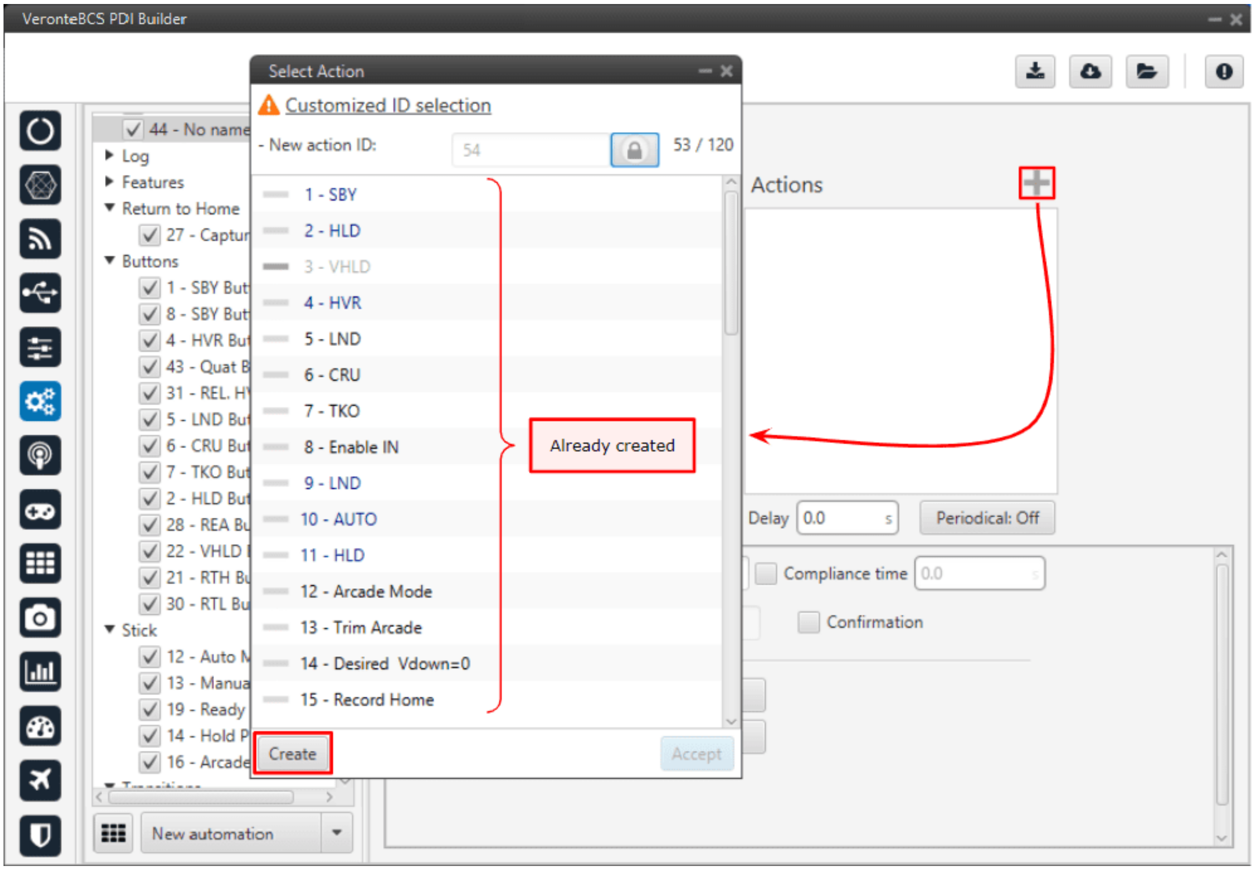
- New automation ID: 44 43 / 100

- New event ID: 107 106 / 500

Already created

Fig. 63: **New automation**

To add an action to the automation, press "+" icon and a new window will be displayed. Users can select **previous one** (if exists) or **Create** new.



VeronteBCS PDI Builder

Select Action

Customized ID selection

- New action ID: 54 53 / 120

1 - SBY
2 - HLD
3 - VHLD
4 - HVR
5 - LND
6 - CRU
7 - TKO
8 - Enable IN
9 - LND
10 - AUTO
11 - HLD
12 - Arcade Mode
13 - Trim Arcade
14 - Desired Vdown=0
15 - Record Home

Already created

44 - No name
Log
Features
Return to Home
☑ 27 - Captur
Buttons
☑ 1 - SBY But
☑ 8 - SBY But
☑ 4 - HVR Bu
☑ 43 - Quat B
☑ 31 - REL. H
☑ 5 - LND Bu
☑ 6 - CRU Bu
☑ 7 - TKO Bu
☑ 2 - HLD Bu
☑ 28 - REA B
☑ 22 - VHLD
☑ 21 - RTH B
☑ 30 - RTL Bu
Stick
☑ 12 - Auto M
☑ 13 - Manua
☑ 19 - Ready
☑ 14 - Hold P
☑ 16 - Arcade

Delay 0.0 s

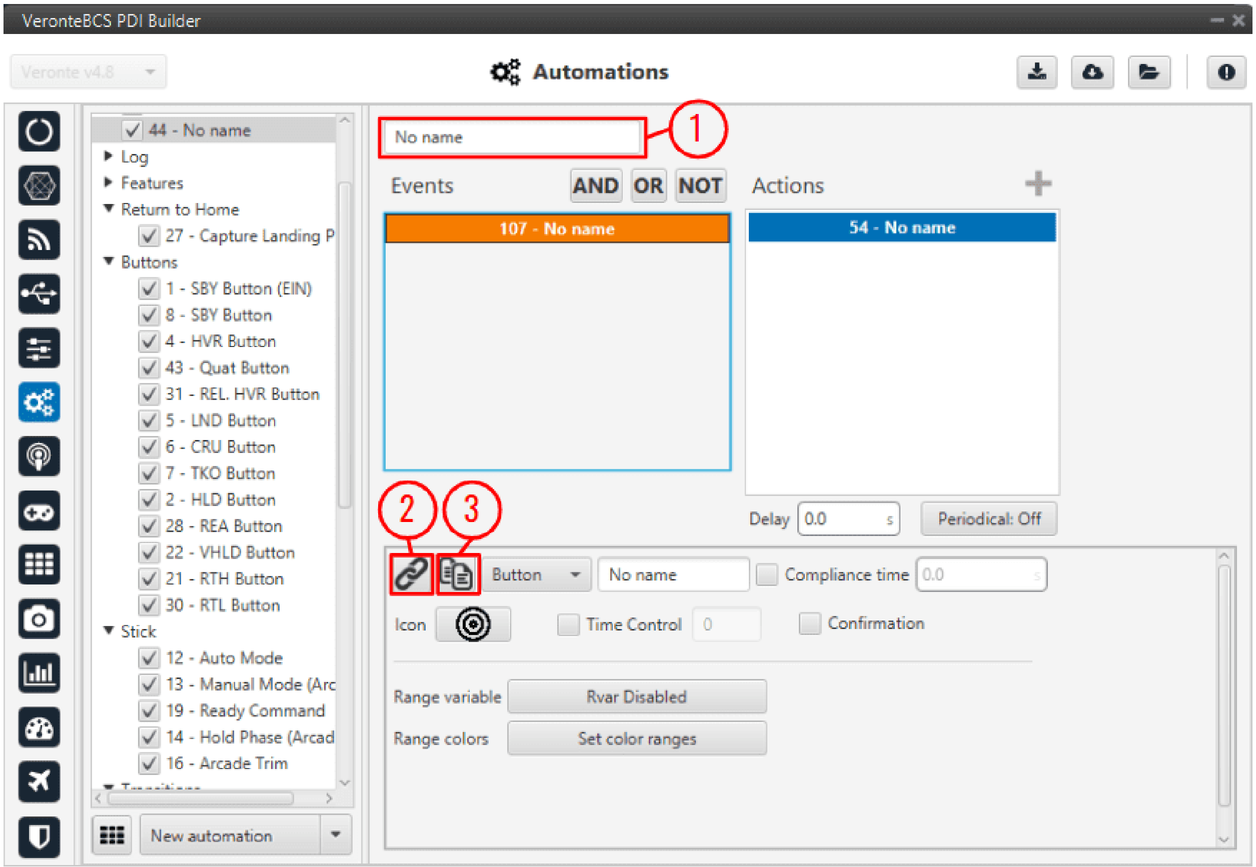
Periodical: Off

□ Compliance time 0.0 s

□ Confirmation

Fig. 64: **New action**

When an automation is created, the following options are available:



- Name: No name
- Delay 0.0 s
- Periodical: Off
- Button: No name
- □ Compliance time 0.0 s
- □ Time Control 0
- □ Confirmation
- Range variable: Rvar Disabled

Fig. 65: **New automation options**

1. The user can rename the automation with the name of his choice.
2. **Use existing** button (🔗 icon): Select an action or event from the available in the system. When modifying an action or event it will be **modified in all automations where it is in use**.
3. **Clone** button (📋 icon): Clone an existing action or event creating a new one with same parameters configured on the start point.

By right clicking on an automation it is possible to **remove** it, **clone** it or **change it of group**. When a group is created, the rest of automations that the user wants to add to the group can be done by drag and drop.

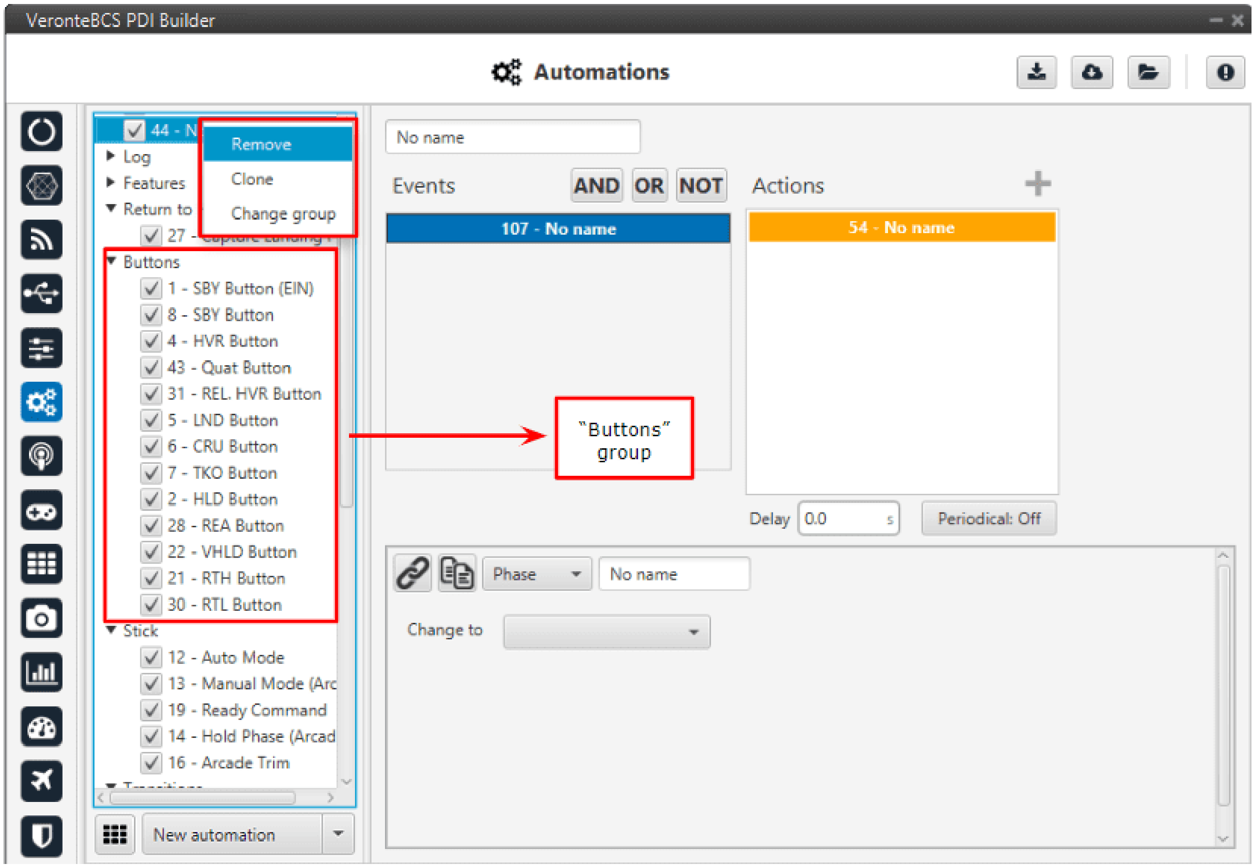
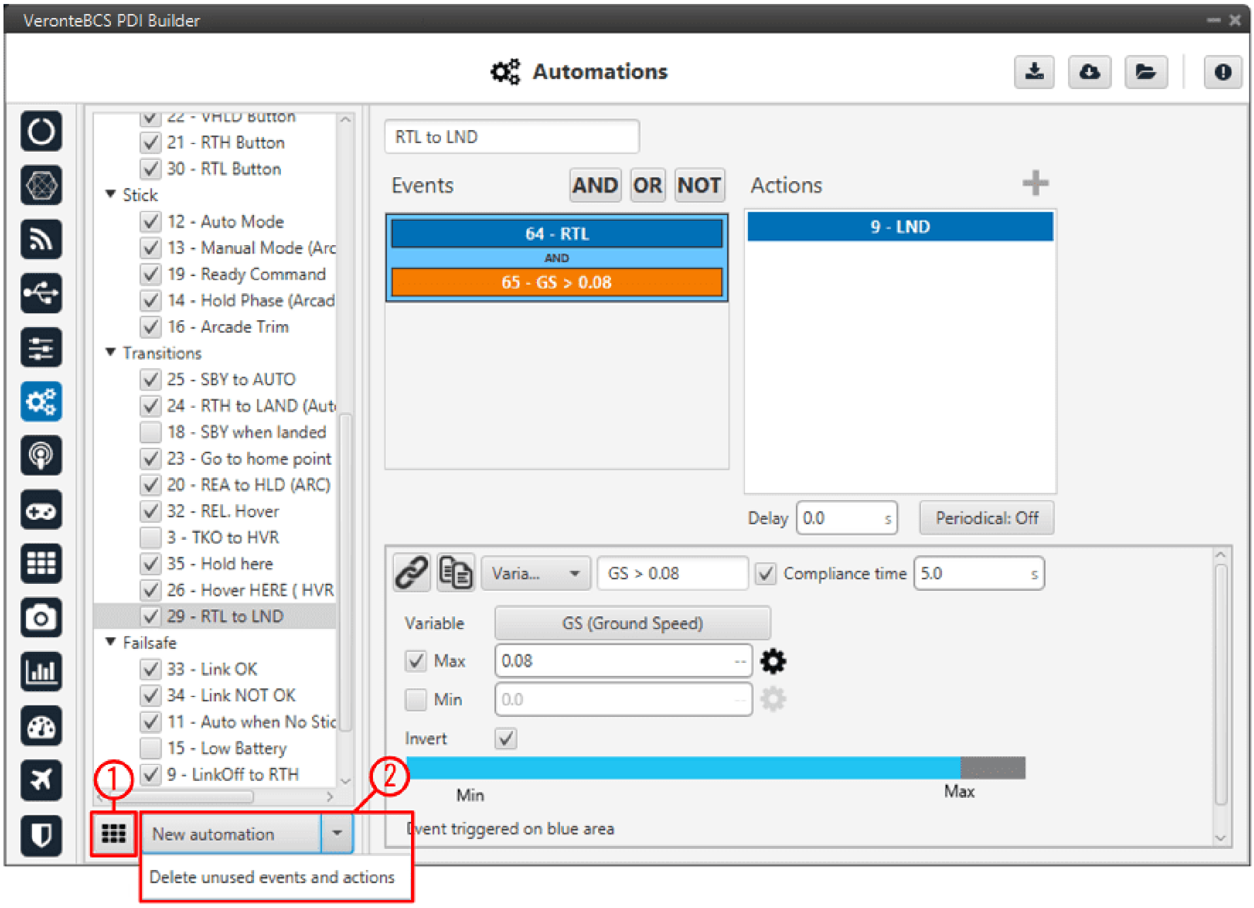


Fig. 66: **Automations groups**

Warning: When a clone of an automation is created, the changes made in the **event panel** will be applied to the other one and vice versa, while the actions can be different in each automation.

2.6.2 Other options

In the figure below, the user can see 2 additional options:



- ☑ 22 - VHLD Button
- ☑ 21 - RTH Button
- ☑ 30 - RTL Button
- Stick
  - ☑ 12 - Auto Mode
  - ☑ 13 - Manual Mode (Arc
  - ☑ 19 - Ready Command
  - ☑ 14 - Hold Phase (Arcad
  - ☑ 16 - Arcade Trim
- Transitions
  - ☑ 25 - SBY to AUTO
  - ☑ 24 - RTH to LAND (Aut
  - □ 18 - SBY when landed
  - ☑ 23 - Go to home point
  - ☑ 20 - REA to HLD (ARC)
  - ☑ 32 - REL. Hover
  - □ 3 - TKO to HVR
  - ☑ 35 - Hold here
  - ☑ 26 - Hover HERE (HVR
  - ☑ 29 - RTL to LND
- Failsafe
  - ☑ 33 - Link OK
  - ☑ 34 - Link NOT OK
  - ☑ 11 - Auto when No Stic
  - □ 15 - Low Battery
  - ☑ 9 - LinkOff to RTH

RTL to LND

Events: 64 - RTL AND 65 - GS > 0.08

Actions: 9 - LND

Delay 0.0 s Periodical: Off

Varia... GS > 0.08 ☑ Compliance time 5.0 s

Variable GS (Ground Speed)

☑ Max 0.08

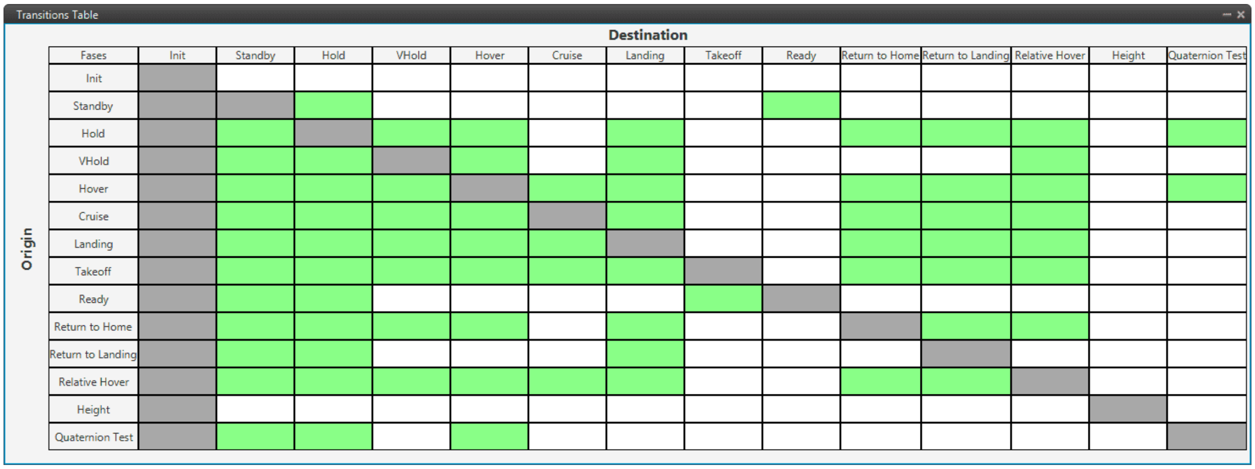
□ Min 0.0

Invert ☑

Event triggered on blue area

Fig. 67: **Automations options**

1. By clicking here, a phase transition table will appear:

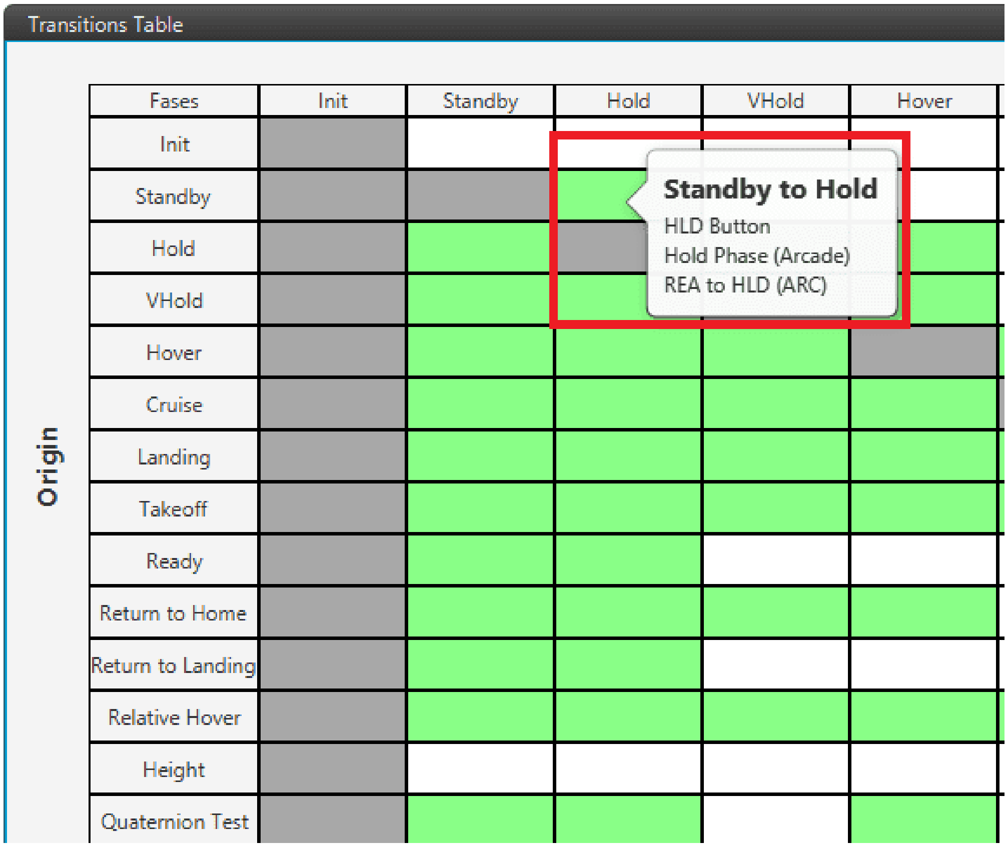


Fases	Init	Standby	Hold	VHold	Hover	Cruise	Landing	Takeoff	Ready	Return to Home	Return to Landing	Relative Hover	Height	Quaternion Test
Init	▨													
Standby	▨	▨	🟩						🟩					
Hold	▨	🟩	▨	🟩	🟩		🟩			🟩	🟩	🟩		🟩
VHold	▨	🟩	🟩	▨	🟩		🟩					🟩		
Hover	▨	🟩	🟩	🟩	▨	🟩	🟩			🟩	🟩	🟩		🟩
Cruise	▨	🟩	🟩	🟩	🟩	▨	🟩			🟩	🟩	🟩		
Landing	▨	🟩	🟩	🟩	🟩	🟩	▨			🟩	🟩	🟩		
Takeoff	▨	🟩	🟩	🟩	🟩	🟩	🟩	▨		🟩	🟩	🟩		
Ready	▨	🟩	🟩					🟩	▨					
Return to Home	▨	🟩	🟩	🟩	🟩		🟩			▨	🟩	🟩		
Return to Landing	▨	🟩	🟩				🟩				▨			
Relative Hover	▨	🟩	🟩	🟩	🟩	🟩	🟩			🟩	🟩	▨		
Height	▨												▨	
Quaternion Test	▨	🟩	🟩		🟩									▨

Fig. 68: **Phase transition table**

In this table, the transitions between each phase can be visualised.

In addition, by clicking on a cell, it is possible to see which automation (and the events) makes the transition between the two phases posible.



Fases	Init	Standby	Hold	VHold	Hover
Init	▨				
Standby	▨	▨	🟩		
Hold	▨	🟩	▨		🟩
VHold	▨	🟩	🟩		🟩
Hover	▨	🟩	🟩	🟩	▨
Cruise	▨	🟩	🟩	🟩	🟩
Landing	▨	🟩	🟩	🟩	🟩
Takeoff	▨	🟩	🟩	🟩	🟩
Ready	▨	🟩	🟩		
Return to Home	▨	🟩	🟩	🟩	🟩
Return to Landing	▨	🟩	🟩		
Relative Hover	▨	🟩	🟩	🟩	🟩
Height	▨				
Quaternion Test	▨	🟩	🟩		🟩

Standby to Hold
HLD Button
Hold Phase (Arcade)
REA to HLD (ARC)

Fig. 69: **Phase transition table - automation**

2. **Delete unused events and actions**: This option deletes those events or actions that has been created but are not in use in any automation.

2.6.2.1 Events

An event is something that has to be accomplished to trigger the actions. All the events can be combined to create a custom event, using the **boolean operations** provided by the software (**AND**, **OR**, **NOT**).

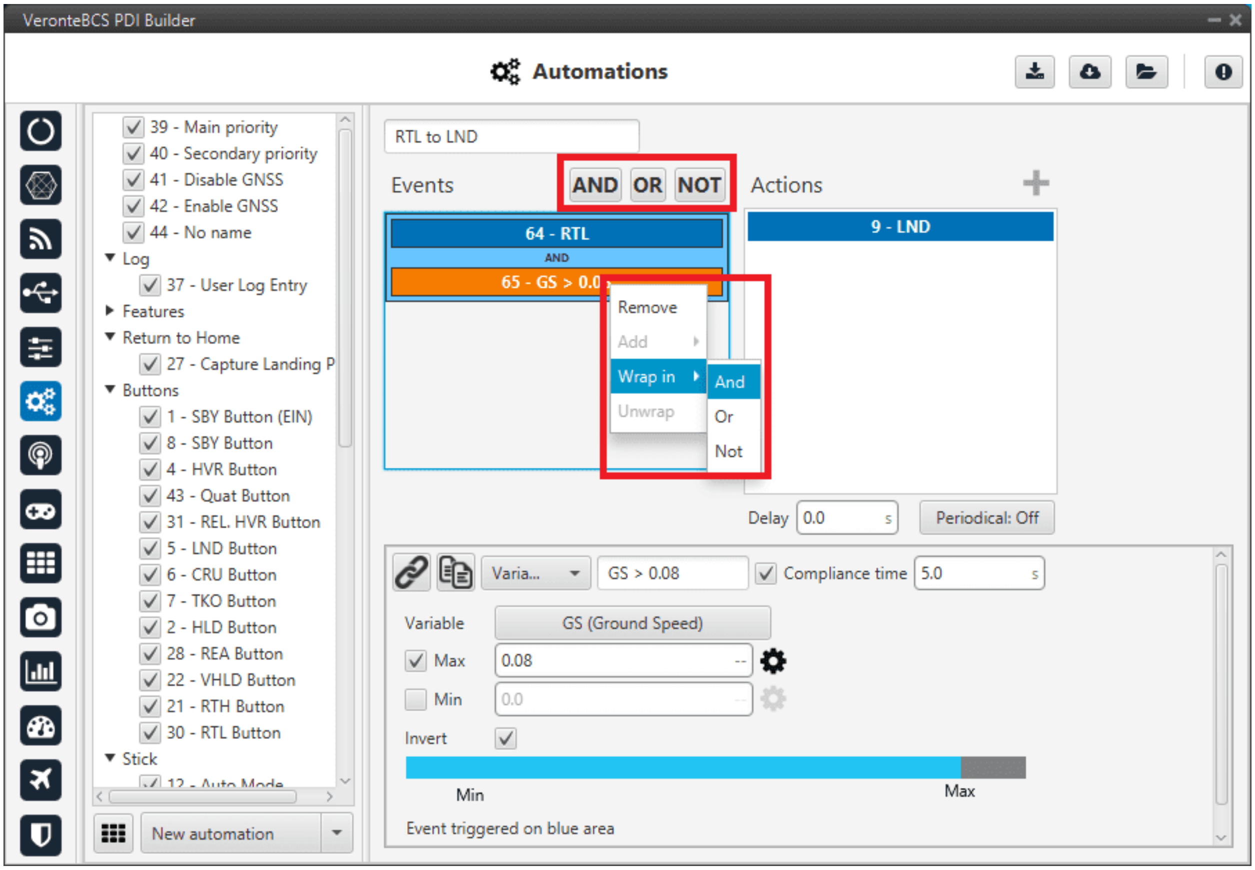


Fig. 70: **Events options**

The following table depicts the meaning of each one of the boolean operators.

Logics	Description
AND	All events grouped on an AND should be accomplished simultaneously in order to activate the automation.
OR	One of the events in the group should be accomplished for activating the automation.
NOT	The event will be active meanwhile the event or event group is not accomplished.

When there is only one event, clicking on the boolean command will create another event linked to the other one according to that operation. By right clicking on an event and selecting Wrap in allows the creation of an operation as if it was inside brackets, i.e it will be evaluated first. Let's consider the following event group as an example.

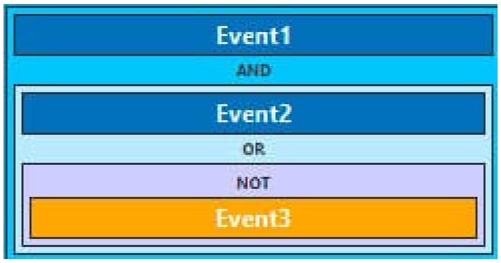


Fig. 71: **Events wrapped**

The first operation that is evaluated is the NOT, then the OR between Event2 and the result of the NOT, and finally the AND between Event1 and the result of the OR.

When createring a new event it is possible to choose from one of the **previously created** on the system or to **create** a **new** one.

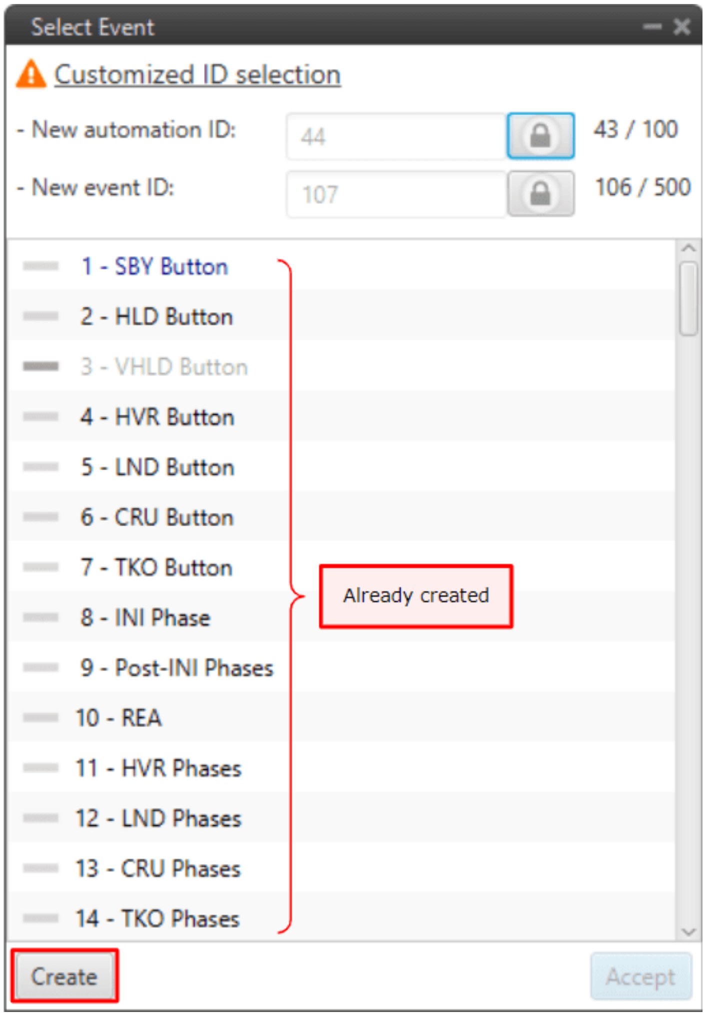


Fig. 72: **New event**

The user can also rename the event with the name of his choice.

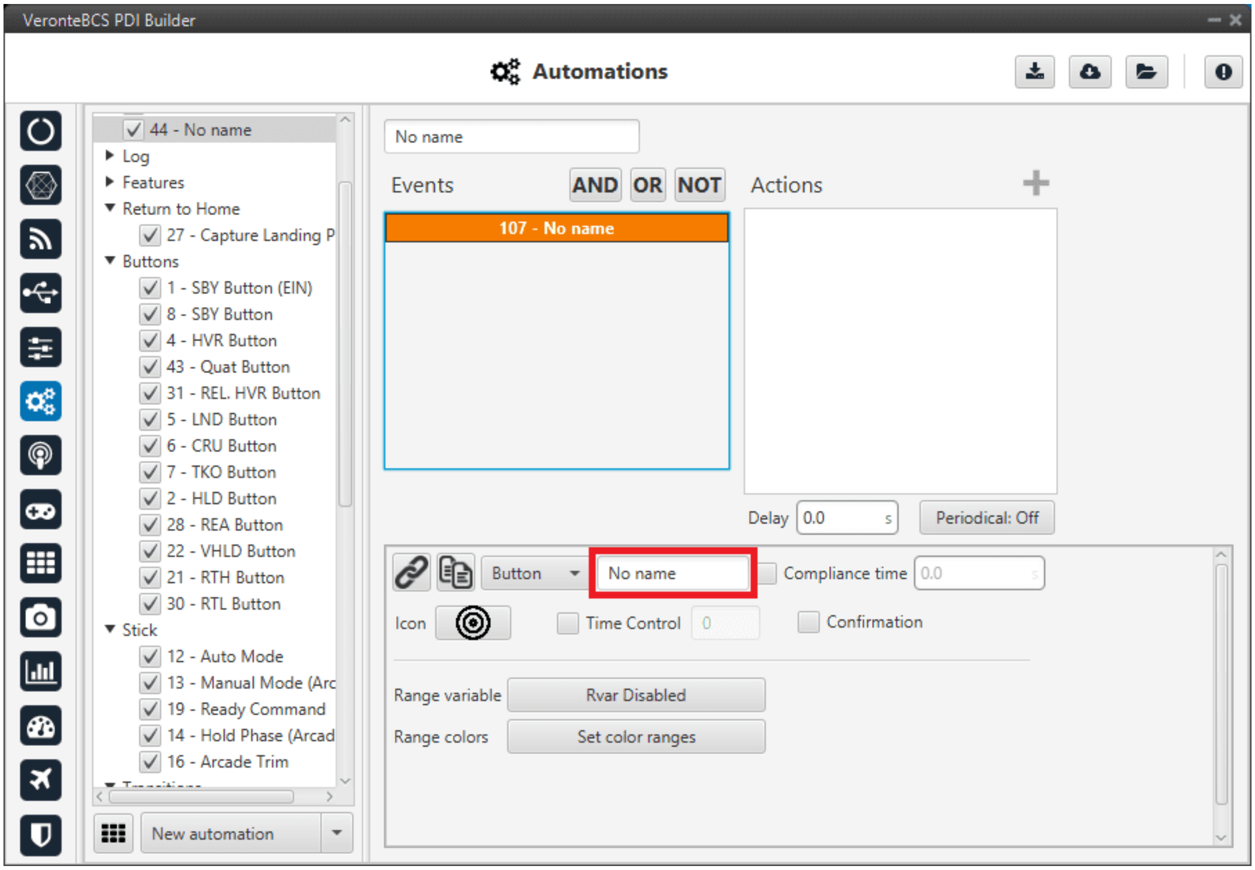
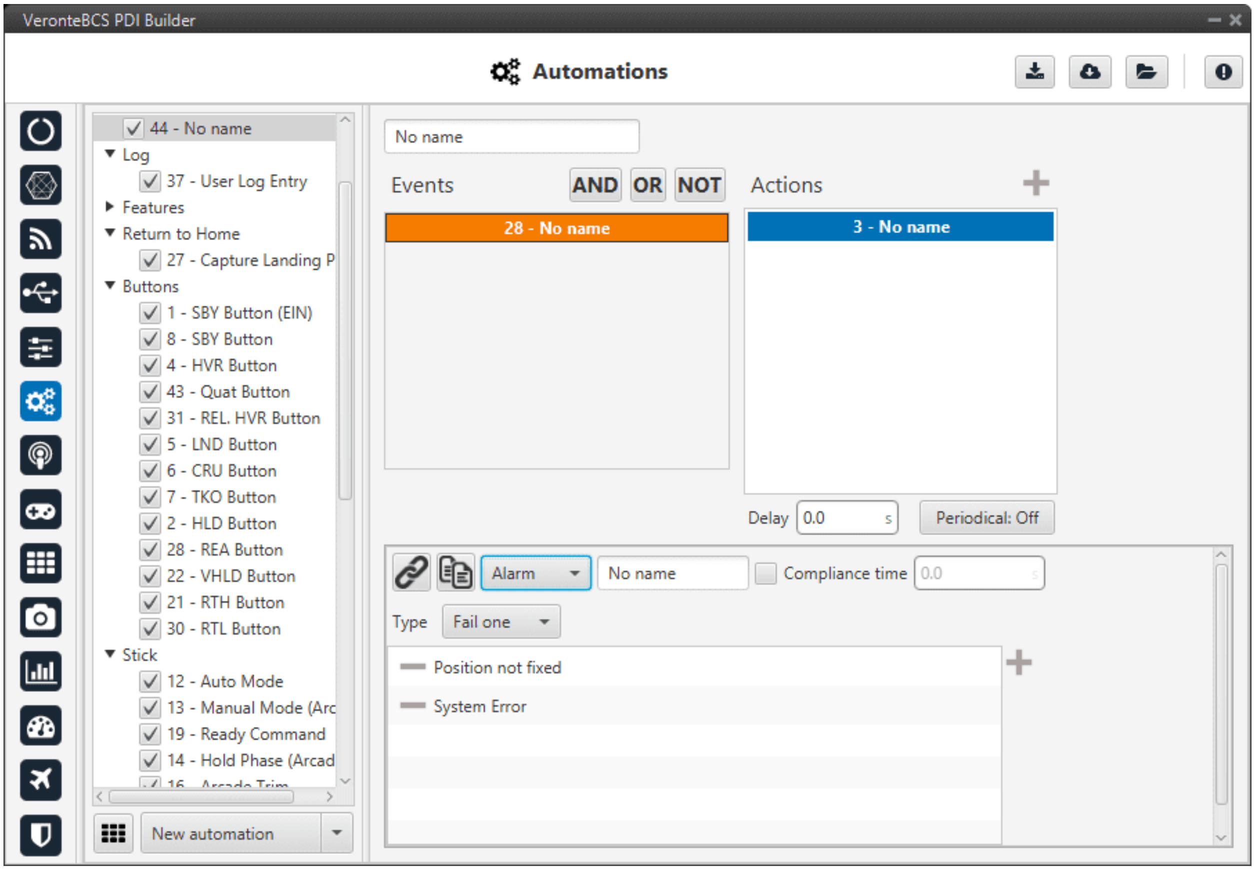


Fig. 73: **New event - name**

Below are present the different types of events that can be created.

2.6.2.1.1 Alarm

This kind of automation allows the user to add any bit of the system as an alarm. Depending on the mode in which it is configured, it will be activated in one way or another.



No name

Delay 0.0 s

Periodical: Off

Alarm

No name

□ Compliance time 0.0 s

Type Fail one

Fig. 74: **Alarm event**

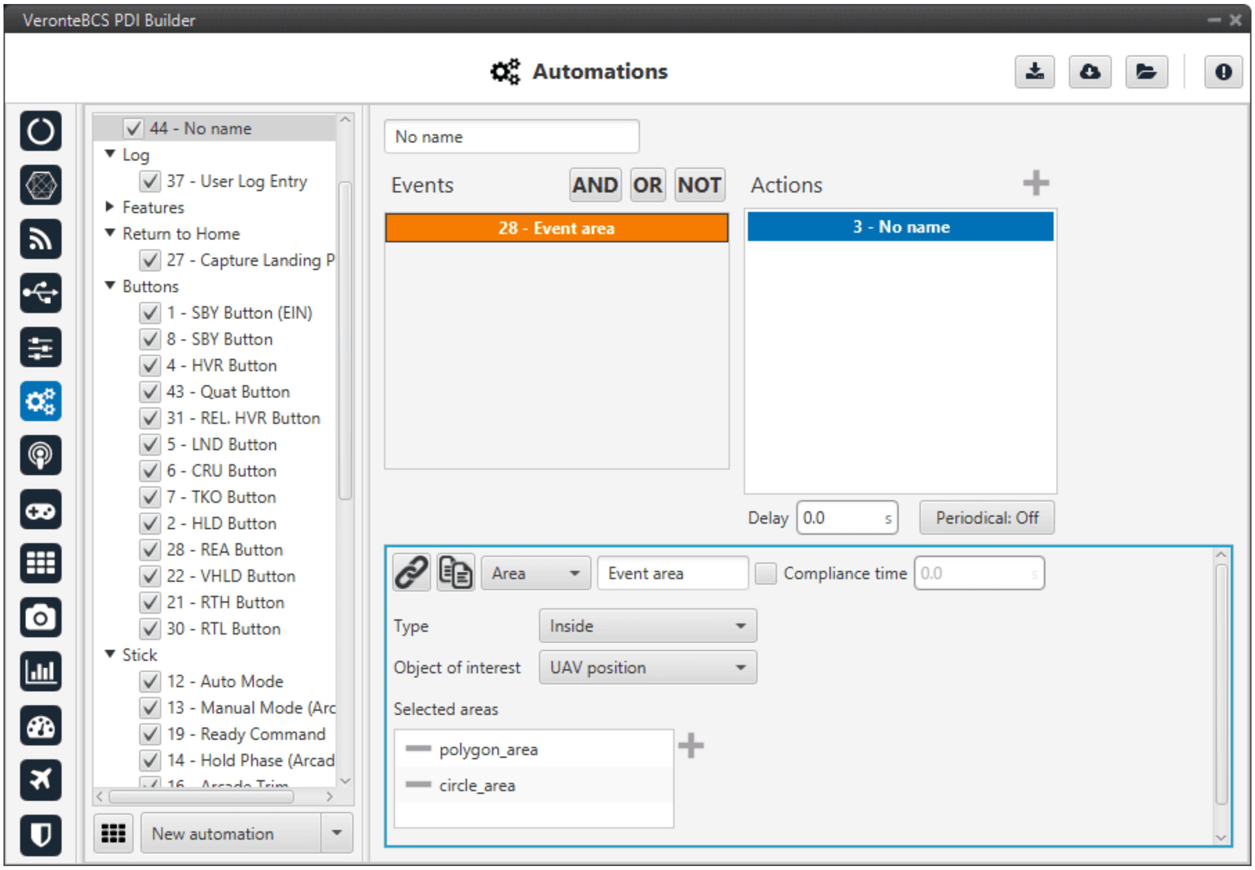
The two possible modes are the following:

- **Fail one**: it is triggered when one of the bits is set to false.
- **All ok**: it is triggered when all bits are set to true.

A common alarm event is the **Position not fixed** in fail one mode, which is triggered when there is not GPS signal.

2.6.2.1.2 Area

The event is triggered when the aircraft is inside or outside an area defined in the mission. For more information on mission creation, take a look at the Veronte Ops manual.



Delay 0.0 s

Area: Event area

☐ Compliance time 0.0 s

Type: Inside

Object of interest: UAV position

Selected areas:

- polygon_area
- circle_area

Fig. 75: **Area event**

- **Type**: Inside or Outside.
- **Object of interest**: The user has to select which object is the one that should fulfill the event.
- **Selected areas**: To select an area, first define the desired areas (**polygons** or **circles**) in the *Operation elements section of the UI menu*.

When the event has been labeled (“Event area” in this case) and saved, it is possible to link it to an area drawn on the map with the **Operation panel** (see more about this at the Veronte Ops manual) .

2.6.2.1.3 Button

This option creates a button that will trigger the event when it is clicked.

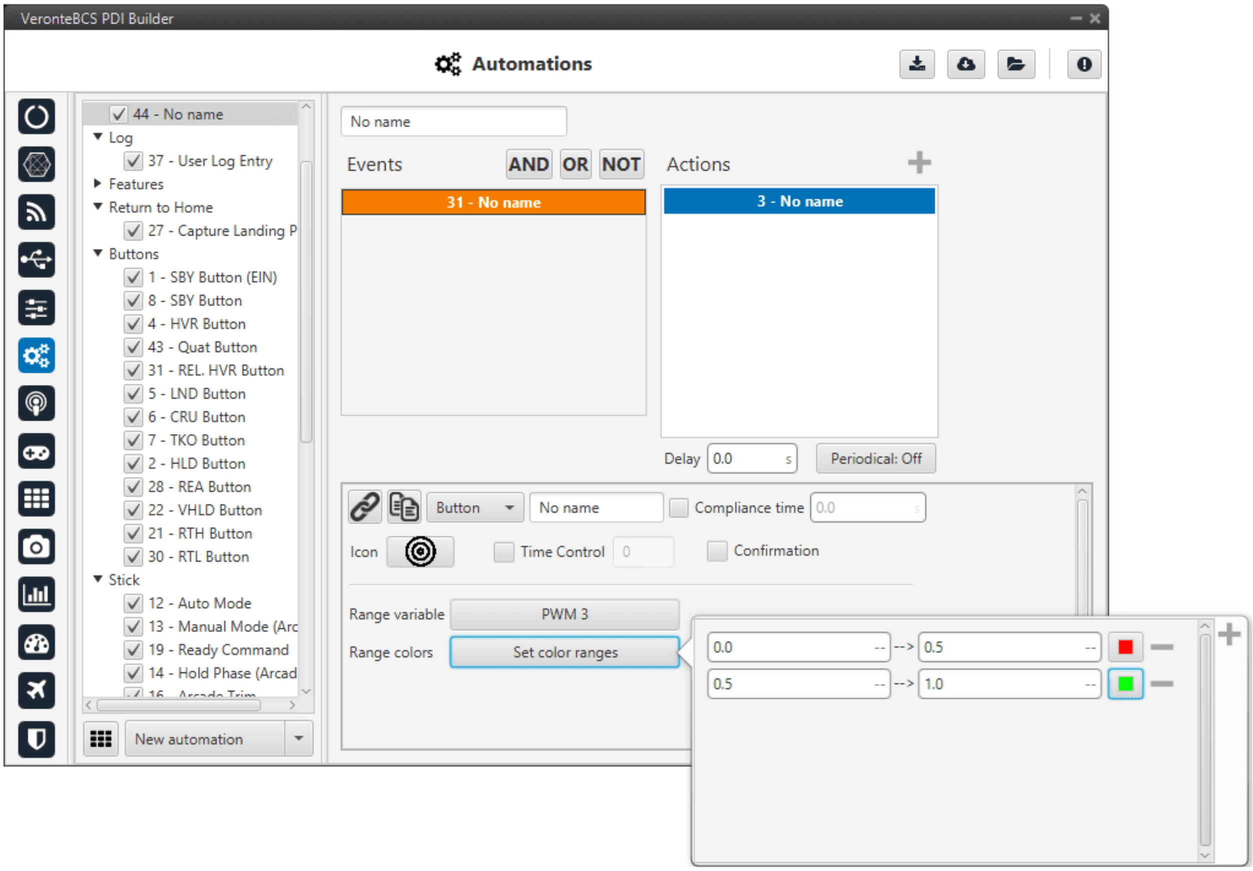


Fig. 76: **Button event**

The following options are available:

- **Icon**: The user can select the most appropriate icon for the event from a list of icons provided by the software.
- **Time Control**: This option is used to trigger the action when the button is being pushed during the time specified in this option.
- **Confirmation**: A pop-up window asking for confirmation will be display after pushing the button, so it is a safety measure.
- **Range variable** and **range colors** options are used to make the button change its color according to the value of a variable. To do that, select a variable and then indicate as many points as desired, each one with its corresponding value and color.

> **Warning:** For the buttons to be colored, it is necessary that the chosen variables have been added to the **mandatory telemetry**, adding it to the completmentary telemetry is not sufficient.

Note:

- If a button event triggers an action that consists of a change to a determined phase, the button will be the one of the **Veronte Panel** with the name of that phase on it.
  - To create the button for changing to a determinated phase, it is only needed to create the button with the same name as the phase.

- If the button event is linked to a different action (servo movement, variable, etc.), it will appear at the top of the **Veronte Panel**, as an '**action button**', with the icon selected by the user.

2.6.2.1.4 Mode

The event is triggered when the aircraft is in one of the modes selected.

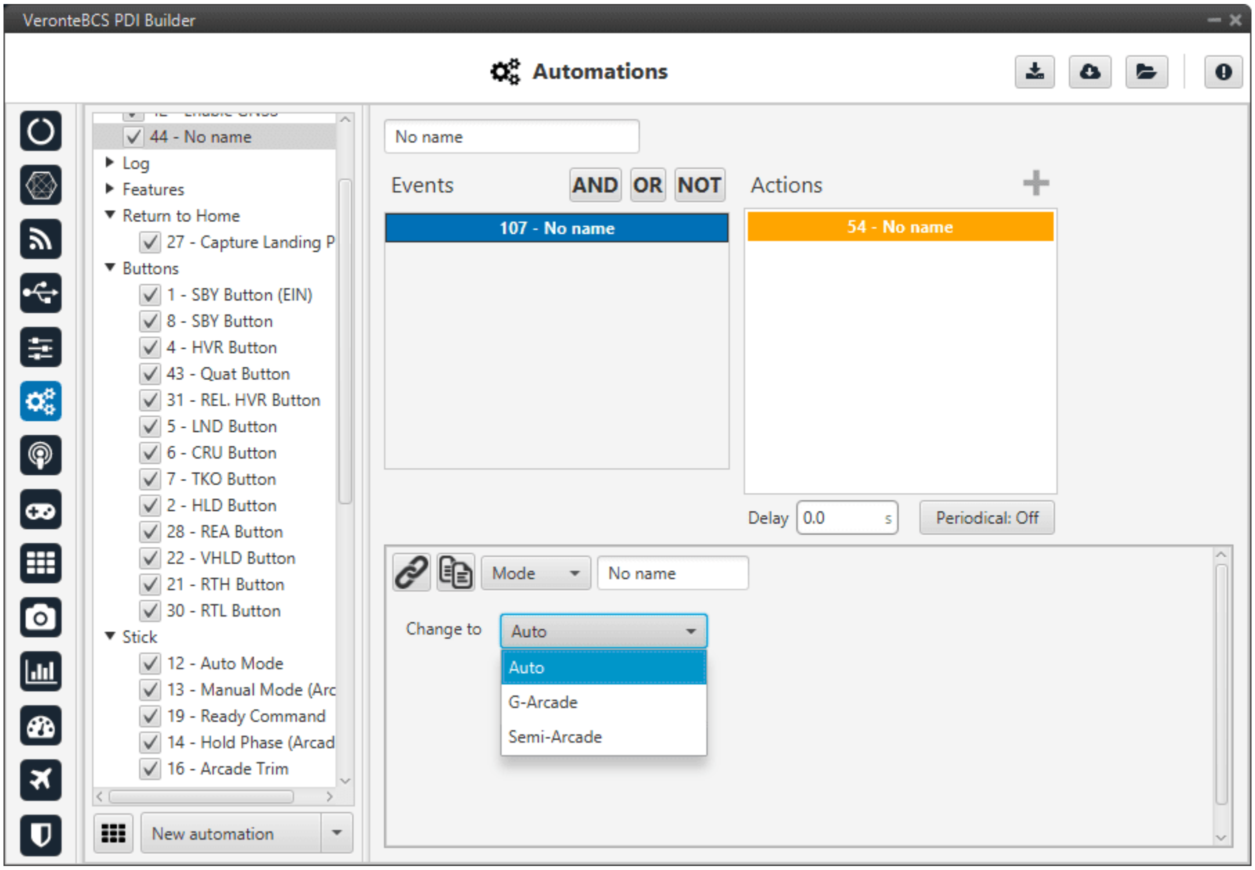


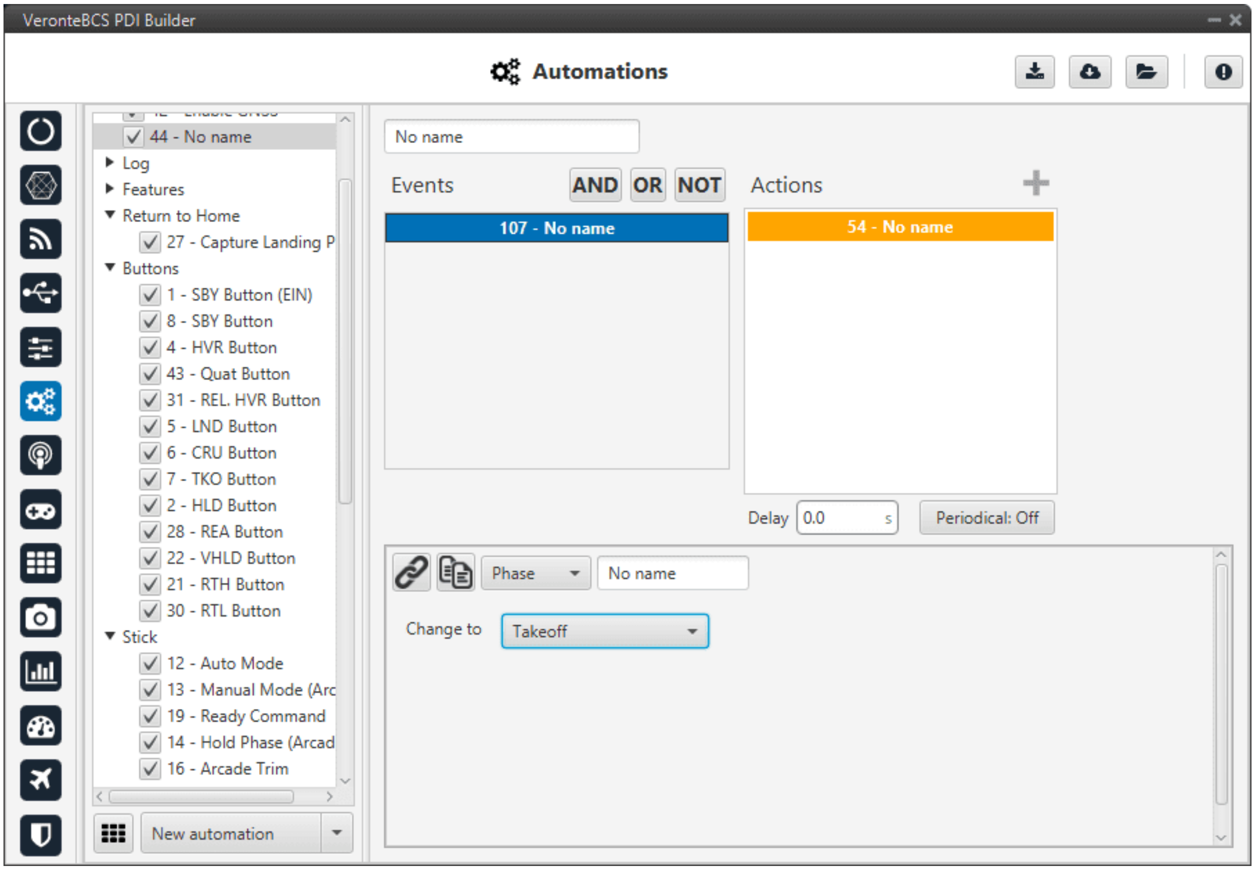
Fig. 77: **Mode event**

These modes have been created previously. See section *Modes*, for more information about creating modes.

The **compliance time** option could be interesting in this type of event.

2.6.2.1.5 Phase

The event is triggered when the aircraft is in the phases selected by clicking on the "+" button, being in any of them will trigger the action.



No name

Delay 0.0 s

Phase: No name

Change to: Takeoff

Fig. 78: **Phase event**

These phases have been created previously. See section *Phases*, for more information about creating phases.

2.6.2.1.6 Route

This event is related with the patches and marks defined by the user in the *Operation elements section of the UI menu* and to those created in the mission (in **Veronte Ops**, see more about the creation of marks and patches in the Veronte Ops manual).

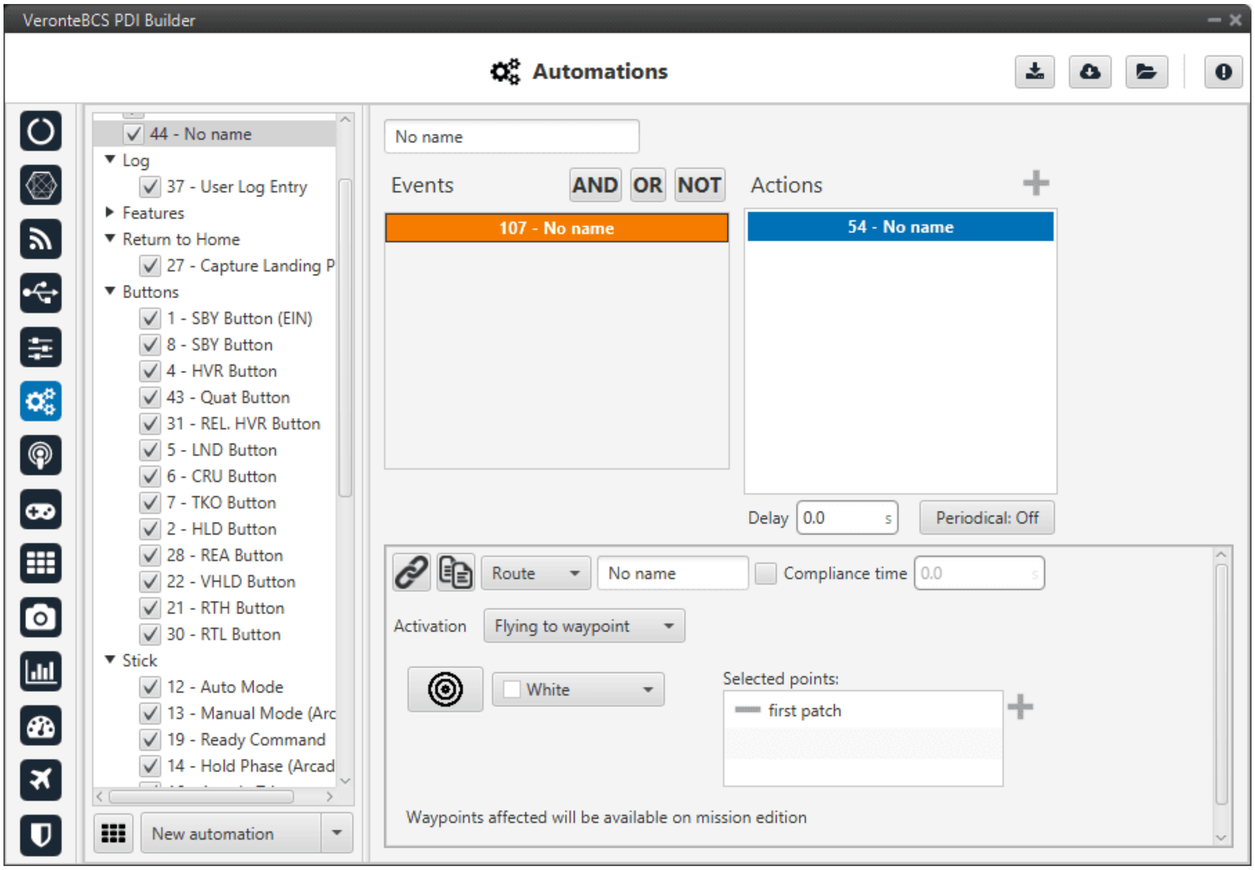


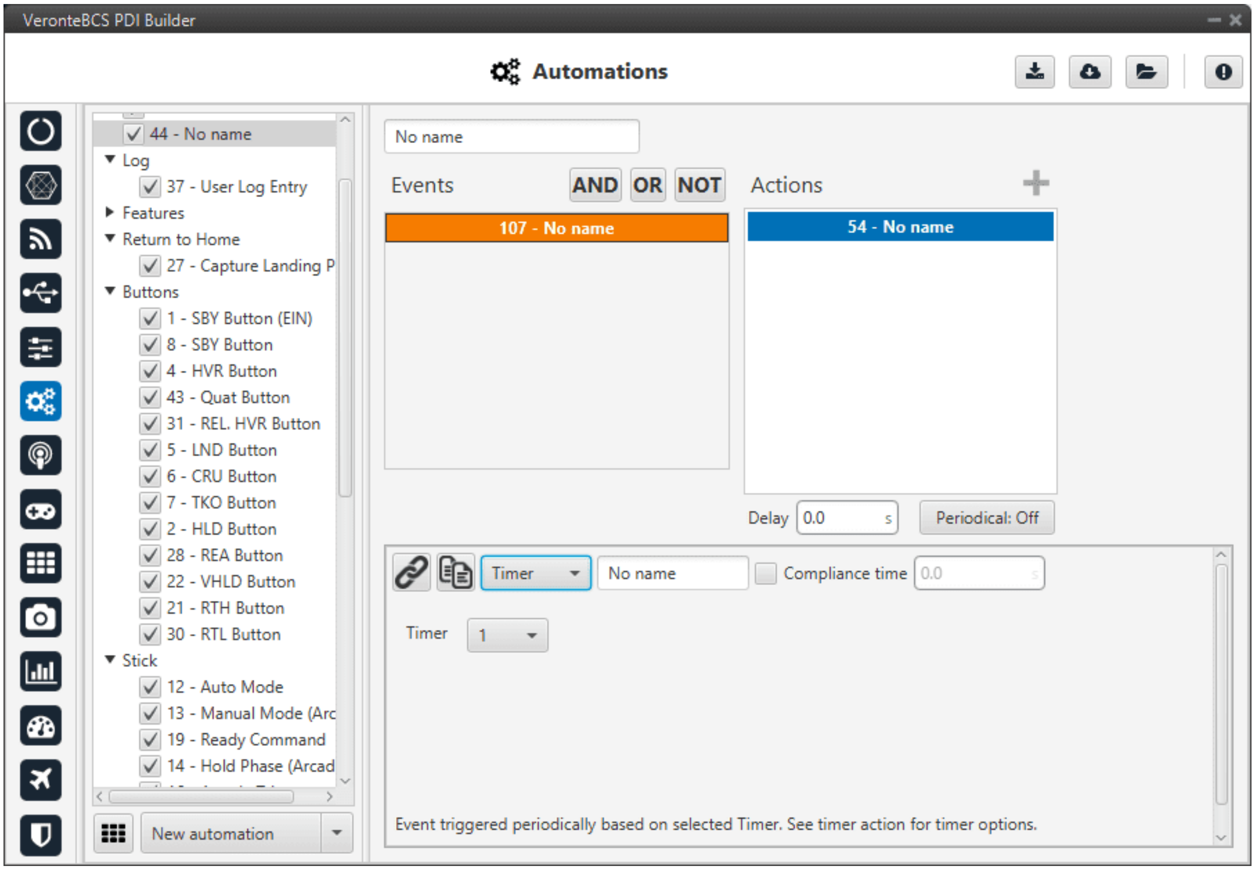
Fig. 79: **Route event**

The following options are available:

- **Activation**: The user can choose between two modes in this event.
  - **Fly to waypoint**: Triggers the action when the platform is flying towards that waypoint (**patch**).
  - **Mark achieved**: Triggers the action when the vehicle has reached the selected mark.
- **Selected marks/points**: To select a mark/waypoint (patch), first define it in the *Operation elements section of the UI menu*.
- **Icon** and **color**: It is possible to change the appearance of the waypoint, selecting an icon from the icon list and a color, so the user can identify easily the waypoint linked to that automation.

2.6.2.1.7 Timer

This event will check the status of the timer selected in the menu. That timer should have been configured previously on the action side of another automation (action type *Periodical*).



Delay 0.0 s

Timer 1

Event triggered periodically based on selected Timer. See timer action for timer options.

Fig. 80: **Timer event**

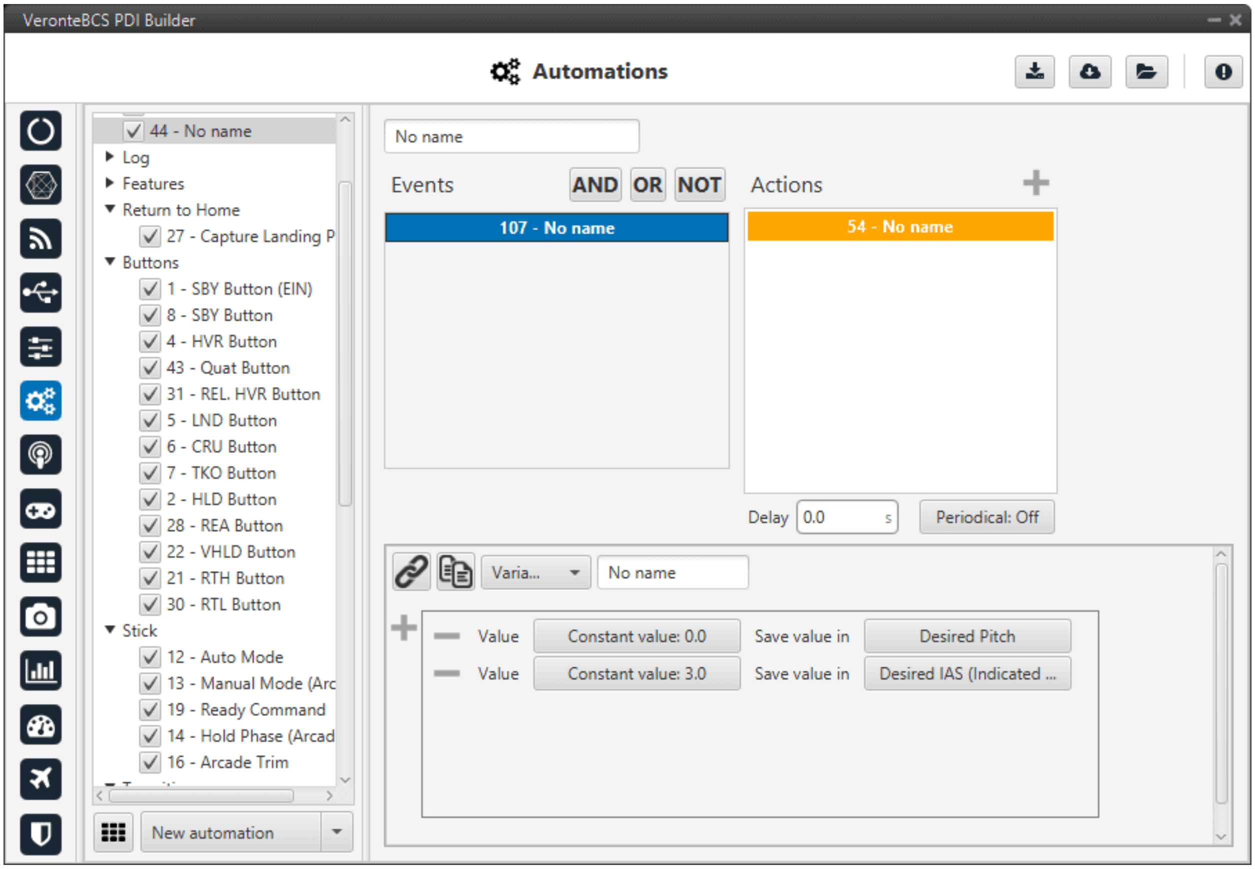
In **Timer** is selected the number that identifies the timer (previously created with the *periodical action*) that is evaluated in this event.

For example, if it is desired to take a photo 10 seconds after the takeoff, two automations are required:

1. The first automation should have the **event of Phase Take Off**, with the correspondent **Periodical action** that will start a timer that lasts 10 seconds.
2. The second one should have a **Timer event** with the timer previously created and then an **action to take a photo** when the timer event is triggered.

2.6.2.1.8 Variable

This event is triggered when a variable selected is between a range established.



Name: No name

Delay: 0.0 s

Action name: No name

Value	Constant value: 0.0	Save value in	Desired Pitch
Value	Constant value: 3.0	Save value in	Desired IAS (Indicated ...

Fig. 81: **Variable event**

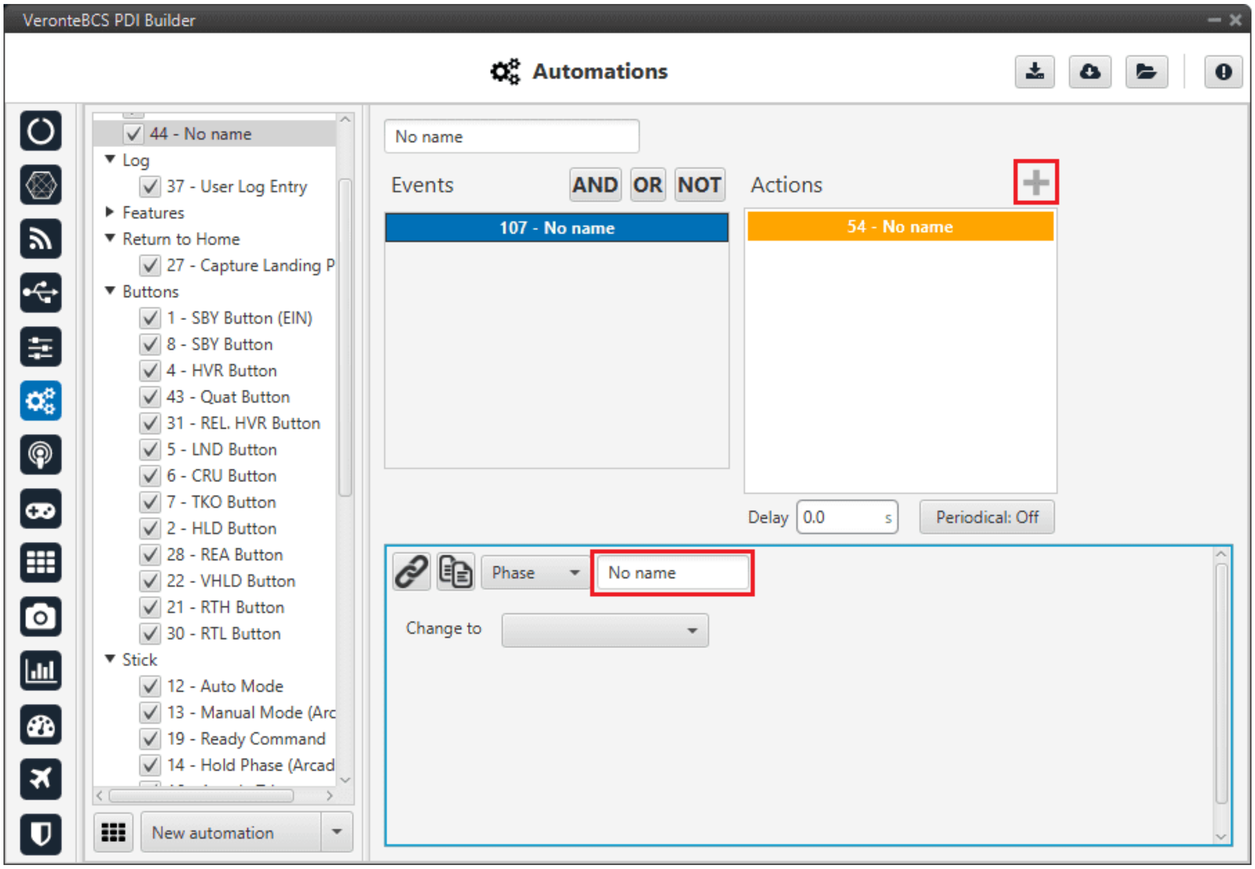
- **Variable**: The user can select the variable to be evaluated.
- **Max/Min**: Maximum and minimum values of the threshold are established here. Custom threshold can be established by clicking on the ⚙ icon.
- **Invert range**: This option will change the interval (the blue area will be gray, and the gray one will be blue).

As an example consider the event of the figure. With that parameters, the event is triggered when the IAS is between 5 and 20 meters per second. If the invert range option is unchecked, the event will be triggered when the IAS is lower that 5 m/s or greater than 20 m/s.

2.6.2.2 Actions

An action is something that will be performed when the event (or group of events) has been accomplished. The actions box contains all the actions created.

The user can also rename the event with the name of his choice.



No name

Delay 0.0 s

Periodical: Off

Phase: No name

Fig. 82: **Actions menu**

When creating a new event it is possible to choose from one of the **previously created** on the system or to **create** a **new** one.

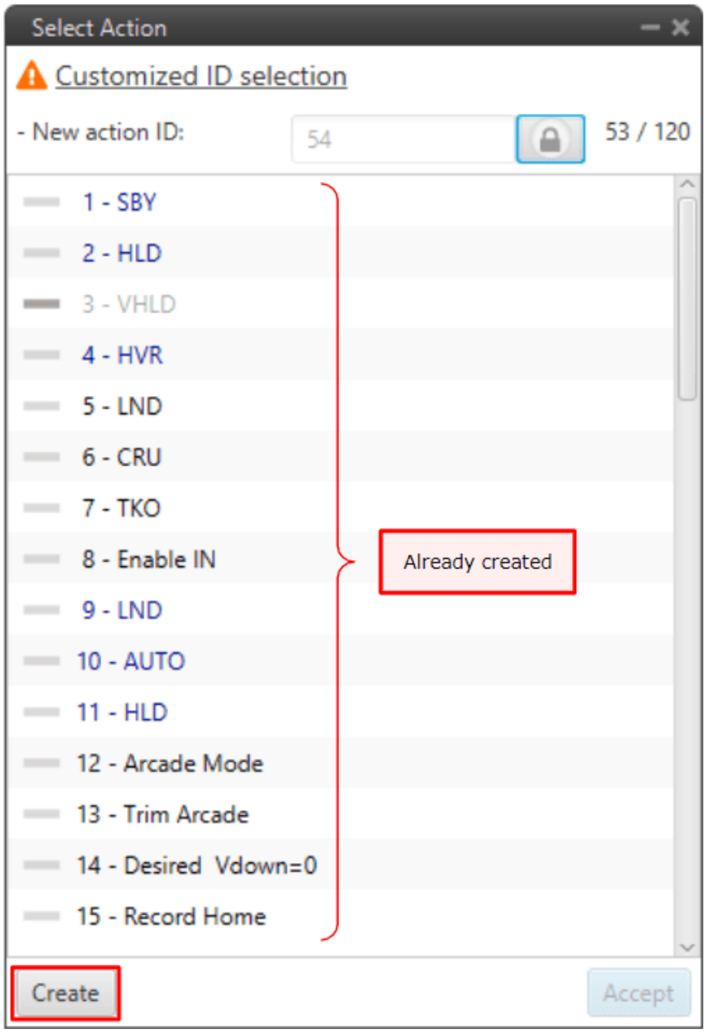
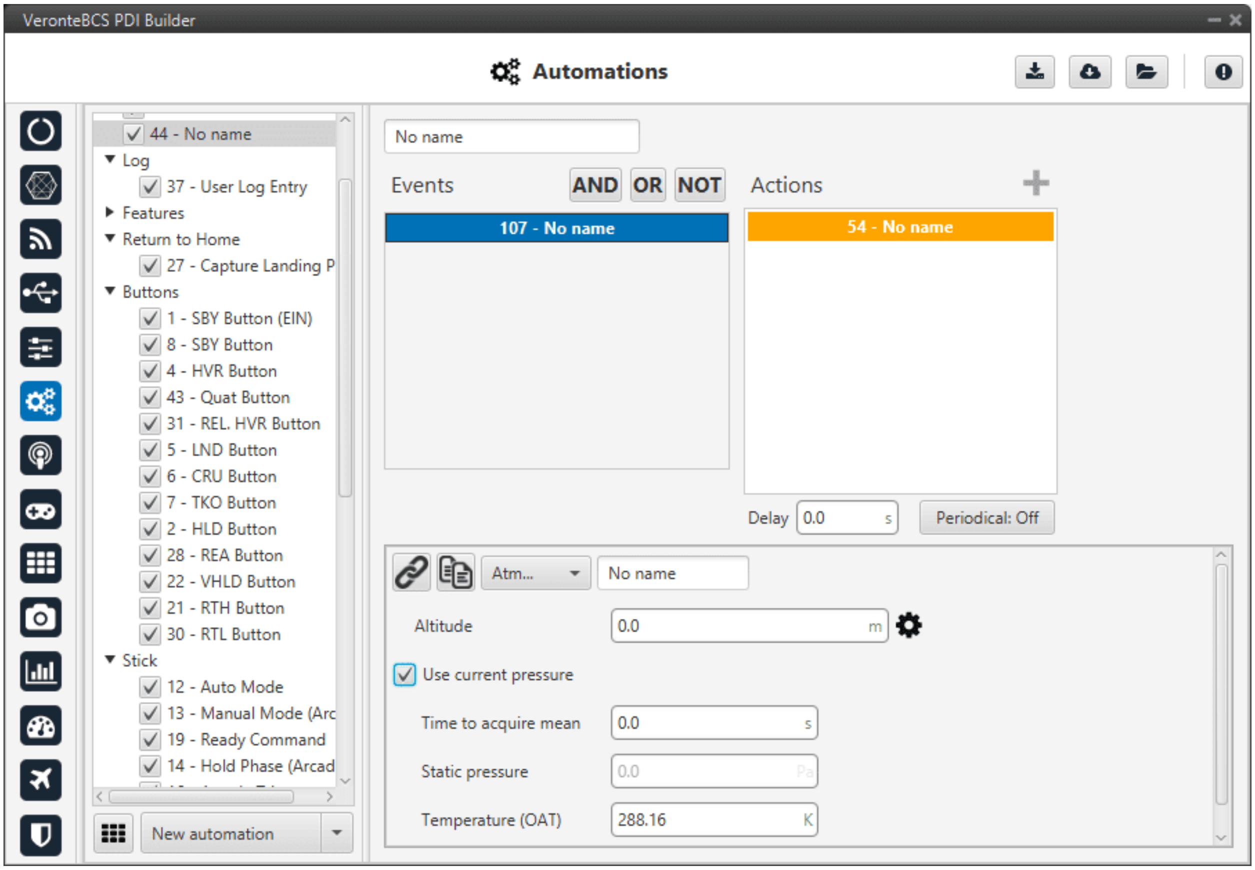


Fig. 83: **New action**

Below are present the different types of actions that can be created.

2.6.2.2.1 Atmosphere Calibration

This action allows the atmosphere calibration in the same way as shown in the **Operational panel of Veronte Ops** (for more information about atmosphere calibration click Veronte Ops manual).



☑ 44 - No name
▼ Log
☑ 37 - User Log Entry
▶ Features
▼ Return to Home
☑ 27 - Capture Landing P
▼ Buttons
☑ 1 - SBY Button (EIN)
☑ 8 - SBY Button
☑ 4 - HVR Button
☑ 43 - Quat Button
☑ 31 - REL. HVR Button
☑ 5 - LND Button
☑ 6 - CRU Button
☑ 7 - TKO Button
☑ 2 - HLD Button
☑ 28 - REA Button
☑ 22 - VHLD Button
☑ 21 - RTH Button
☑ 30 - RTL Button
▼ Stick
☑ 12 - Auto Mode
☑ 13 - Manual Mode (Arc
☑ 19 - Ready Command
☑ 14 - Hold Phase (Arcad

No name

Events: 107 - No name

Actions: 54 - No name

Delay 0.0 s

No name

Altitude 0.0 m

☑ Use current pressure

Time to acquire mean 0.0 s

Static pressure 0.0 Pa

Temperature (OAT) 288.16 K

Fig. 84: **Atmosphere Calibration action**

The following options can be configured:

- **Altitude**: Actual MSL altitude. The user must choose between entering this value manually or selecting a system variable.
- **User current pressure**: By enabling it, the static pressure will be read from the static pressure sensor during the specified time (**Time to acquire mean**).
- **Static pressure**: If the above option is not enabled, the actual static pressure should be specified manually.
- **Temperature (OAT)**: Outside air temperature.

2.6.2.2.2 Change active sensor

This option allows the change of the actual sensor used as dynamic pressure.

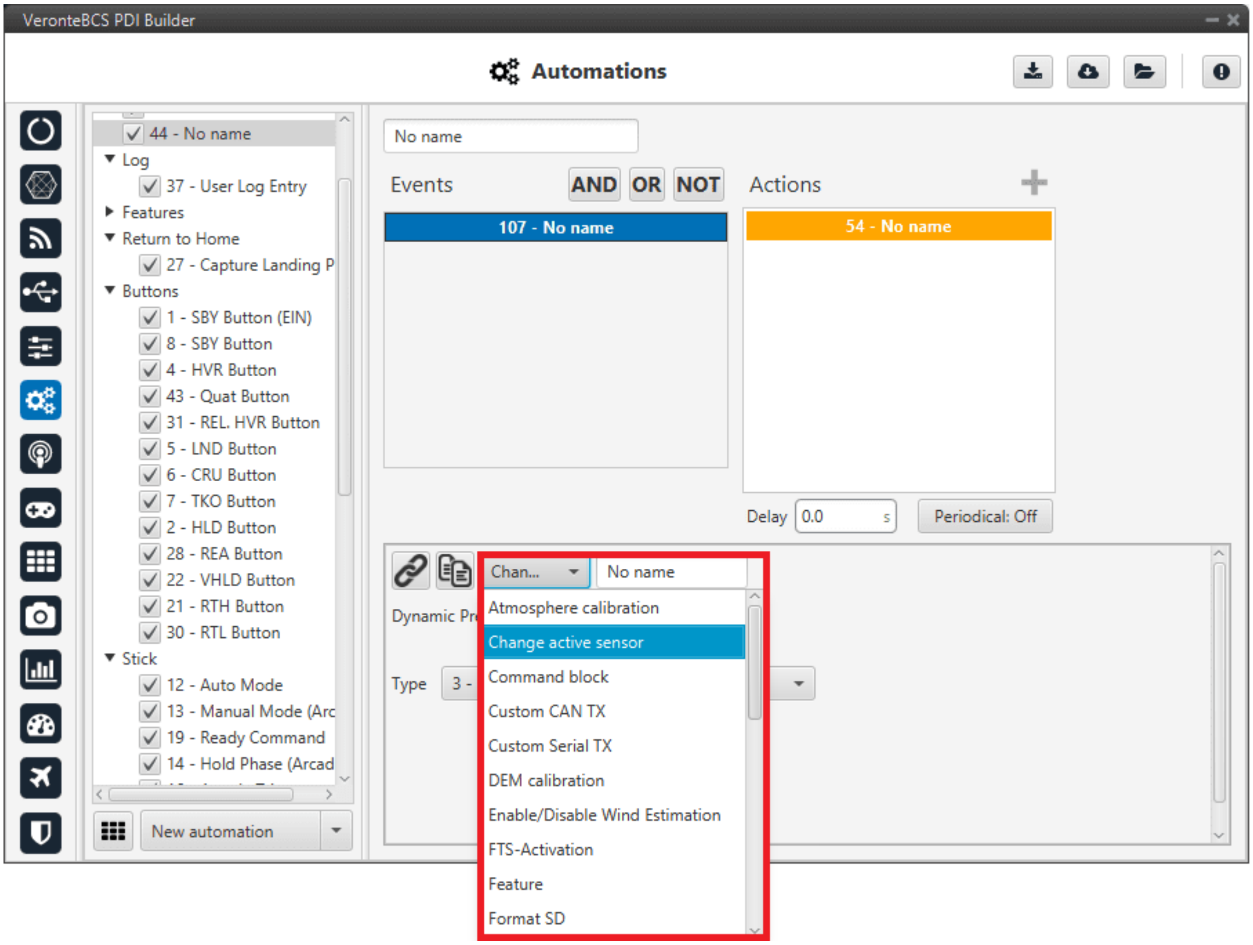
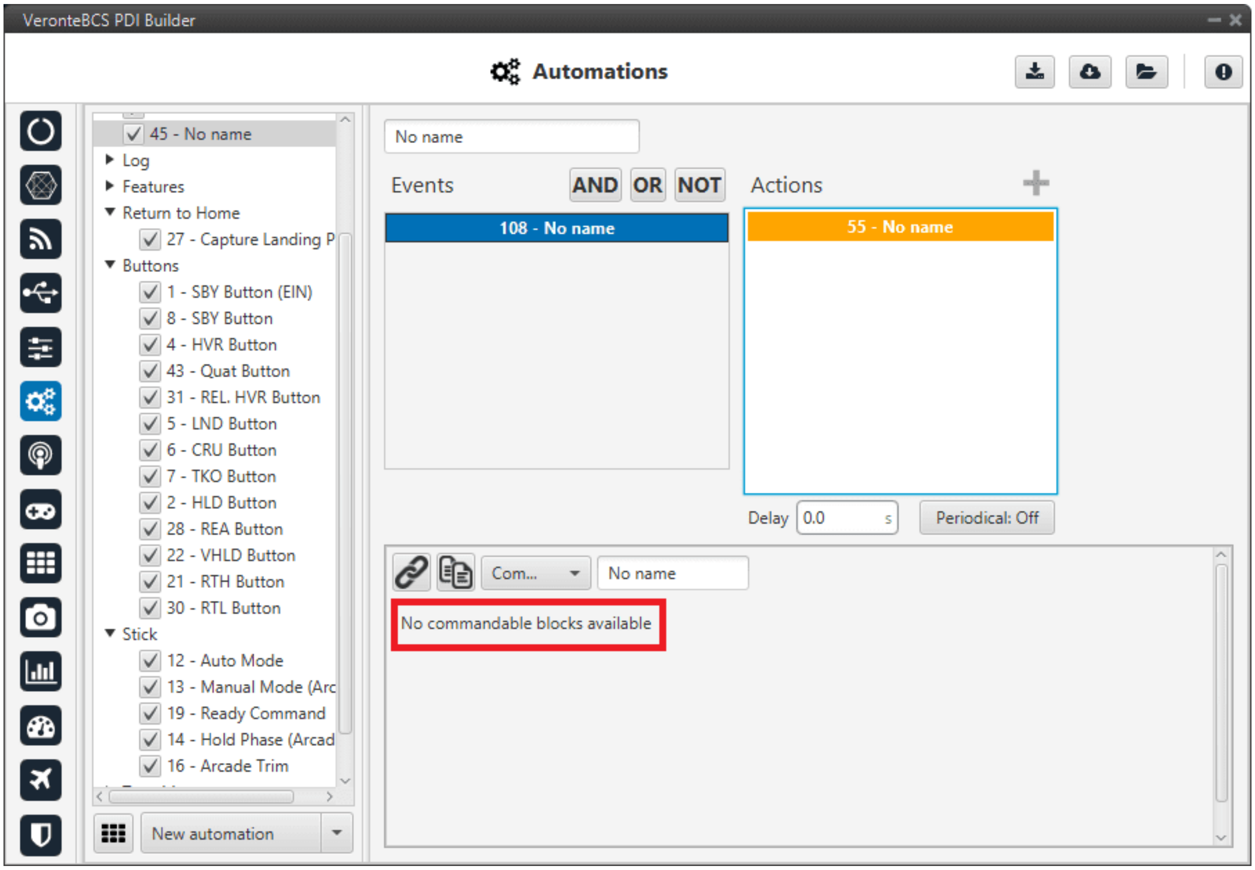


Fig. 85: **Change active sensor action**

2.6.2.2.3 Command block

This action allows the user to configure gimbal or trim the radio controller.

Note: This action is disabled by default when **BCS** is started. To activate it, the user have to create a gimbal block or an arctrim block (see more information about it in *Block Programs*).

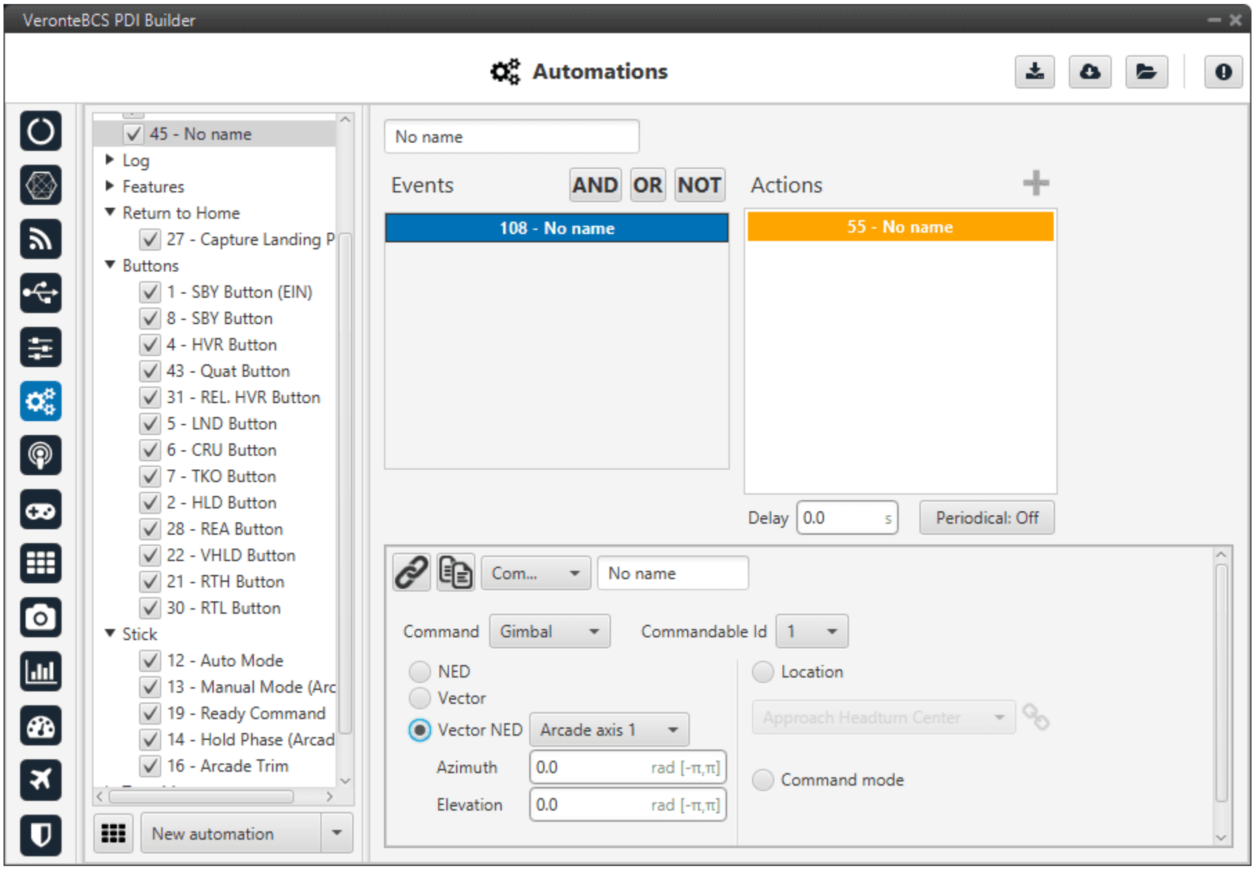


No commandable blocks available

Fig. 86: **Command block action**

Gimbal

When this action is triggered, the gimbal control is enabled. There are several control modes that are explained below.



No name

Delay 0.0 s

Periodical: Off

No name

Command: Gimbal

Commandable Id: 1

○ NED

○ Vector

● Vector NED: Arcade axis 1

Azimuth: 0.0 rad [-π,π]

Elevation: 0.0 rad [-π,π]

○ Location

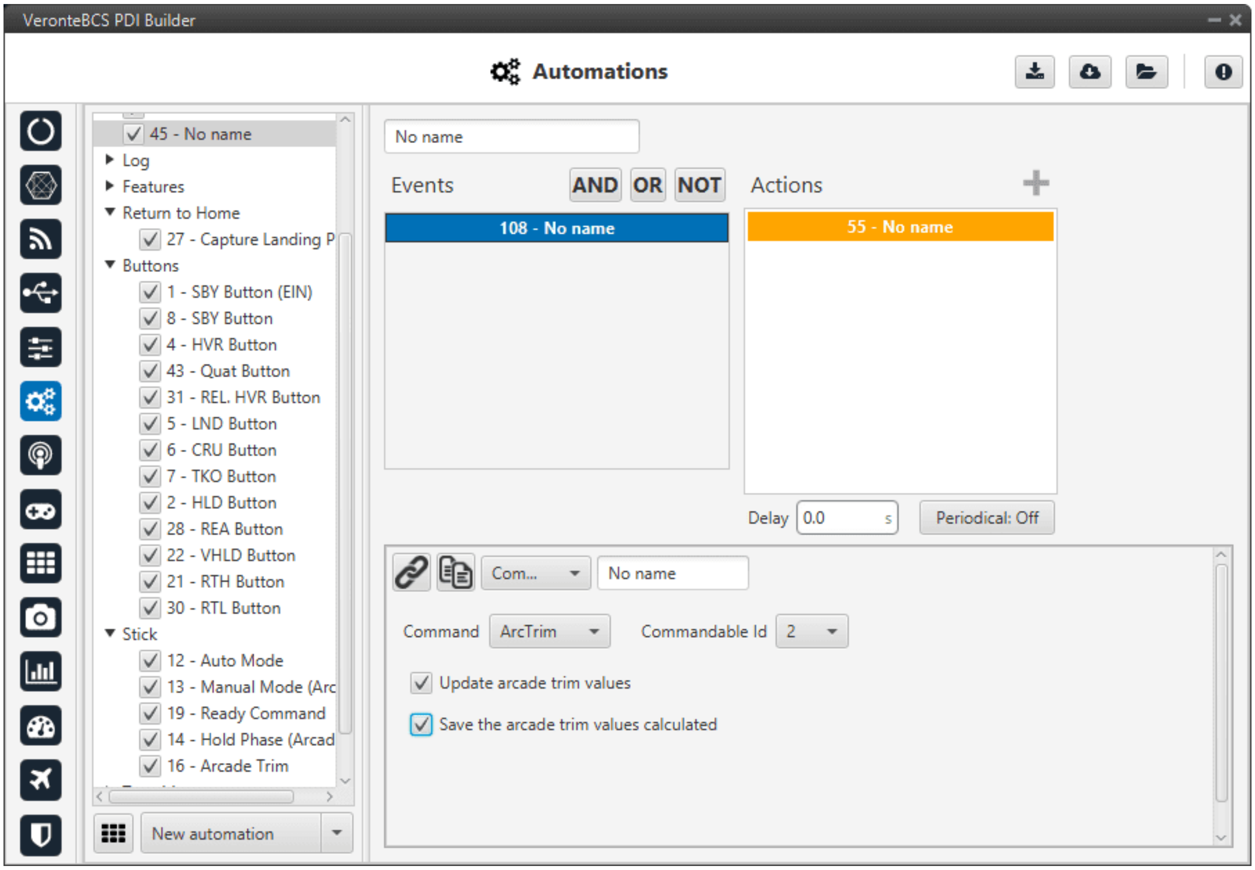
○ Command mode

Fig. 87: **Command block action - Gimbal**

- **Commandable Id**: Id of the commandable Gimbal block (users can look up this Id in the block to be commanded, in *Block Programs* section).
- **NED**: Control using NED axis, defining the initial position through azimuth and elevation.
- **Vector**: This control uses aircraft body axis. The initial position is defined through roll and tilt.
- **Vector NED**: In this case the axis should have been defined in *Arcade Axis*.
- **Location**: The gimbal will point towards the projection on the ground at the specified point.
- **Command mode**: Gimbal control is done externally, e.g. via VCP commands.

ArcTrim

This action trims the radio controller, i.e sets as zero the current sticks positions.



No name

Events: 108 - No name

Actions: 55 - No name

Delay 0.0 s

Periodical: Off

Com... No name

Command ArcTrim

Commandable Id 2

☑ Update arcade trim values

☑ Save the arcade trim values calculated

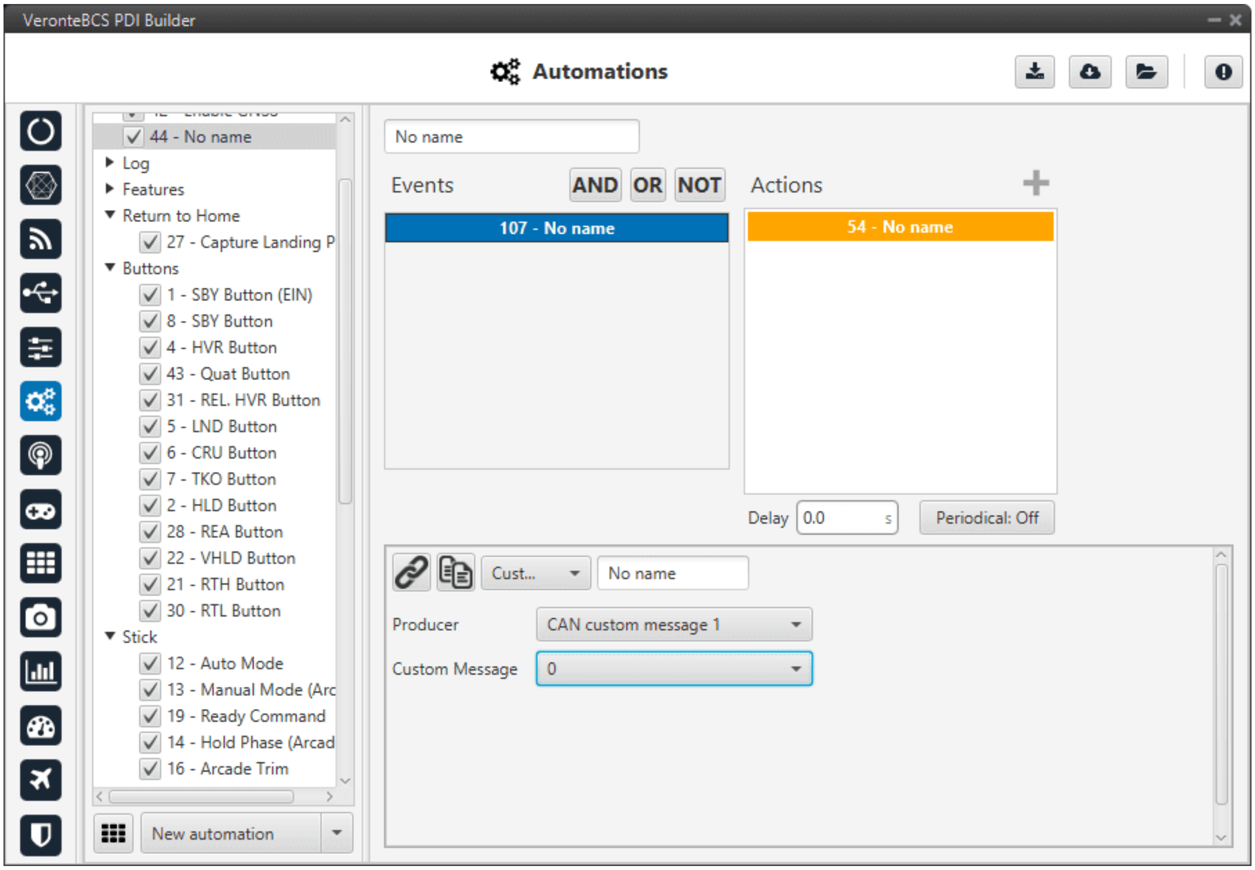
Fig. 88: **Command block action - ArcTrim**

- **Commandable Id**: Id of the commandable ArcTrim block (users can look up this Id in the block to be commanded, in *Block Programs* section).
- **Update the arcade trim values**: If this option si enabled, the stick is trimmed but not saved in the configuration. This means that **if BCS is restarted the trimming is lost**.
- **Save the arcade trim values calculated**: Trim values are **stored** for future flights.

2.6.2.2.4 Custom CAN TX

When this action is triggered, a previously configured CAN message is sent through the CAN bus. The message has to be configured in *Custom messages* section of the Input/Output menu.

> **Warning:** As this automation is used to send a single message on demand, in its configuration in **Custom Messages**, the user has to set its **period to -1**. This way, this **message will only be sent when this action is triggered**.



No name

Delay 0.0 s

Periodical: Off

No name

Producer: CAN custom message 1

Custom Message: 0

Fig. 89: **Custom CAN TX action**

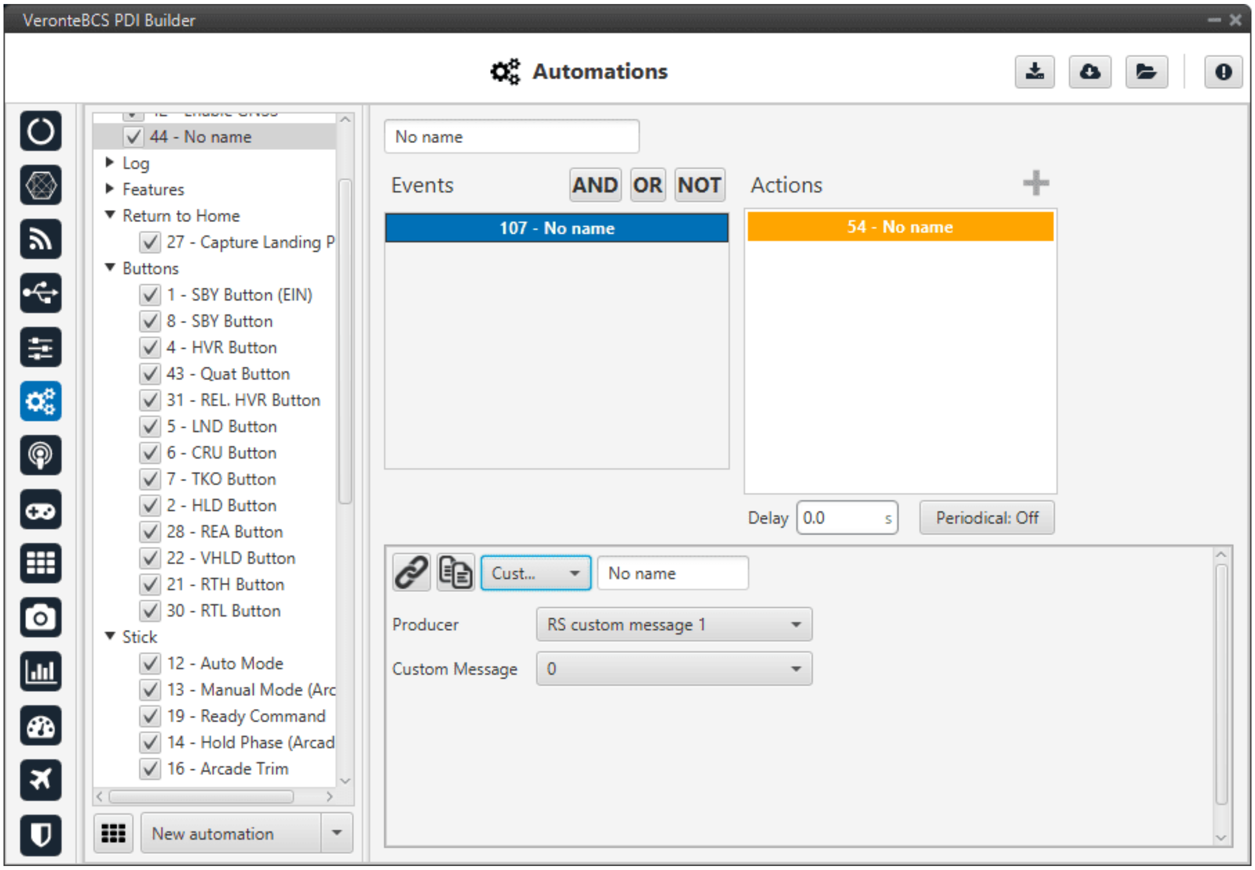
The two parameters to configure in this action are:

- **Producer**: The user has to specify where the custom message is located: CAN custom message 1, 2 or 3.
- **Custom message**: The number of the custom messages that will be sent.

2.6.2.2.5 Custom Serial TX

When this action is triggered, a previously configured serial message is sent through the serial port (RS232 or RS485). The message has to be configured in *Serial custom messages* section of the Input/Output menu.

> **Warning:** As this automation is used to send a single message on demand, in its configuration in **Custom Messages**, the user has to set its **period to -1**. This way, this **message will only be sent when this action is triggered**.



Producer: RS custom message 1

Custom Message: 0

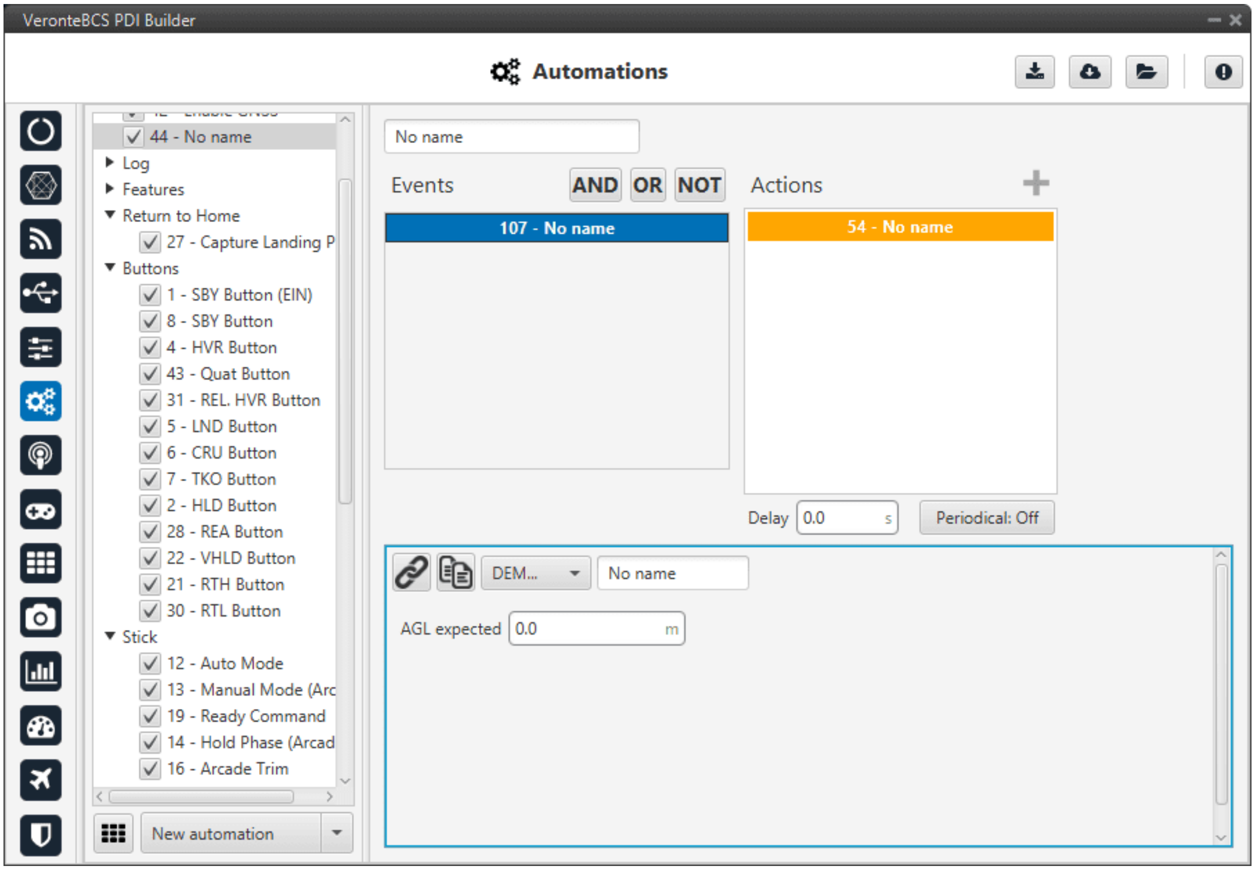
Fig. 90: **Custom Serial TX action**

The two parameters to configure in this action are

- **Producer**: The user has to specify where the custom message is located: RS custom message 1, 2 or 3.
- **Custom message**: The number of the custom messages that will be sent.

2.6.2.2.6 DEM calibration

This option allows the calibration of the digital elevation model by setting the **actual AGL value** in the same way as shown in the **Operational panel of Veronte Ops** (for more information about DEM calibration click Veronte Ops manual).



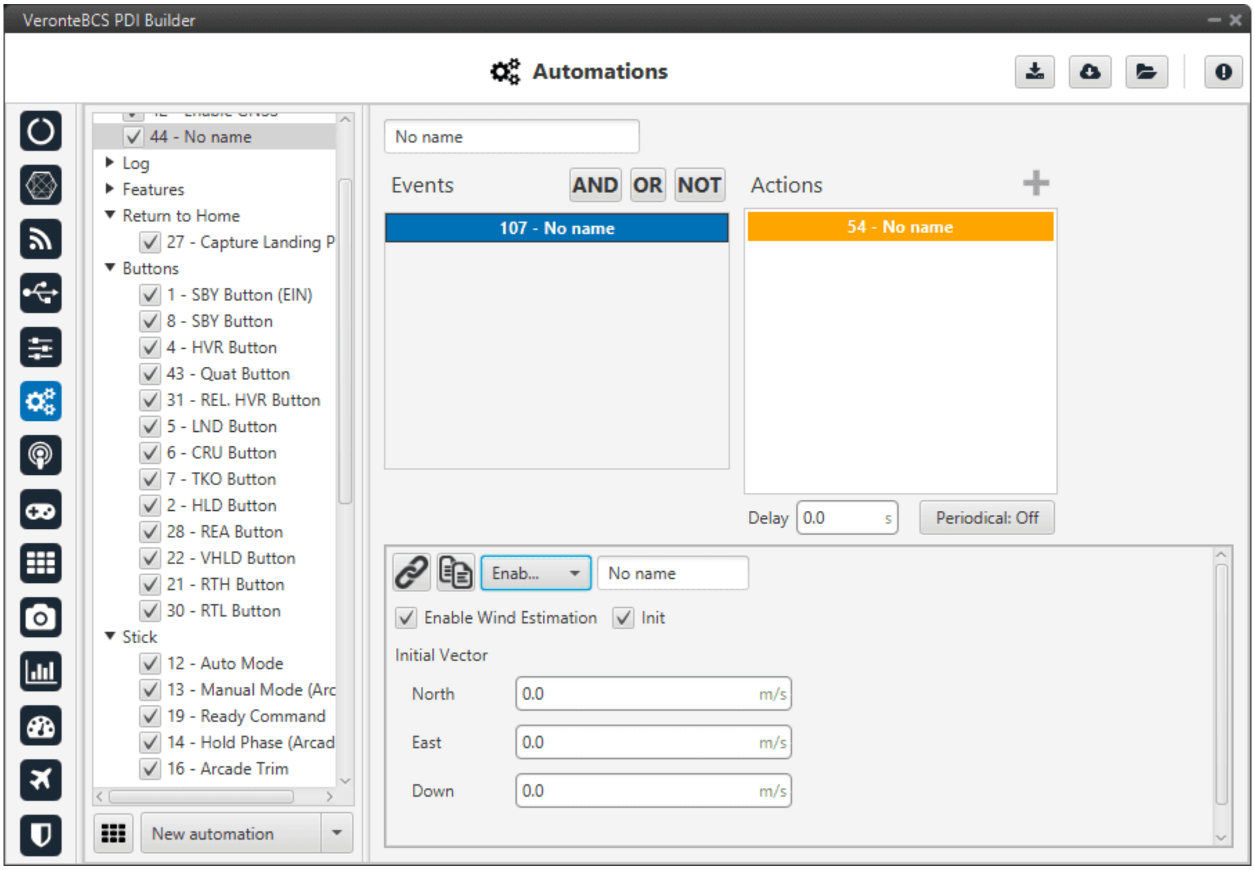
Delay 0.0 s

AGL expected 0.0 m

Fig. 91: **DEM calibration action**

2.6.2.2.7 Enable/Disable Wind Estimation

This action allows the user to enable or disable the wind estimation.



No name

Delay 0.0 s

No name

☑ Enable Wind Estimation ☑ Init

Initial Vector

North 0.0 m/s

East 0.0 m/s

Down 0.0 m/s

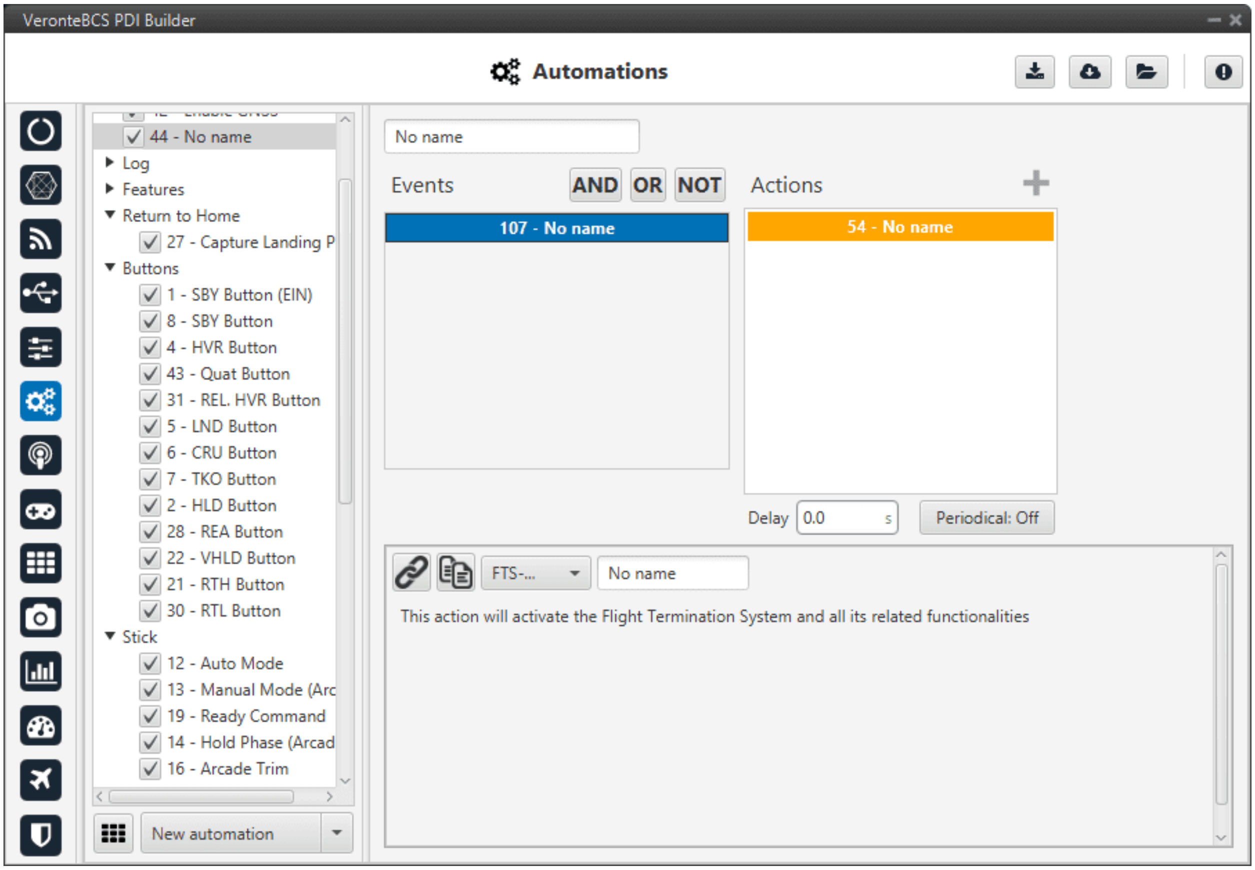
Fig. 92: **Enable/Disable Wind Estimation action**

The following parameters can be configured:

- **Enable Wind Estimation**: Enabled/Disabled.
- **Init**: By enabling it, an initial wind vector can be set to a faster convergence of the estimation.
  - **North, East, Down**: Inital wind vector.

2.6.2.2.8 FTS Activation

This action activate the flight termination system (FTS) bit.



Name: No name

Delay: 0.0 s

Action name: No name

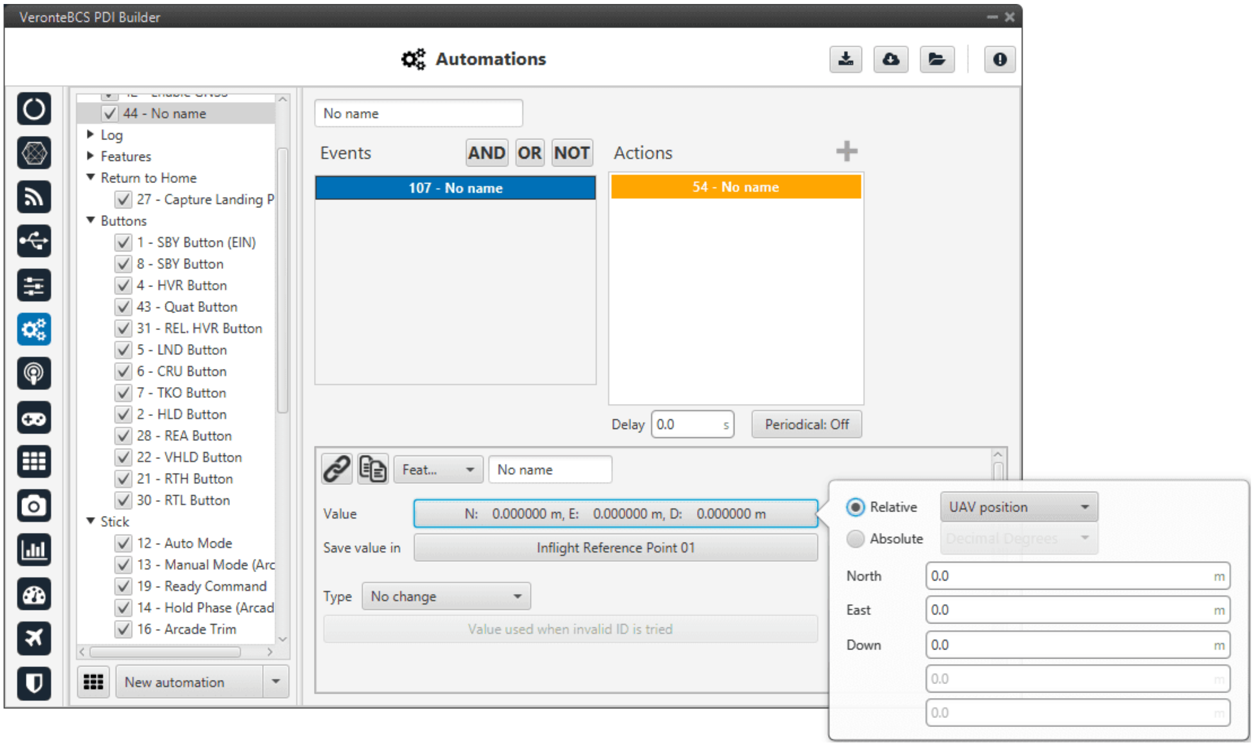
This action will activate the Flight Termination System and all its related functionalities

Fig. 93: **FTS Activation action**

In a 4x autopilot, when two or more autopilots activate their FTS the arbiter can activate a safe system such as a parachute.

2.6.2.2.9 Feature

When this action is triggered, a position is stored in the desired variable. This position can be absolute or relative (in the figure below the current position of the aircraft would be saved):



Delay 0.0 s

Periodical: Off

Value N: 0.000000 m, E: 0.000000 m, D: 0.000000 m

Save value in Inflight Reference Point 01

Type No change

● Relative UAV position

○ Absolute

North 0.0 m

East 0.0 m

Down 0.0 m

Fig. 94: **Feature action**

The following options should be configured:

- **Value**: Specified the position to be stored, this position can be **absolute** or **relative**.
  - **Absolute**: The coordinates can be set in UTM, MGRS, Decimal Degrees or Degress º ” ‘. They are indicated through the latitude, longitude and altitude (being possible to define this last one with respect to the ellipsoid, WGS84, to the sea level, MSL or to the ground, AGL).
  - **Relative**: In this case, the position of the point is relative to another point. That point could be any platform fitted with an **1x Autopilot**.
- **Save value in**: The position has to be saved in an ‘Inflight Reference Point’.
- **Types**: There are 2 types of features:
  - **Fixed**: Once the point has been generated it remains fixed.
  - **No change**: If the **point** has been created **relative**, it remains relative all the time.

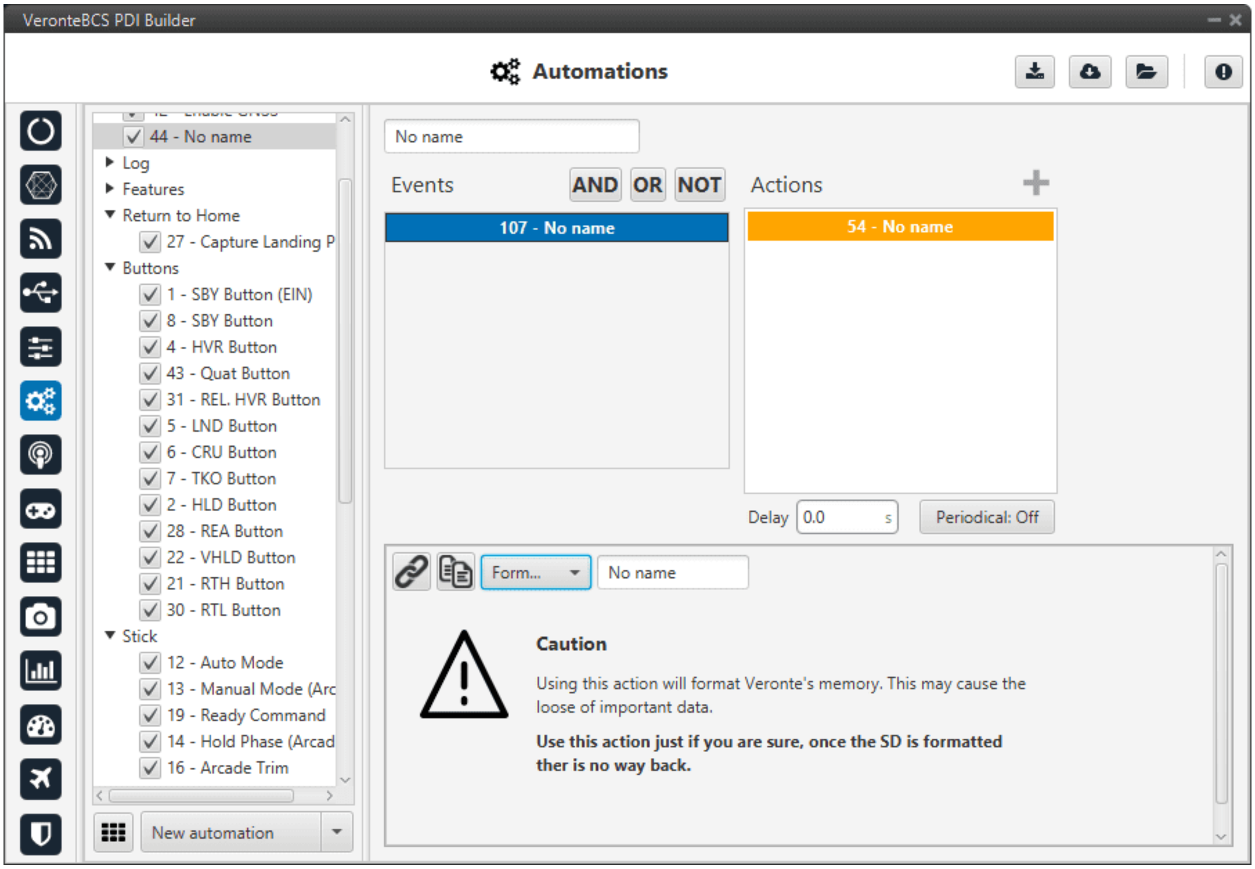
    Note: This option only appears when the position has been previously defined as **Relative**.

This action is very useful for storing the take-off point for later landing at the same place.

2.6.2.2.10 Format SD

> **Danger:** This action will have irreversible effects on the **BCS**. Formatting the SD card will delete important and mandatory files for the correct functioning of **BCS**. In order to recover a formatted unit, please contact the support team (create a ticket in the customer's **Joint Collaboration Framework**; for more information, see Tickets section of the JCF manual).

This action will format the SD card, deleting the configuration and flight logs from it.



Caution

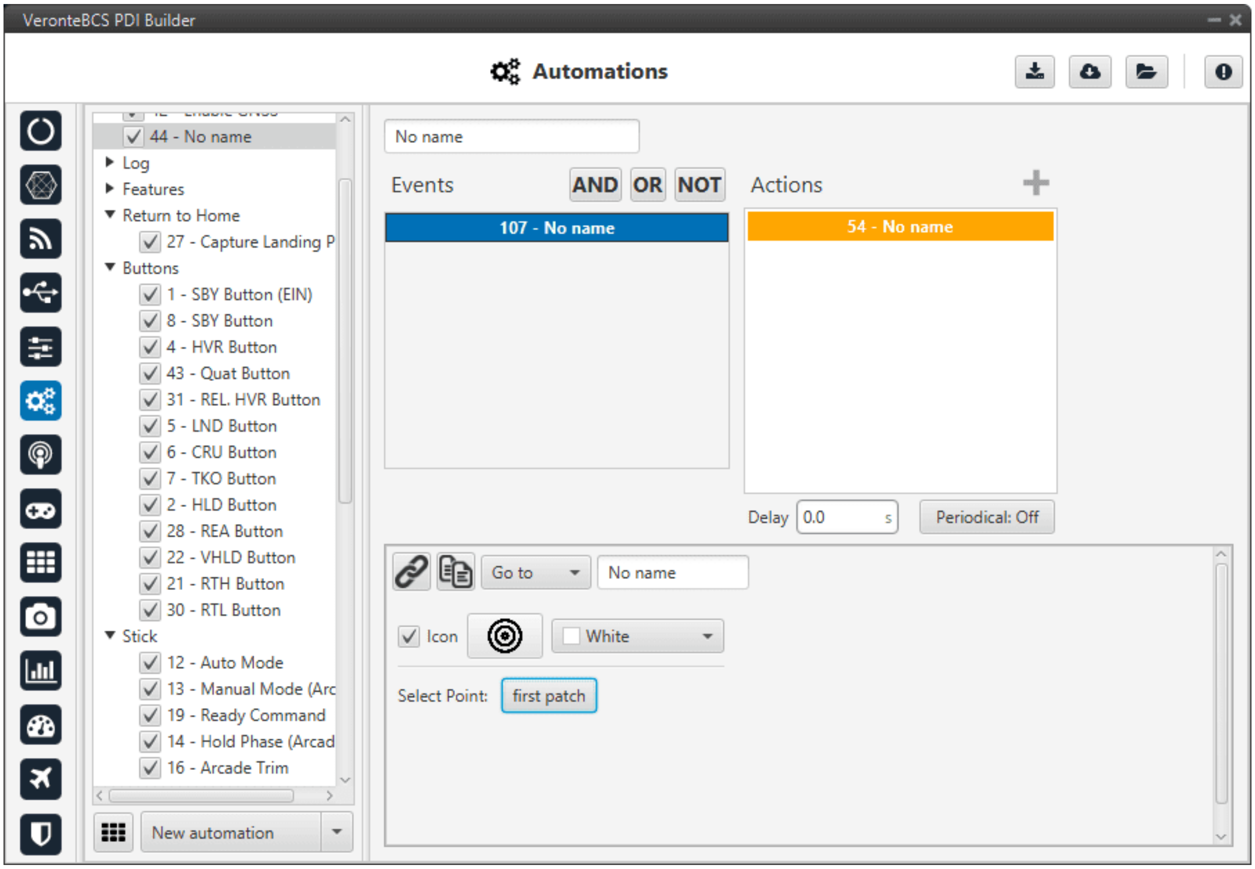
Using this action will format Veronte's memory. This may cause the loose of important data.

Use this action just if you are sure, once the SD is formatted ther is no way back.

Fig. 95: **Format SD action**

2.6.2.2.11 Go to

This action is used to make the aircraft go to a patch created by the user with the mission toolbar of **Veronte Ops**. For more information about the mission toolbar, take a look at the Veronte Ops manual.



No name

Delay 0.0 s

Periodical: Off

Go to ▾ No name

☑ Icon ◎ □ White ▾

Select Point: first patch

Fig. 96: **Go to action**

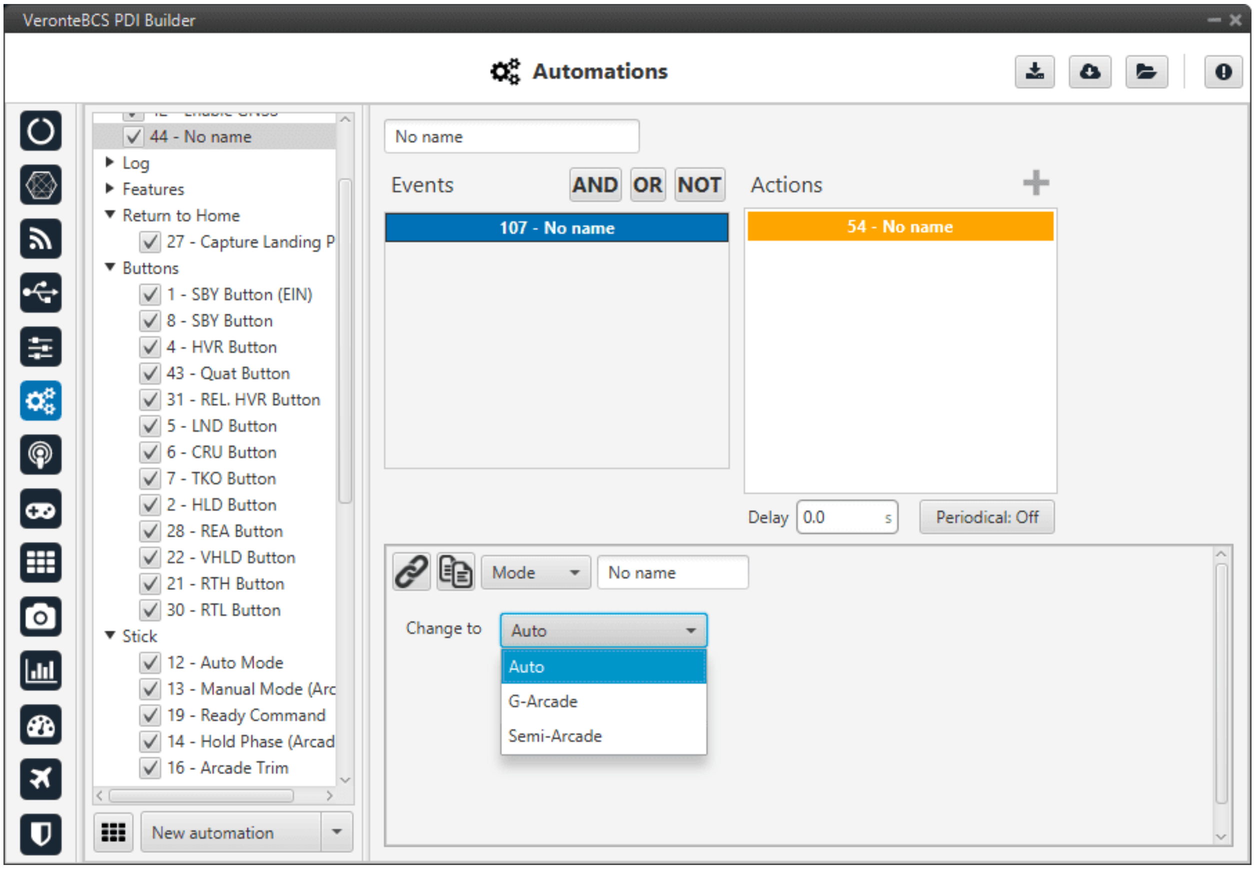
In this action two parameters can be configured:

- **Select Point**: To select point (**patch**), first define it in the *Operation elements section of the UI menu*.
- **Icon** and **color**: It is possible to change the appearance of the point, selecting an icon from the icon list and a color, so the user can identify easily the point linked to that automation.

Once the action is triggered, the vehicle will go to that patch. If the patch is on a route, the vehicle will follow the selected patch and then it will continue the route going to its adjacent.

2.6.2.2.12 Mode

The flight mode is changed to the one specified in this option.



No name

Delay 0.0 s

Mode No name

Change to Auto

Fig. 97: **Mode action**

These modes have been created previously. See section *Modes*, for more information about creating modes.

2.6.2.2.13 Obstacle avoidance

This action enables avoidance of any obstacle previously created in **Veronte Ops** (for more information, see Veronte Ops manual).

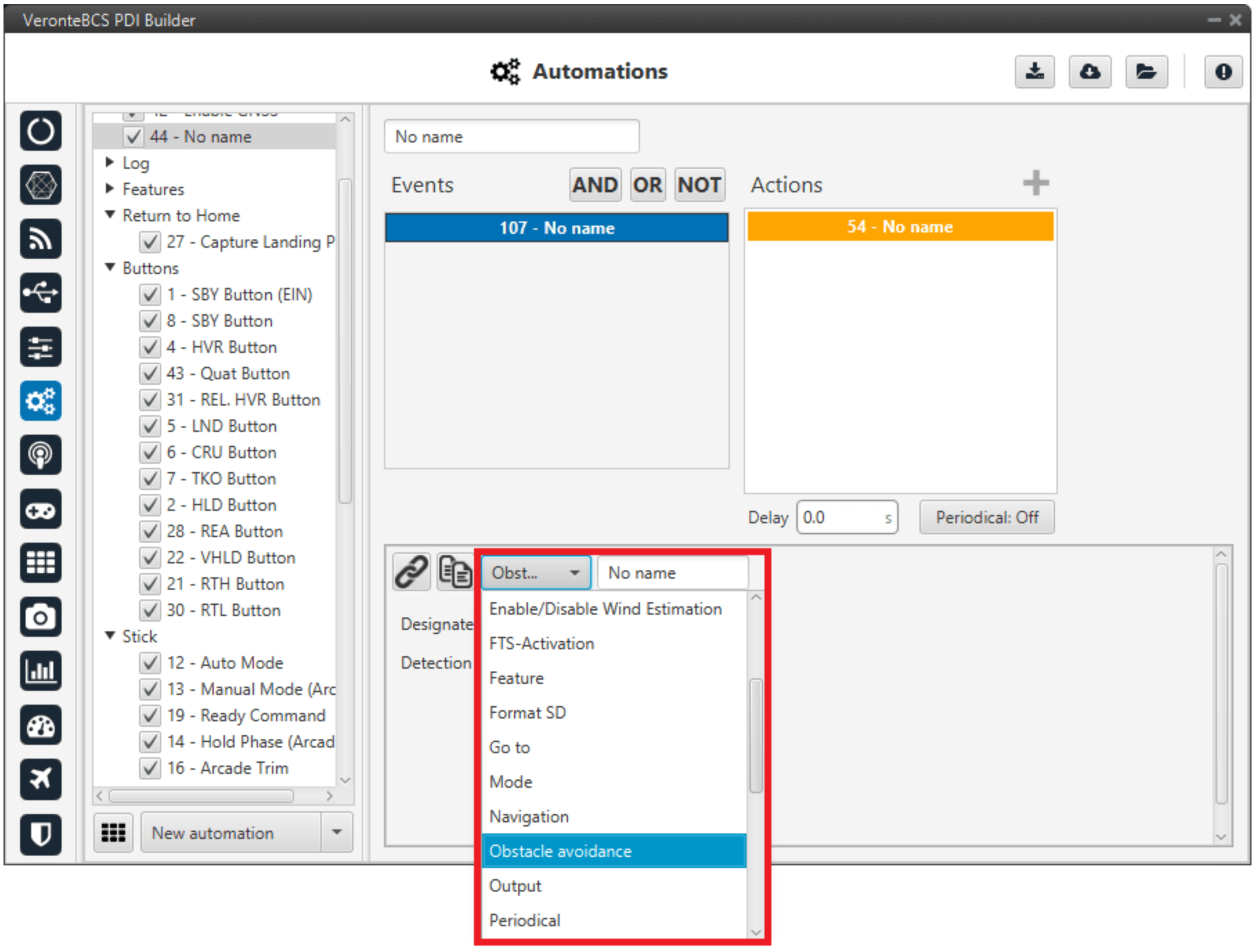


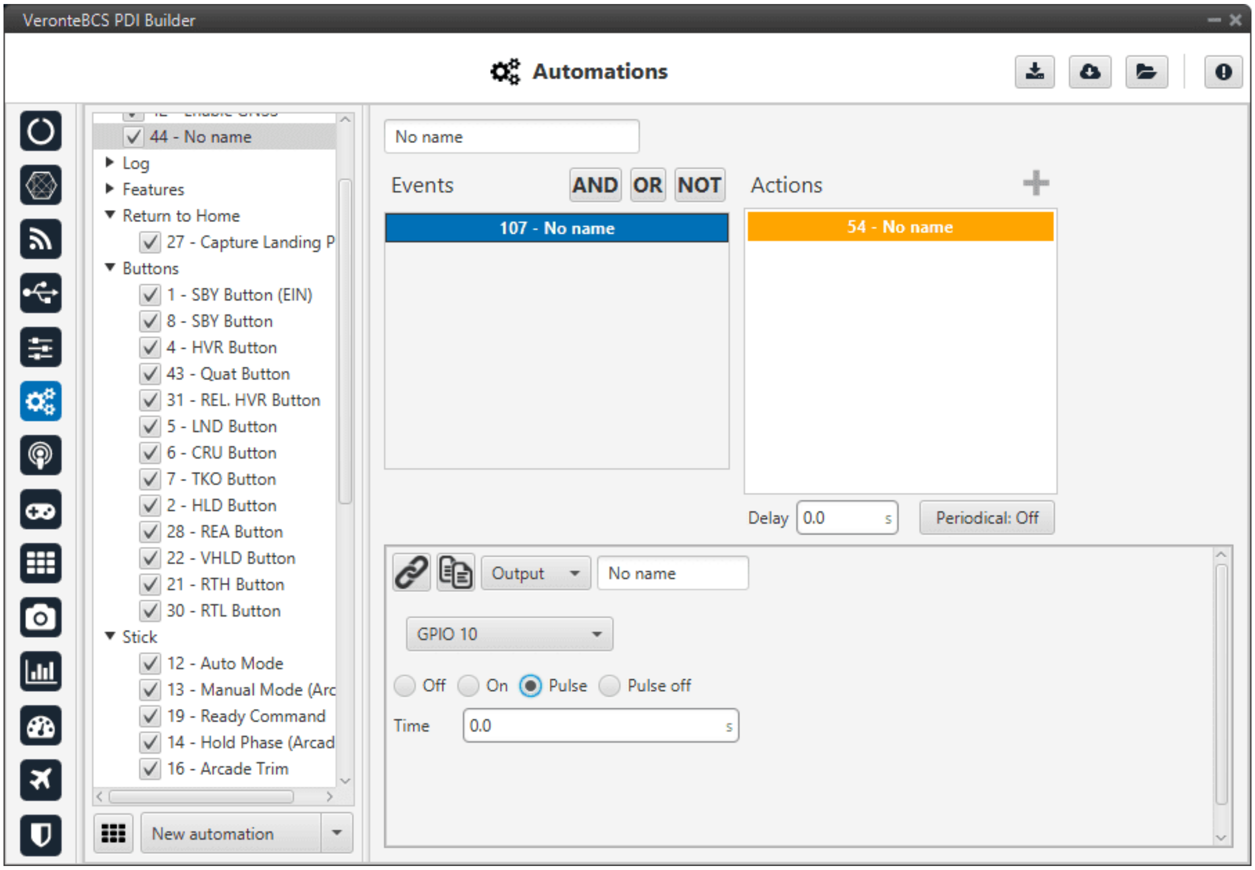
Fig. 98: **Obstacle avoidance action**

Two types of obstacles can be enabled:

- **Designated on map**: Those obstacles that appear on the map, obstacle zones and aircraft received by ADS-B.
- **Detection sensor**: Obstacles previously defined in **Veronte Ops**.

2.6.2.2.14 Output

This action is used to set an output value in a GPIO pin. The output pin must have been configured as a **GPIO output** (visit section *GPIO*).



No name

107 - No name

54 - No name

Delay 0.0 s

Periodical: Off

Output

No name

GPIO 10

○ Off ○ On ● Pulse ○ Pulse off

Time 0.0 s

Fig. 99: **Output action**

The user can select, in the drop-down menu, between a GPIO output or a virtual output. The virtual option works like a normal GPIO output, but physically this output is not in the **BCS**. It is used, for example, with a CEX.

There are four possible output signals:

- **Off**: Provides continuous 0V output.
- **On**: Provides continuous 3.3V output.
- **Pulse**: Provides 3.3V for the specified time and after that 0V.
- **Pulse off**: Provides 0V for the specified time and after that 3.3V.

2.6.2.2.15 Periodical

This action is used to set a timer during a flight operation.

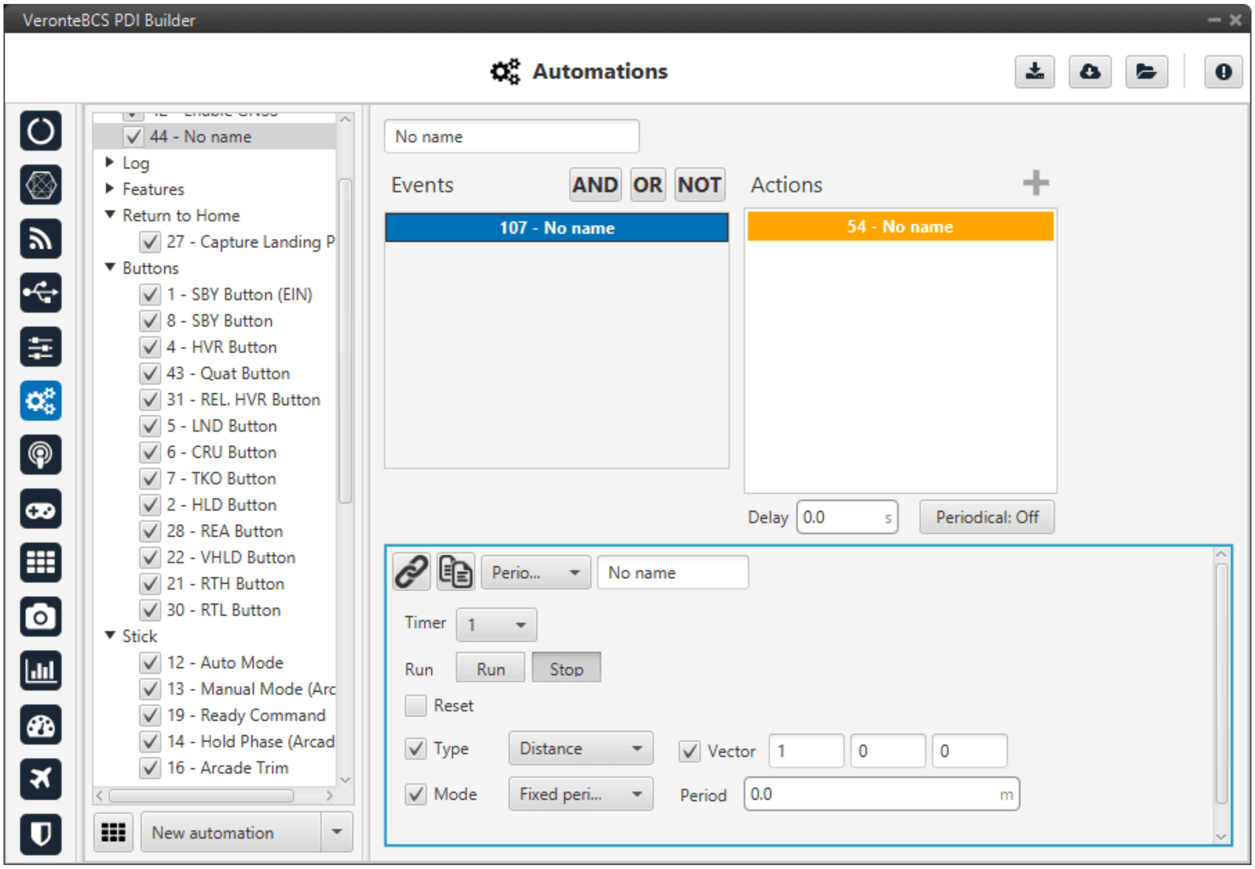


Fig. 100: **Periodical action**

The following parameters are configured:

- **Timer**: This parameter is an identifier for the timer, so it can be used in an event for another automation.
- **Run**: The timer will start.
- **Stop**: The timer will be stopped. Another automation should be created to run it again.
- **Reset**: When this action is active, the timer is reset to zero before starting to measure.
  - **Stop + Resert**: The timer will be stopped and set back to zero.
- **Type**: These available options have been explained in *Automations*.
- **Mode**: The difference between fixed delay and fixed period has been explained in *Automations*.

For a better understanding of this action, a set of examples are detailed below with possible combinations of the different options.

- **Run + Distance/Time + Continuous**: When the action is triggered, the timer will be started and will measure distance/time from that instant until the moment when the **BCS** is turned off (or until another automation acts on the same timer).

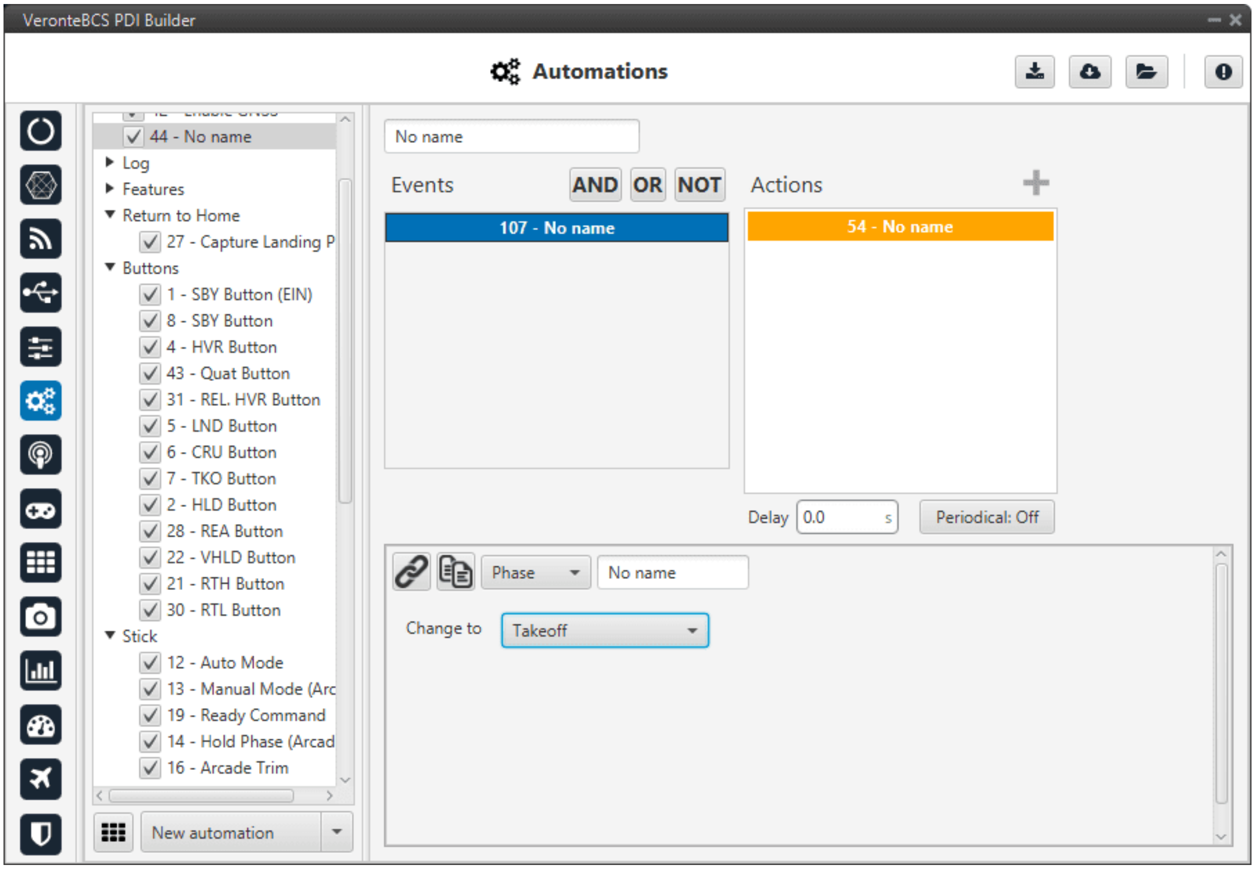
- **Run + Distance/Time + Fixed Delay/Period**: Once the action has been triggered, the timer will start to measure a distance/time. Each time the value indicated in Period is reached, the event linked to this timer (in another automation) will be triggered.

  For example, if the user wants to take a photo each 25 meters, in a first automation, the timer should have Distance in the Type option and 25 meters in Period, then in the second automation, an event of type Timer is created (and linked with the timer before created), so each time the timer reaches 25 meters the event will be triggered and the action will be carried out.

- **Distance + Vector**: The distance is measured in the direction indicated by the vector.

2.6.2.2.16 Phase

The flight phase is changed to the one selected in this action.



No name

107 - No name

54 - No name

Delay 0.0 s

Phase No name

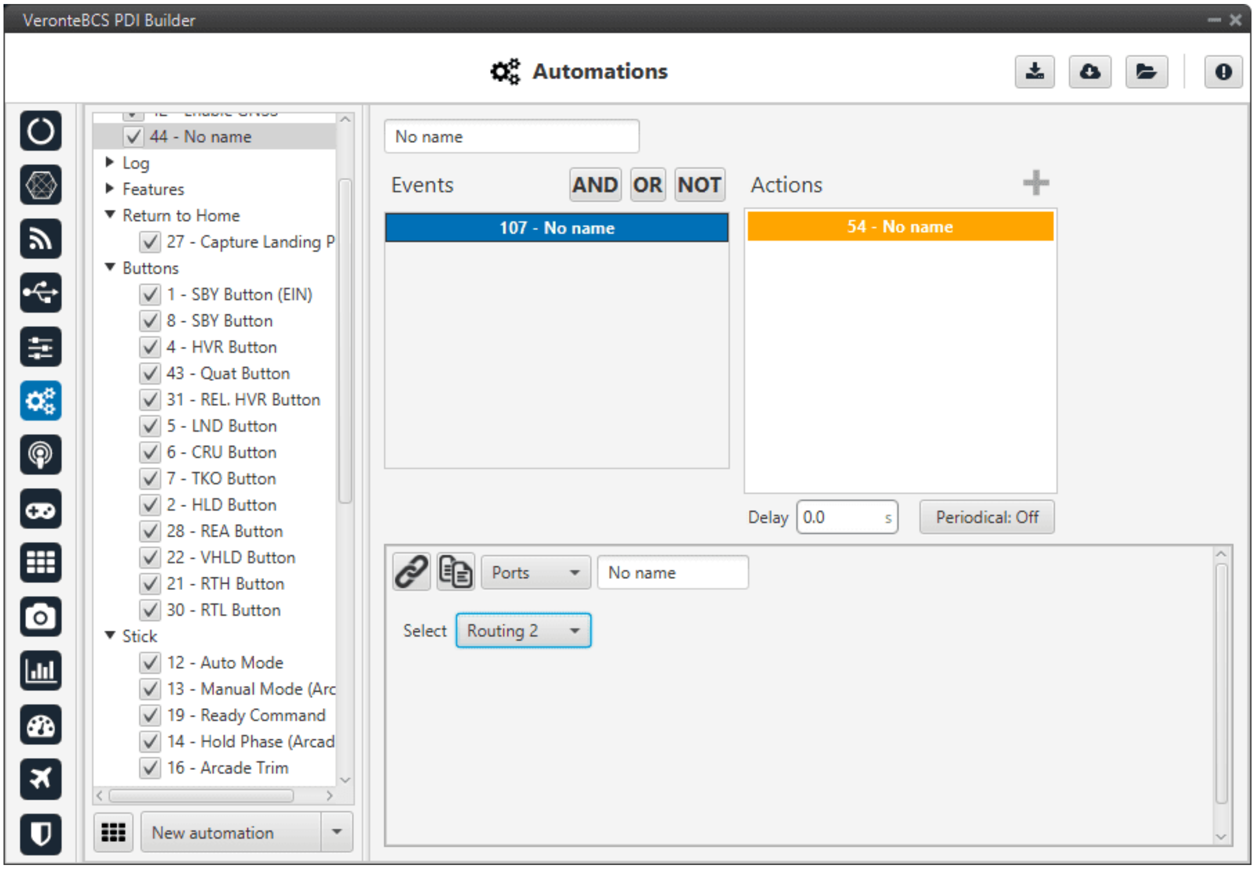
Change to Takeoff

Fig. 101: **Phase action**

These phases have been created previously. See section *Phases*, for more information about creating phases.

2.6.2.2.17 Ports

This action allows the user to switch between 4 pre-set configurations defined in the *Ports* section of the Communication menu.



No name

Delay 0.0 s

Periodical: Off

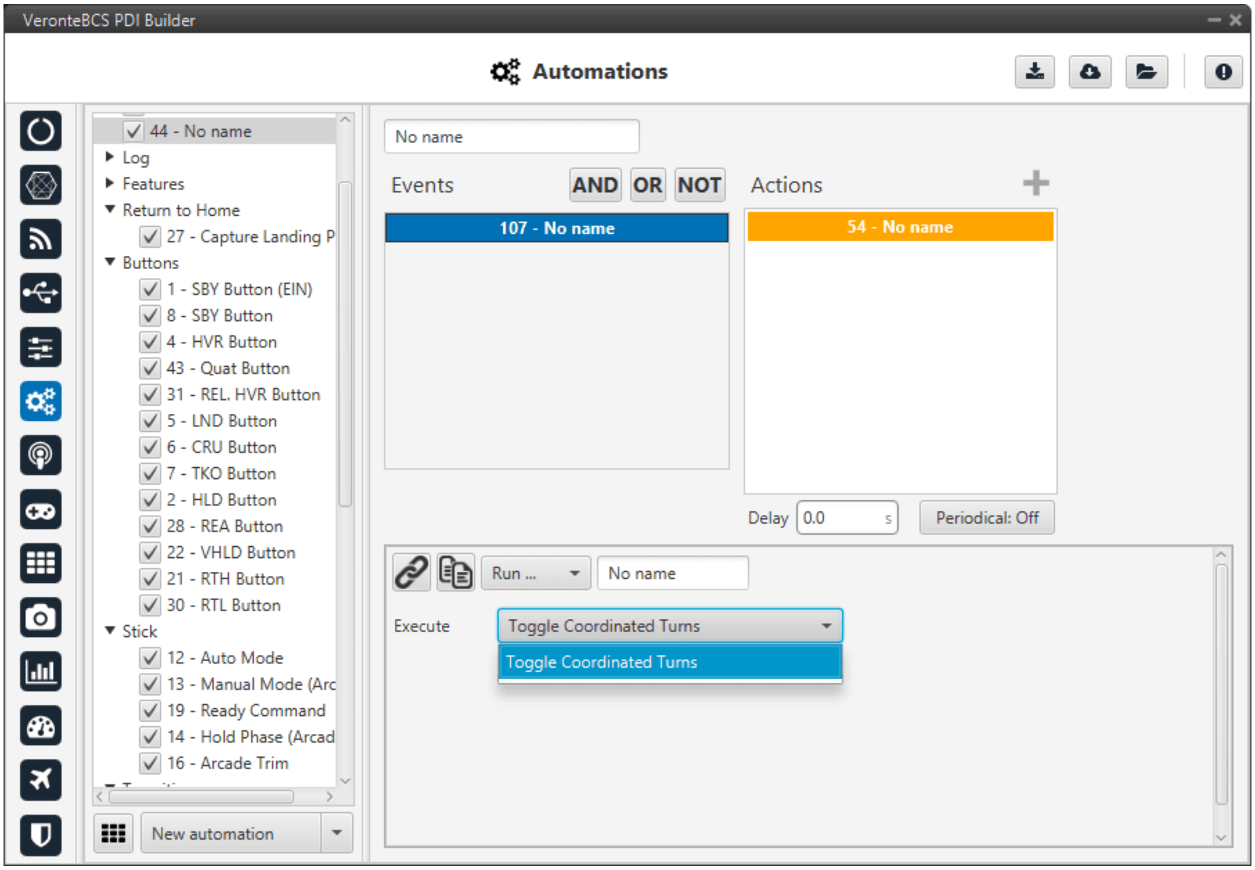
Ports | No name

Select Routing 2

Fig. 102: **Ports action**

2.6.2.2.18 Run block program

When this action is triggered, the block program specified in the “Execute” label is executed.



No name

107 - No name

54 - No name

Delay 0.0 s

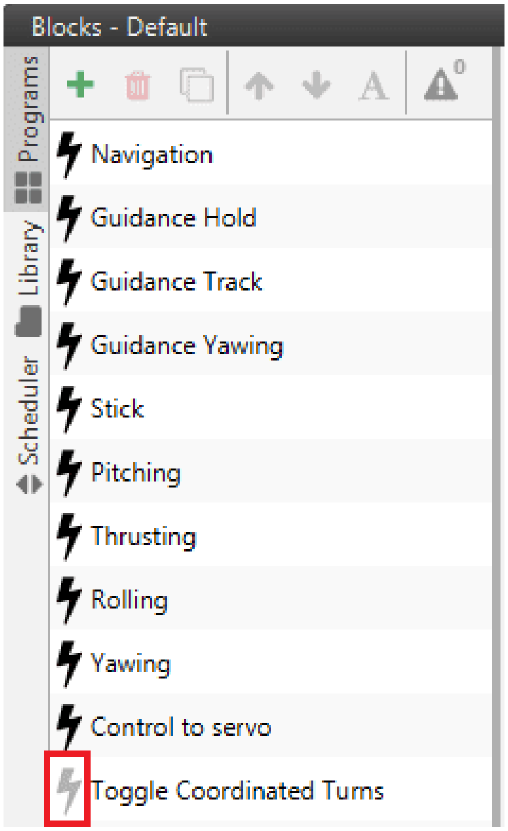
Periodical: Off

Run ... No name

Execute Toggle Coordinated Turns

Fig. 103: **Run block program action**

This action can run those programs that have the lightning icon in grey color as those programs with the lightning icon in black color run continously. For more information about programs, see section *Block Programs*.



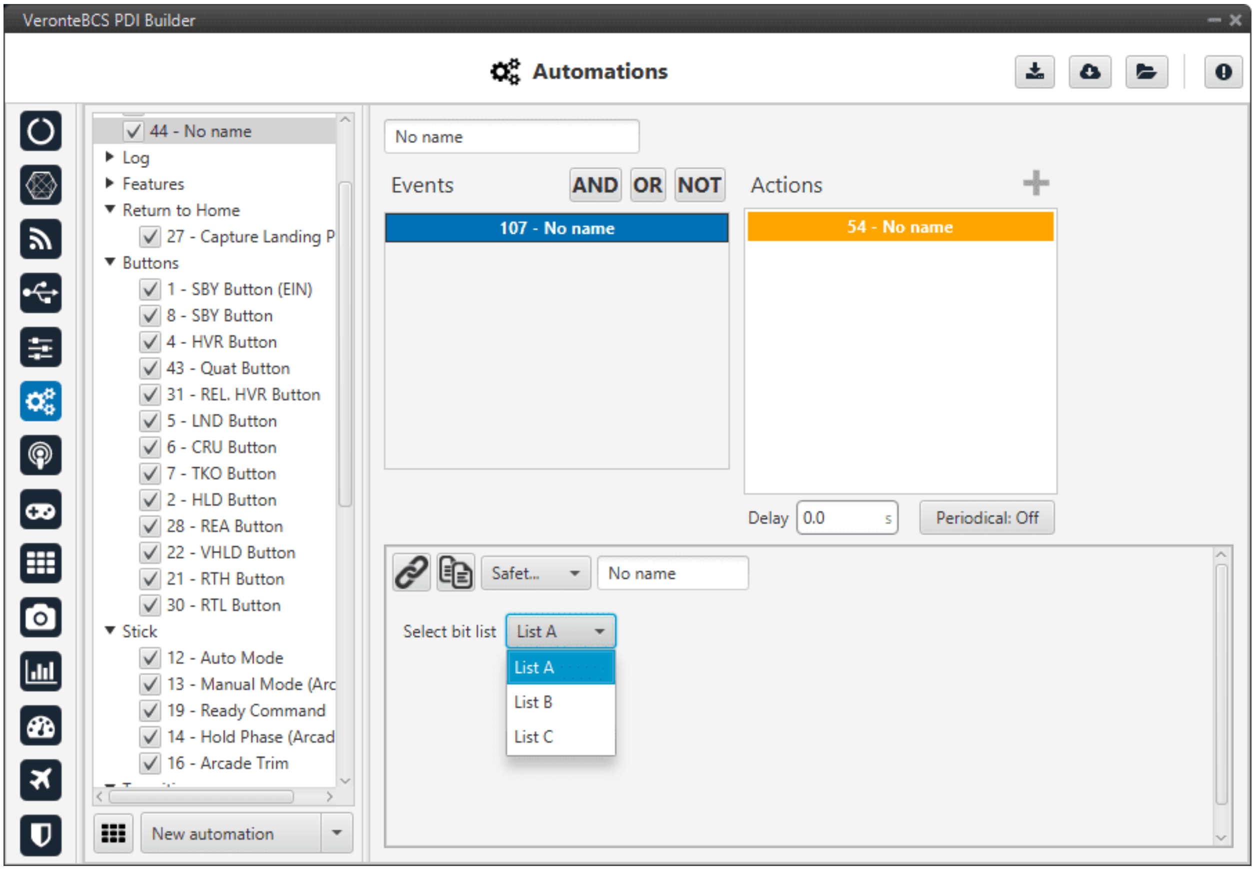
Blocks - Default

- Navigation
- Guidance Hold
- Guidance Track
- Guidance Yawing
- Stick
- Pitching
- Thrusting
- Rolling
- Yawing
- Control to servo
- Toggle Coordinated Turns

Fig. 104: **Grey lightning icon**

2.6.2.2.19 Safety Bits

This action selects a predefined safety bits list.



No name

Delay 0.0 s

Periodical: Off

No name

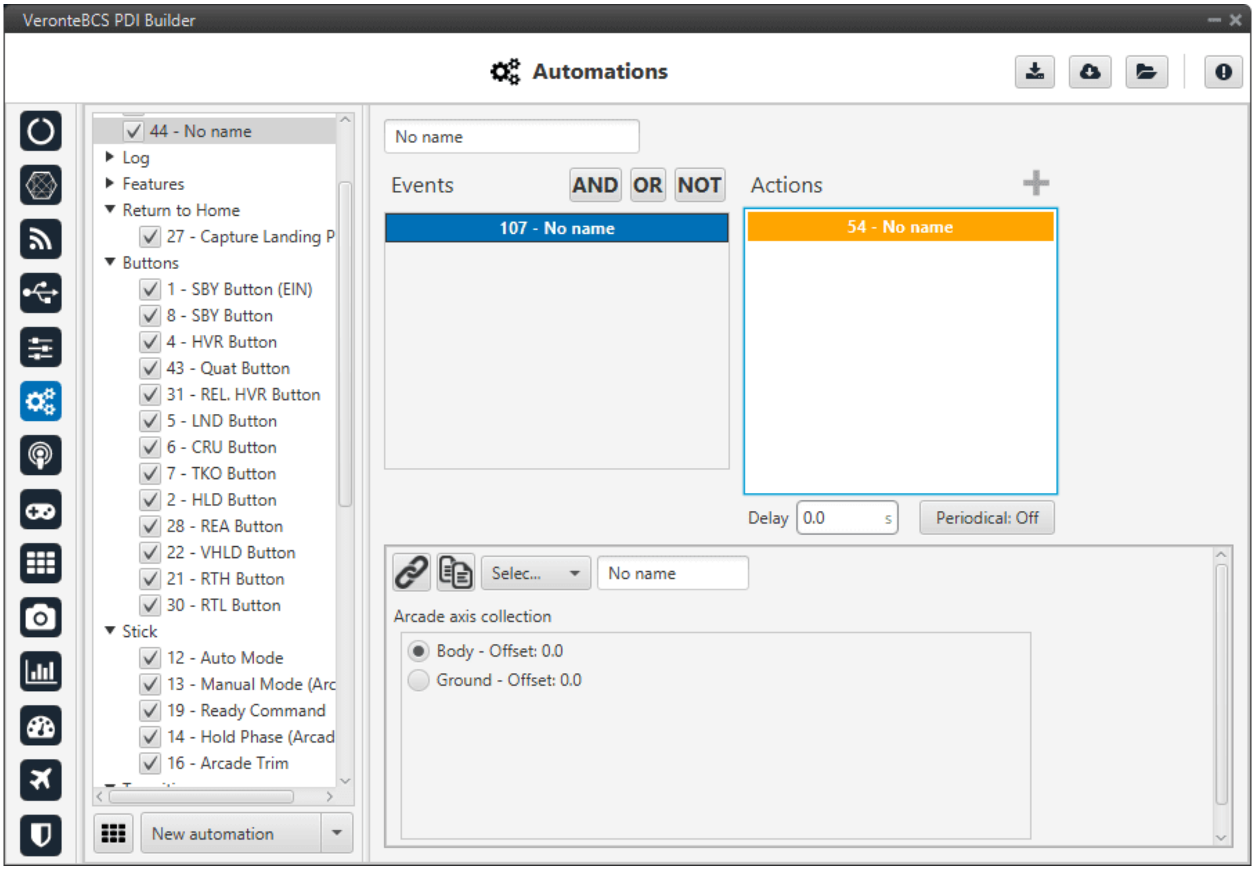
Select bit list List A

Fig. 105: **Safety Bits action**

This list must be configured previously. For more information about this, see section *Safety bits*.

2.6.2.2.20 Select Arcade axis

The axes system of the aircraft is changed to one that has been previously created, see section *Arcade Axis* of the Control menu.



Delay 0.0 s

Periodical: Off

Arcade axis collection

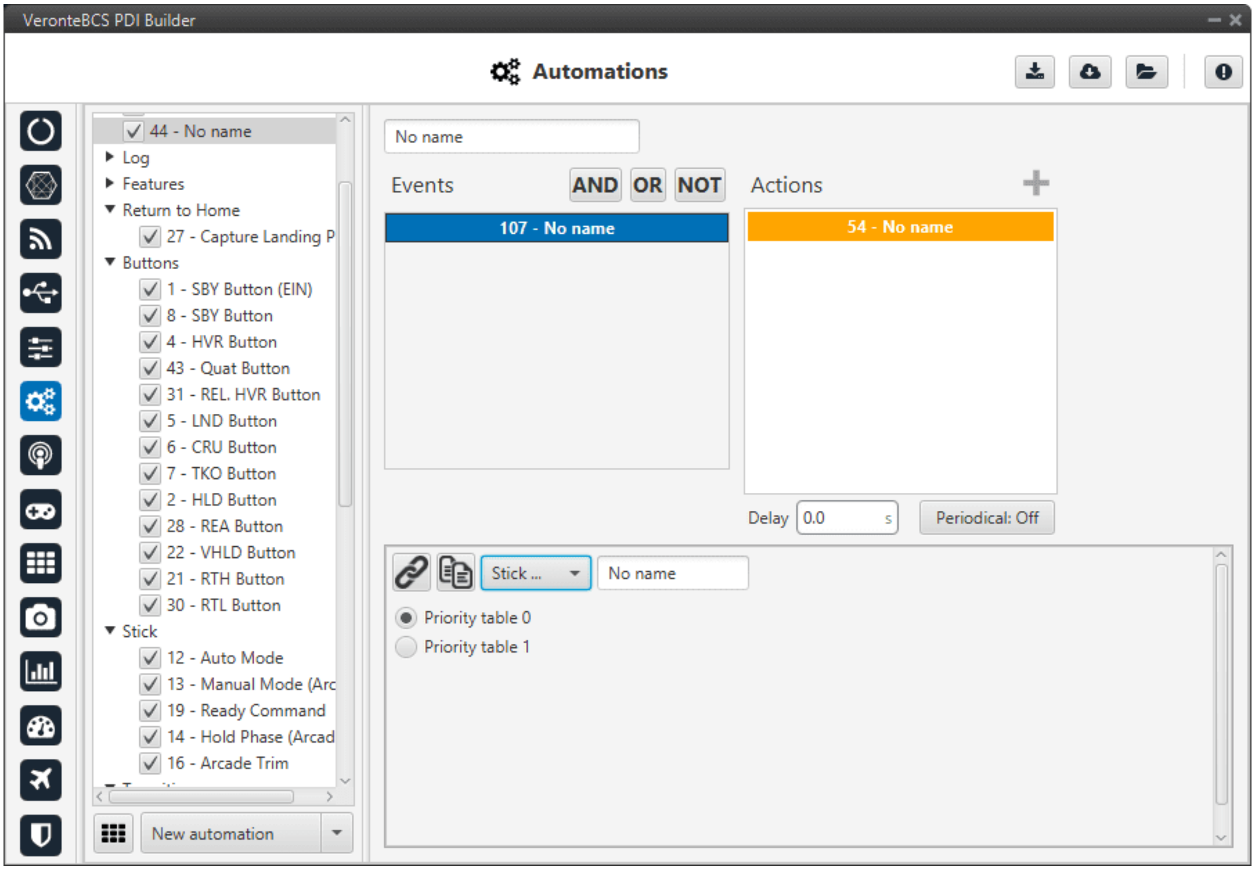
● Body - Offset: 0.0

○ Ground - Offset: 0.0

Fig. 106: **Select Arcade axis action**

2.6.2.2.21 Stick priority

The user can switch between the two priority tables of the **Stick block** (for more information about the stick block, see *Block Programs* section). By default priority table 0 is selected when **BCS** starts.



- ☑ 44 - No name
- Log
- Features
- Return to Home
  - ☑ 27 - Capture Landing P
- Buttons
  - ☑ 1 - SBY Button (EIN)
  - ☑ 8 - SBY Button
  - ☑ 4 - HVR Button
  - ☑ 43 - Quat Button
  - ☑ 31 - REL. HVR Button
  - ☑ 5 - LND Button
  - ☑ 6 - CRU Button
  - ☑ 7 - TKO Button
  - ☑ 2 - HLD Button
  - ☑ 28 - REA Button
  - ☑ 22 - VHLD Button
  - ☑ 21 - RTH Button
  - ☑ 30 - RTL Button
- Stick
  - ☑ 12 - Auto Mode
  - ☑ 13 - Manual Mode (Arc
  - ☑ 19 - Ready Command
  - ☑ 14 - Hold Phase (Arcad
  - ☑ 16 - Arcade Trim

No name

Events: 107 - No name

Actions: 54 - No name

Delay 0.0 s

Periodical: Off

Stick ... No name

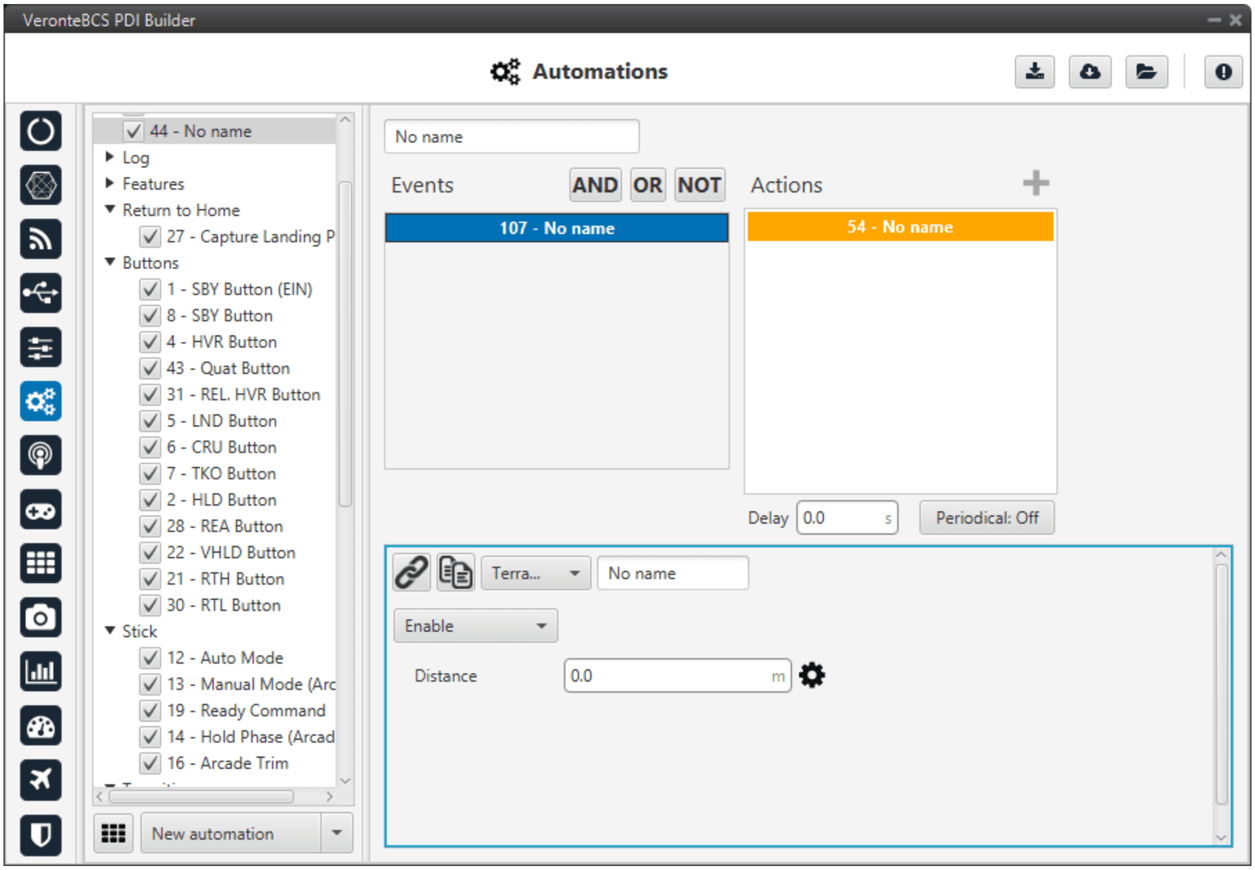
● Priority table 0

○ Priority table 1

Fig. 107: **Stick priority action**

2.6.2.2.22 Terrain obstacle

This option is used to make the aircraft climb when is reaching an altitude of zero meters, for example, when flying towards a mountain. This option can't be activated all the time because it will not allow the aircraft to land.



No name

Delay 0.0 s

Periodical: Off

Distance 0.0 m

Fig. 108: **Terrain obstacle action**

- **Distance**: Establish how the aircraft will climb, it can be said to be a repulsion value. High values made the platform ascent quickly. This effect is more noticeable when the aircraft is close to the ground.

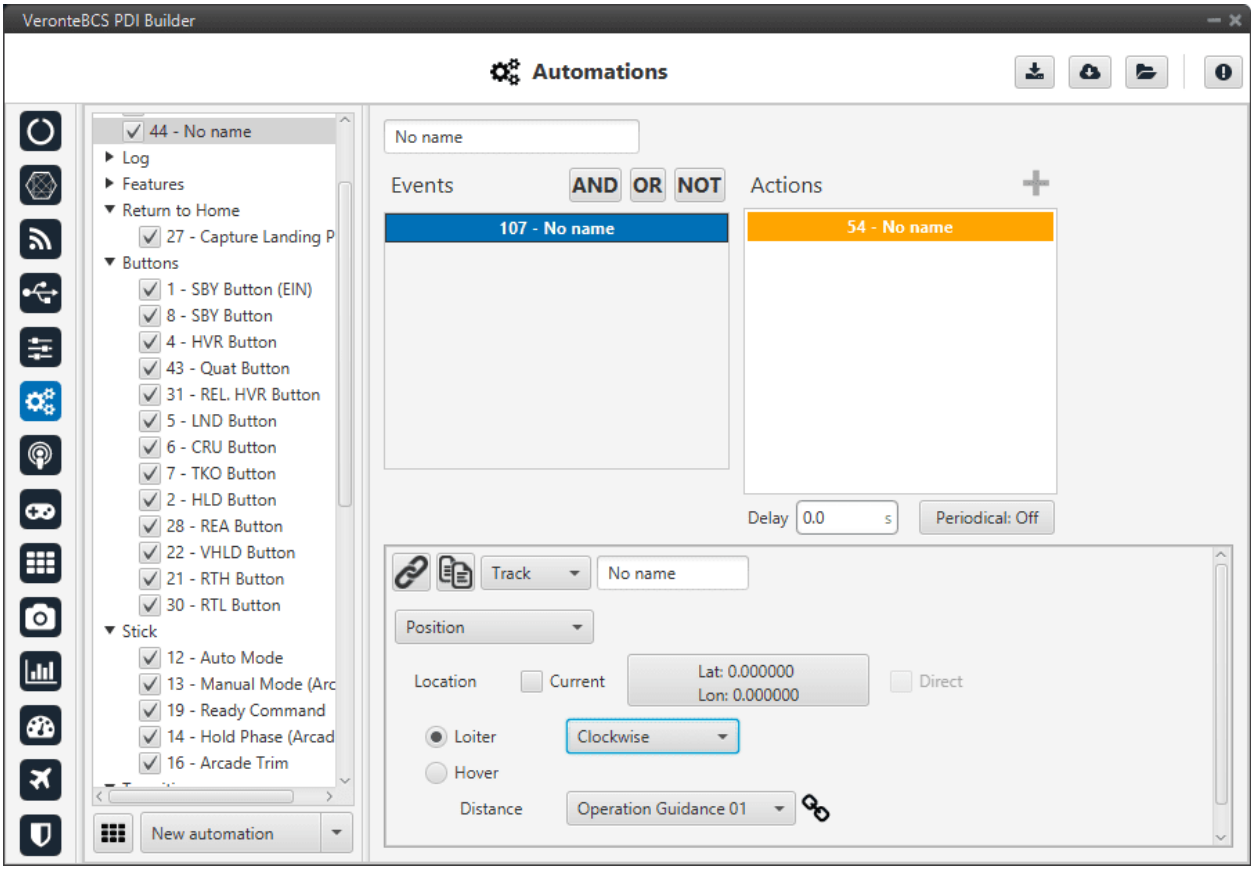
2.6.2.2.23 Track

This action is used to configure a hover/loiter route (depending if it is a multicopter or an airplane) for the platform. Besides, there exists an option to follow a moving object.

There are 3 different options for the Track action, selecting **Disabled** no action will have effect on the guidance. The others are explained below.

Position

The aircraft will loiter/hover in a selected point.



No name

Delay 0.0 s

Periodical: Off

Track — No name

Position

Location ☐ Current Lat: 0.000000 Lon: 0.000000 ☐ Direct

● Loiter Clockwise

○ Hover

Distance Operation Guidance 01

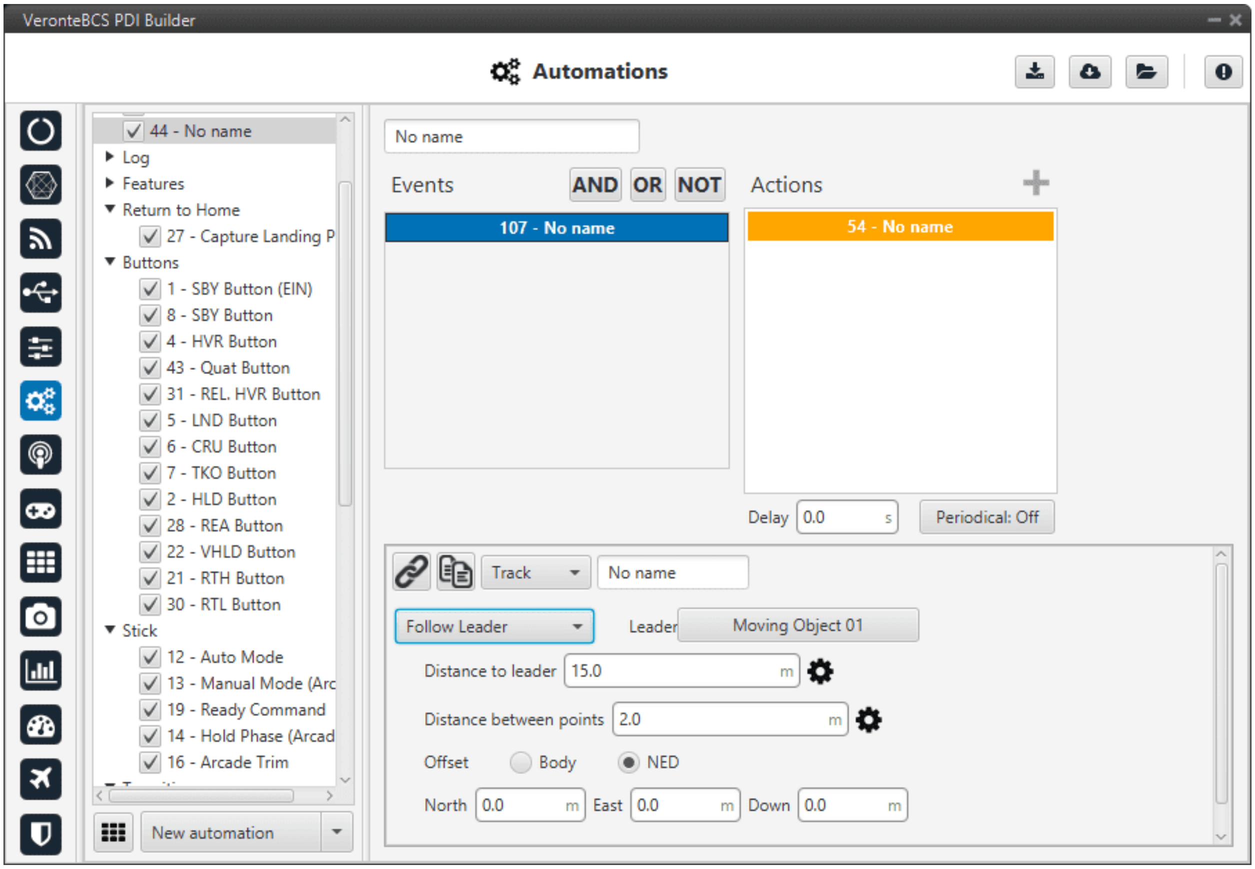
Fig. 109: **Track action - Position**

The following options are available:

- **Location**:
  - Selecting **Current** will make the platform to **hover** over the position that the vehicle has when this action is triggered, or **loiter** around that point in a circular route.
  - The box **Longitude**, **Latitude** allows the user to select the point where the hover/loiter will be performed.
- **Loiter**: It is also possible to select the direction of the loiter (**Auto**, **Clockwise** and **Anticlockwise**).
- When **Hover** is selected the option **Direct** can be enabled.
  - If direct is **enabled**, the **BCS** will calculate the control actions to reach the desired point based on the position error with that point.
  - If direct is **disabled**, the **BCS** will trace a path to the desired point and calculate the necessary control actions for this 'new route'.
- **Distance**:
  - **Distance + Loiter**: In this case, distance indicate the radius of the loiter circular route.
  - **Distance + Hover**: This option allows the user to define an acceptance radius around the position of the hover centre. If the UAV position is inside this circle, then **BCS** considers it is hovering correctly and will keep the position. If the centre of the hover changes its position and the UAV position is out of the hovering area, it will fly to the hovering centre, and once it is inside the circle, the hover will start.

Follow Leader

The platform will follow a moving object.



No name

Delay 0.0 s

Periodical: Off

Track | No name

Follow Leader

Leader: Moving Object 01

Distance to leader: 15.0 m

Distance between points: 2.0 m

Offset: ○ Body ● NED

North 0.0 m East 0.0 m Down 0.0 m

Fig. 110: **Track action - Follow Leader**

In this action the following parameters can be configured:

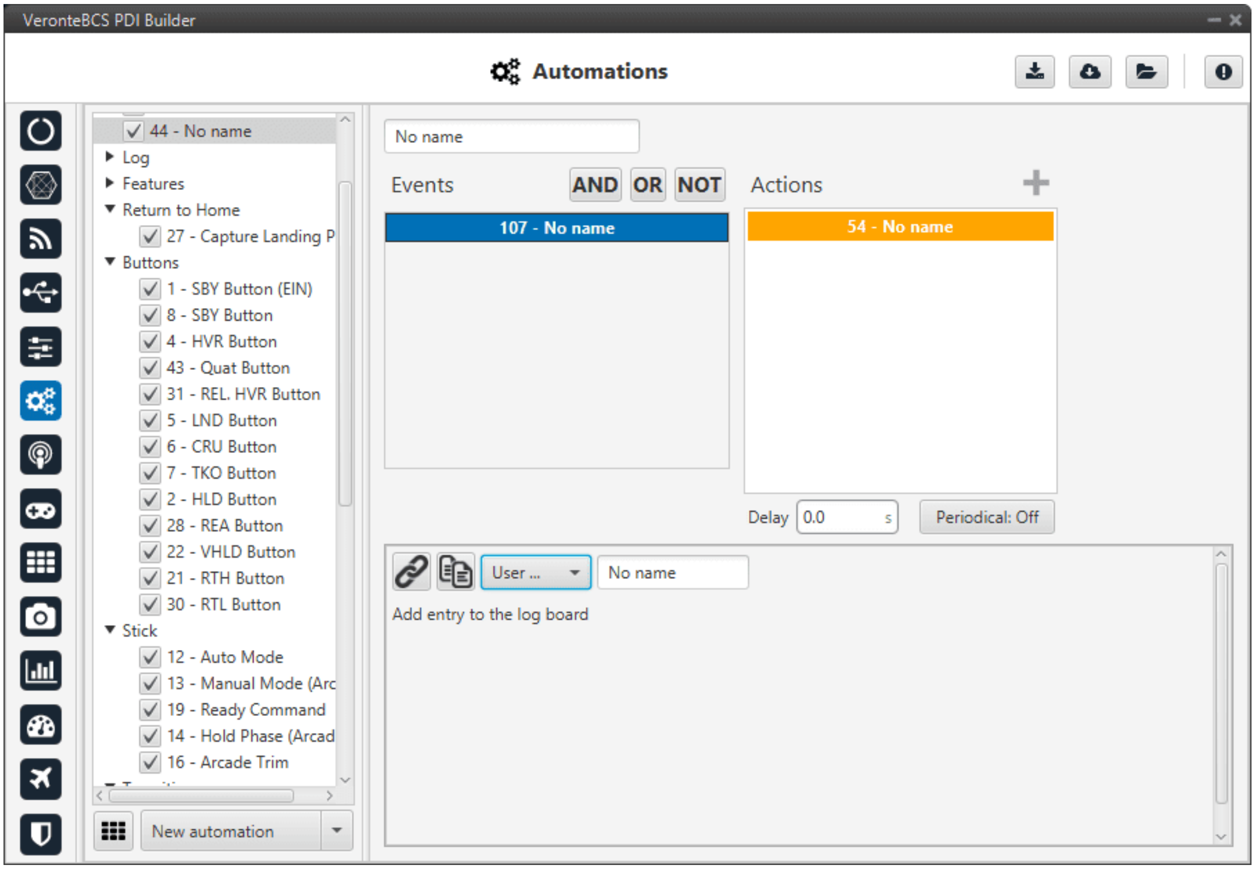
- **Leader**: Here is selected the object to follow, e.g. Moving Object.
- **Distance to leader**: Distance to leader over trajectory.
- **Distance between points**: Leader route is generated by points separated by the distance specified here.
- **Offset**: User can establish offset parameters related to trajectory in Body or NED coordinates.

Note: To configure correctly this automation, user has to follow the next steps:

- Configure Telemetry in Air and Ground units.
- Configure the automation as desired.

2.6.2.2.24 User Log

An entry, previously configured in *User Log -> Telemetry section*, is added to the on-board log.



No name

Delay 0.0 s

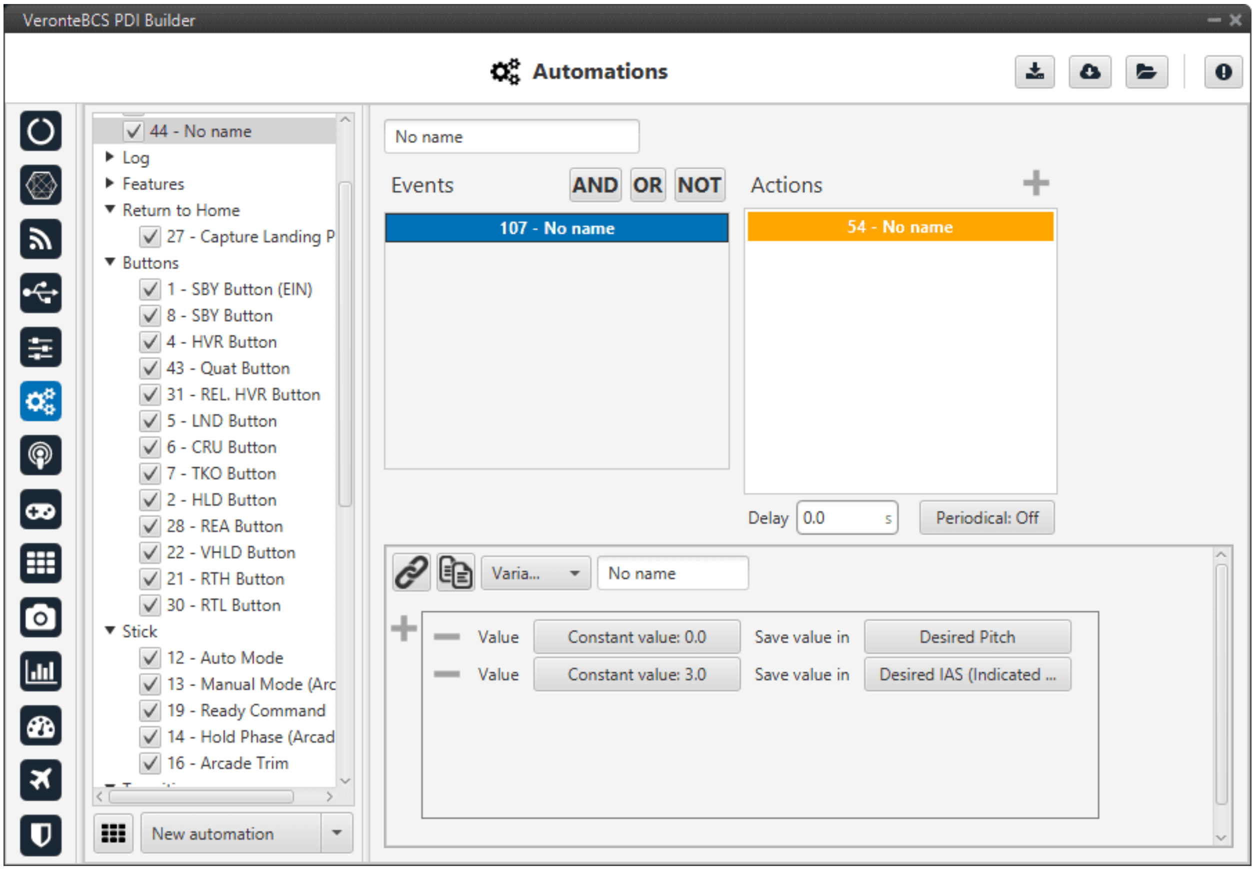
User ... No name

Add entry to the log board

Fig. 111: **User Log action**

2.6.2.2.25 Variable

This action allows the user to select variables and save them in user variables.



No name

Events: 107 - No name

Actions: 54 - No name

Delay 0.0 s

Periodical: Off

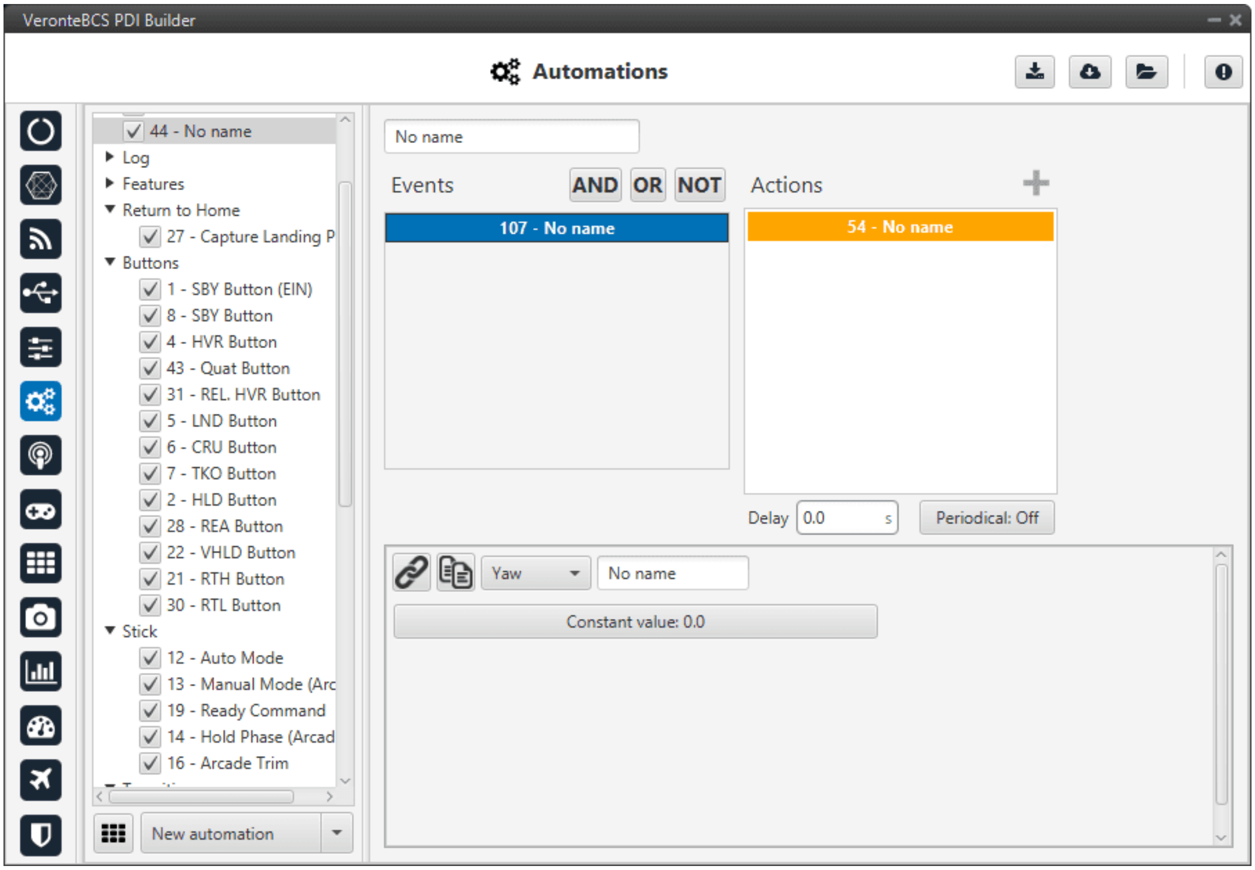
No name

Value	Constant value: 0.0	Save value in	Desired Pitch
Value	Constant value: 3.0	Save value in	Desired IAS (Indicated ...

Fig. 112: **Variable action**

2.6.2.2.26 Yaw

When this action is triggered the actual yaw can be commanded.



No name

Delay 0.0 s

Periodical: Off

Yaw

No name

Constant value: 0.0

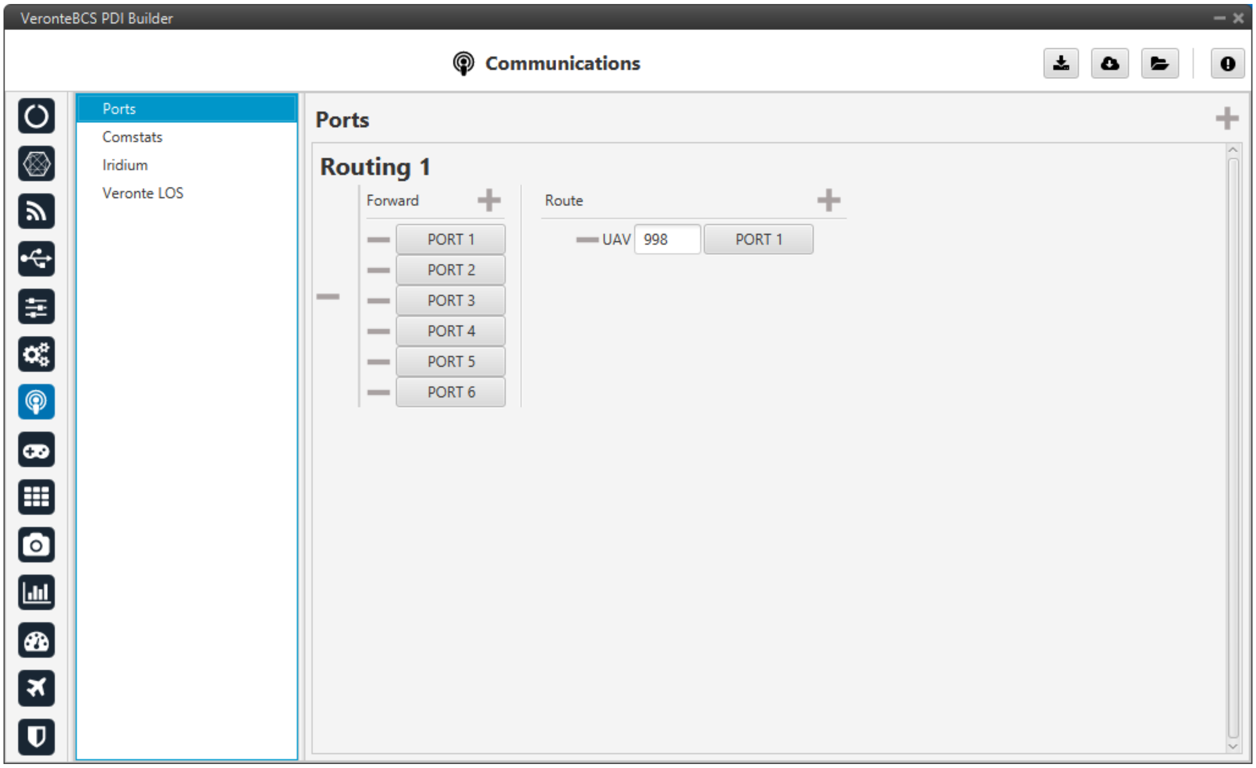
Fig. 113: **Yaw action**

This action is useful when flying without a magnetometer, as the user can establish the current yaw value when it is known.

2.7 Communications

2.7.1 Ports

Ports configuration allows the user to configure which communication ports (Commgr Ports in *I/O setup*) will be used for communication. When using the **Route** feature, **BCS** can be configured to **route** VCP messages for an external Veronte device with a known address (ID) through a given port.



Forward: PORT 1, PORT 2, PORT 3, PORT 4, PORT 5, PORT 6

Route: UAV 998 PORT 1

Fig. 114: **Ports menu**

Each of the different ports can be configured as either of the following options:

- **Forward**: Any messages generated by this unit (i.e. Telemetry or response messages to certain commands) will be sent through these ports.
- **Route**: Any messages received at any Commgr Port with the defined address will be re-sent through the defined port. It is possible to route several addresses through the same port, but is **not possible** to route the same address through several ports. Only the first configured port will be used. Routing also applies to messages generated by the unit for the defined address.

Note: The same **port** cannot be used as **Forward** and **Route** at the same time.

It is possible to define up to 4 routing setups, which can be switched unsing the *Ports action* of the Automations menu. Routing 1 will always be selected by default when booting **BCS**.

Application example

A practical example of the use of this menu are **BCS** Ground unit configurations. These configurations should have configured a Routing of **Address 2** (**Veronte Ops**) through the Commgr port to which the **USB consumer** is connected (to allow connection to the PC). In this example, **Port 1 producer** is the one connected to USB consumer.

This way, any messages that are received through a Commgr Port (i.e. through Veronte LOS), will be re-routed through Port 1 (USB) and received by **Veronte Ops** Software, including any messages generated by **BCS** ground unit itself.

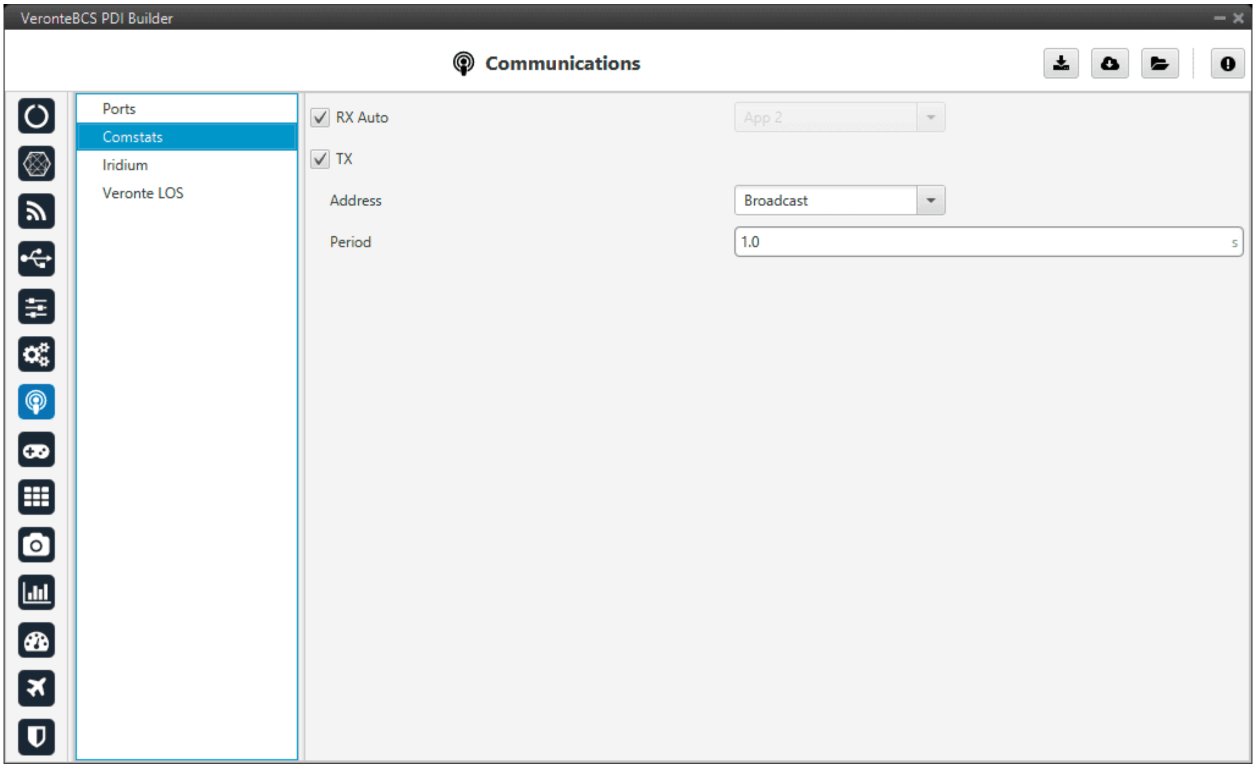
> **Warning:** An incorrect Port configuration can **disable USB communication**. If this happens, **BCS** will not be able to be detected through **Veronte Link** software. If this is the case, please visit *Maintenance mode - Troubleshooting section*.

2.7.2 Comstats

The Comstats feature allows **BCS** to make an estimation of the overall **quality of the communication** channel.

BCS will send periodically (**If enabled**) a message with its current communication statistics (Packets sent and received per second). Then, any other **BCS** unit can receive this information and compare against its own statistics to estimate the average amount of packets lost in the communication.

The results of this estimation can be monitored in variables **RX Packet Error Rate** (ID 2000) and **TX Packet Error Rate** (ID 2001). These variables can be used to enable, for example, failsafe actions in case of degradation or loss of communications.



☑ RX Auto — App 2

☑ TX

Address: Broadcast

Period: 1.0 s

Fig. 115: **Comstats menu**

It is possible to configure the source or destination of the statistics, as well as the frequency at which the Comstats message is sent:

- **RX Auto:** Enabling this option will use the first remote AP found. If this option is disabled, the user must choose manually the address of the unit used for Comstats calculation.
- **TX:** When enabled, the unit will periodically send its Comstats message (set the period). Select the address to which the message should be sent:

- App 2: Veronte Ops address.
- Broadcast: All units on the network.
- Veronte v4.X XXXX: To a specific unit.

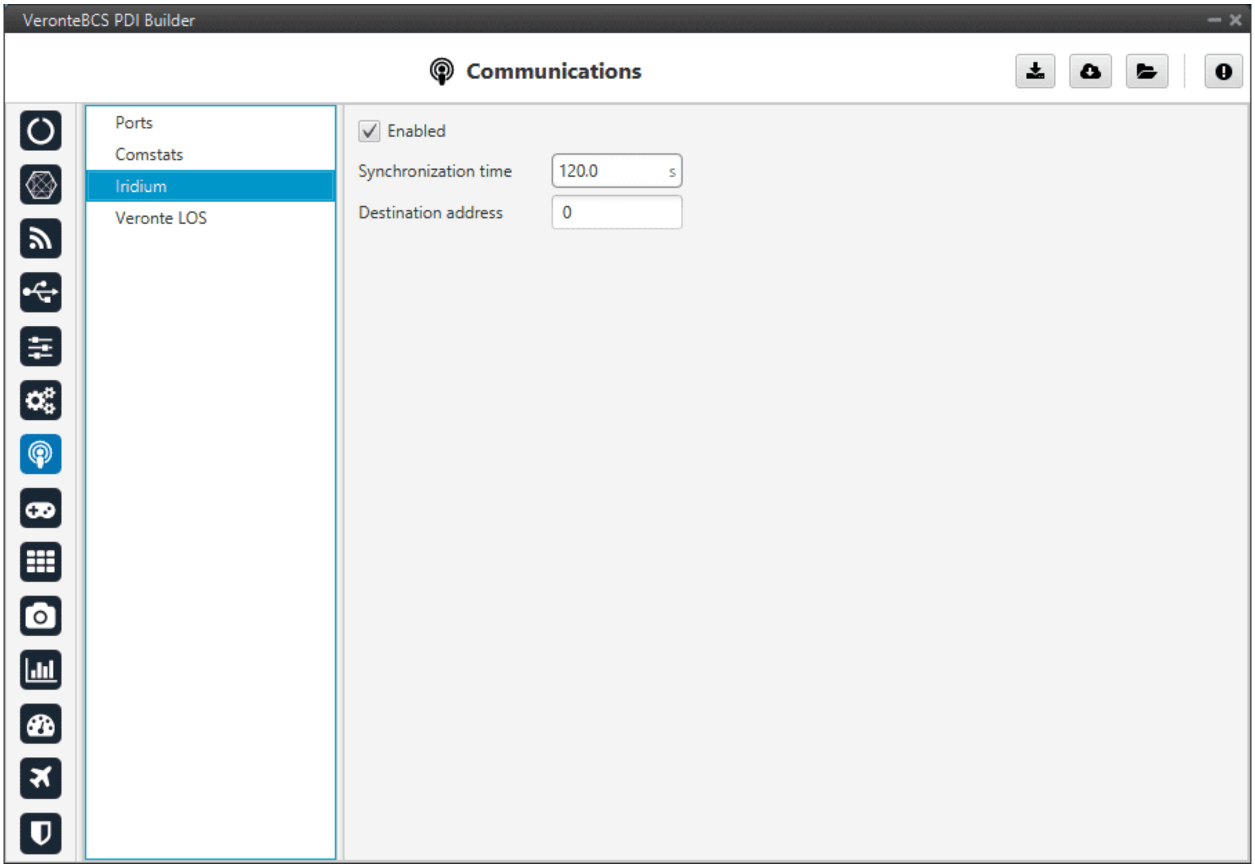
Note: Enabling **Tx** will enable **BCS** to send its Comstats message, but in order to compute Packet Error rate it's necessary to **receive** the **TX** message from a different unit.

Warning: **Packet error rate** is a good indicator of the status of the communication, but it is not representative of the radiolink status. For monitoring the status of the radiolink RSSI Variables (820-822) shall be used instead. Depending on the configuration it is posible to have bad Error rates with good RSSI (overloaded radiolink) or good Error rates with bad RSSI (degraded communication with low load on radiolink). For the best results, it is recommended to use a combination of both statistics for failsafe automations.

2.7.3 Iridium

Checking the **Enable** box will enable the use of Iridium communication through the **Iridium** Consumer/Producer in the *I/O setup*.

Warning: Before using the module, the user will have to register both of them (sender/receiver) in the RockBlock website. Please find more information about the registration process in this link: https://docs.rockblock.rock7.com/docs/rockblock-management-system.



☑ Enabled

Synchronization time: 120.0 s

Destination address: 0

Fig. 116: **Iridium menu**

In this menu the following parameters have to be set:

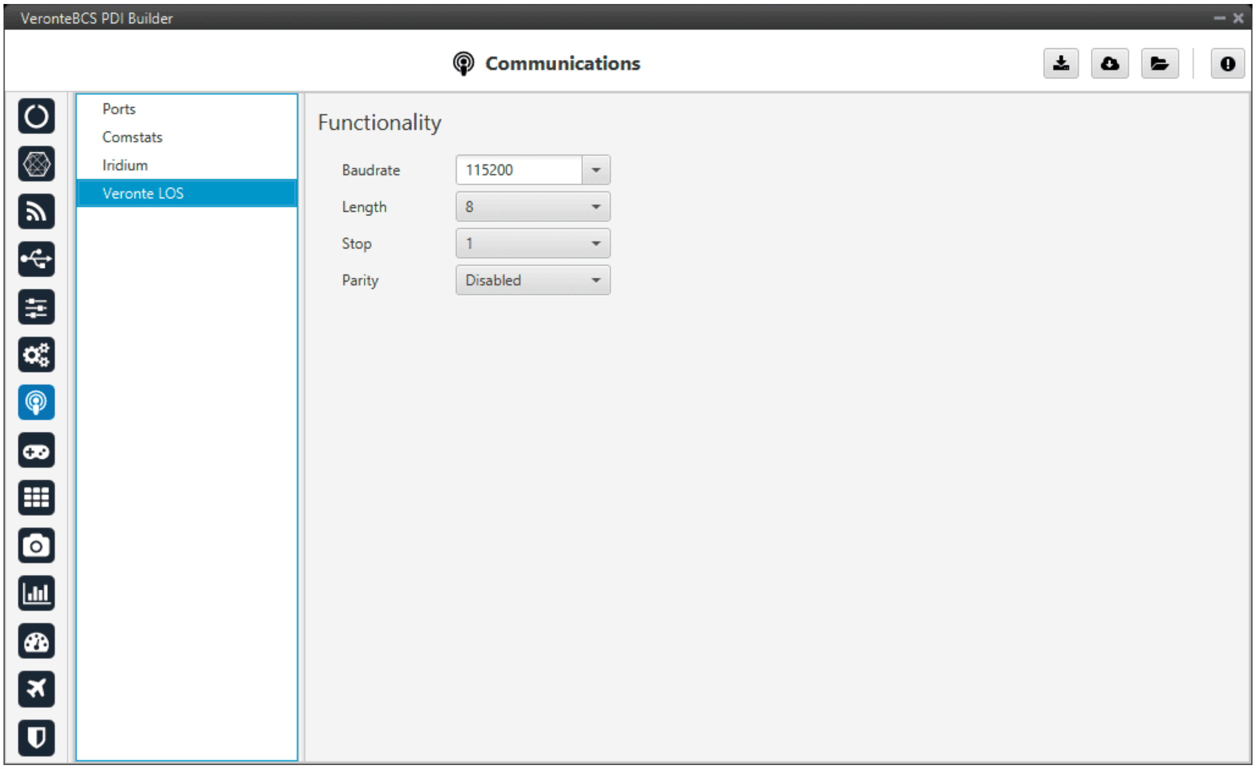
- **Synchronization time**: This is the transmission period, i.e., the time between 2 consecutive messages. This is a parameter that the user should configure taking into consideration its mission.
- **Destination address**: SN (Serial Number) of the destination Iridium module.

Note: To configure the syncronization time, it would be advisable to think about how the user want to use the Iridium communication. The user will pay for credits, and each credit means one message. Each individual message has to be paid, so the syncronization time can be configured in order not to run out of credits.

2.7.4 Veronte LOS

In this section, the serial port that communicates from the microcontroller to the internal radio is configured.

Warning: If the user changes the baudrate on the internal radio, it is also required to change it here and vice versa.



Functionality

Baudrate: 115200

Length: 8

Stop: 1

Parity: Disabled

Fig. 117: **Veronte LOS menu**

- **Baudrate**: This specifies how fast data is sent over a serial line.
- **Length**: This defines the number of data bits in each character: 4 to 8 bits.
- **Stop**: Number of stop bits sent at the end of every character: 1, 1.5, 2.
- **Parity**: Is a method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit. **Disabled**, **odd** or **even**.

Note: All these settings are already specified for a given device, therefore, **BCS** should match with them in order to be able to communicate.

2.8 Stick

In this section, the stick configuration on **BCS PDI Builder** is explained.

This allows the user to set **up to four transmitters** and **one virtual stick**. The **BCS** capabilities allow it to receive information from four different transmitters at the same time plus transforming some values into a virtual stick.

The content presented in the next menus covers:

- Setting of the transmitter's parameters.
- Definition of exponential response-curves for the desired channels.

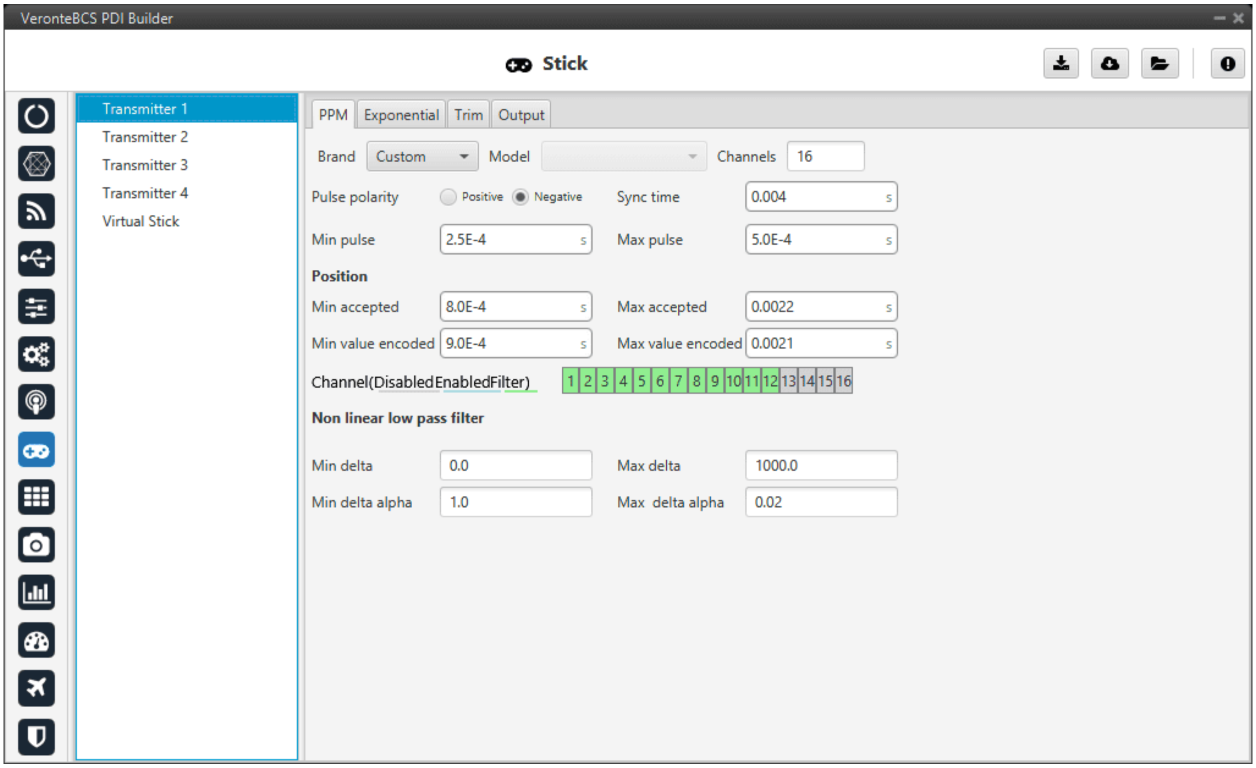
- Trimming of the channels' neutral position.
- Setting of the data receiving port on the **BCS**.
- Definition of a virtual stick.

2.8.1 Transmitter (1-4)

The wired connected transmitters are configured through the following tabs.

2.8.1.1 PPM

This tab provides the options to configure a Pulse Postion Modulation (PPM) radio controller to control the platform fitted with the **BCS**.



Brand: Custom
Model:
Channels: 16

Pulse polarity: ○ Positive ● Negative
Sync time: 0.004 s

Min pulse: 2.5E-4 s
Max pulse: 5.0E-4 s

Position

Min accepted: 8.0E-4 s
Max accepted: 0.0022 s

Min value encoded: 9.0E-4 s
Max value encoded: 0.0021 s

Channel(DisabledEnabledFilter): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Non linear low pass filter

Min delta: 0.0
Max delta: 1000.0

Min delta alpha: 1.0
Max delta alpha: 0.02

Fig. 118: **Stick - PPM menu**

- **Brand**, **Model** and **Channels**: **BCS PDI Builder** has been configured to provide the user with the expected parameters to configure different transmitters models.

Brand	Models	Channels
Futaba	8J/10J/12K/14SG	8
	12K/14SG	12
	T18SZ	8
Jeti	DC 16/DC 24	16
FrSky	Taranis X9D	8
	Horus X12S	8
TBS	Croosfire	8
Embention	Stick Expander	16
Custom	-	-

  - Custom: If the user's transmitter is not among those mentioned above, choose this option and replace the parameter values with the appropriate ones.

- **Pulse polarity**: Indicates the pulse polarity:
  - **Positive**: Default signal is low and goes up to high.
  - **Negative**: Default signal is high and goes down to low.
- **Sync time**: Minimum time on the PPM output till the next frame. It tells the receiver to reset its channel counter.
- **Minimum/Maximum pulse**: Pulse length, it depends on the system and it is a constant value (usually 0.2-0.5 ms).
- **Position**
  - **Minimum/Maximum accepted**: Pulse length accepted for each channel. Standard for R/C servos uses a pulse of 1 ms for the maximum position at one end, 1.5 ms for the midpoint and 2 ms for the maximum position at the opposite end.
  - **Minimum/Maximum encoded**: If there is noise and the signal is varying around the minimum/maximum values accepted, **BCS** will encode those values to the ones set here. For instance, a pulse length between 0.8-0.9 ms will be considered as one of 0.9 ms.
  - **Channels**: They set the number of accepted channels. Besides, it is possible to Disable/Enable/Filter each channel individually.
- **Non linear low pass filter**
  - **Minimum/Maximum delta**: Default parameters are recommended.
  - **Minimum/Maximum delta alpha**: Default parameters are recommended.

The figure below shows the PPM signal that arrives to **Veronte BCS**:

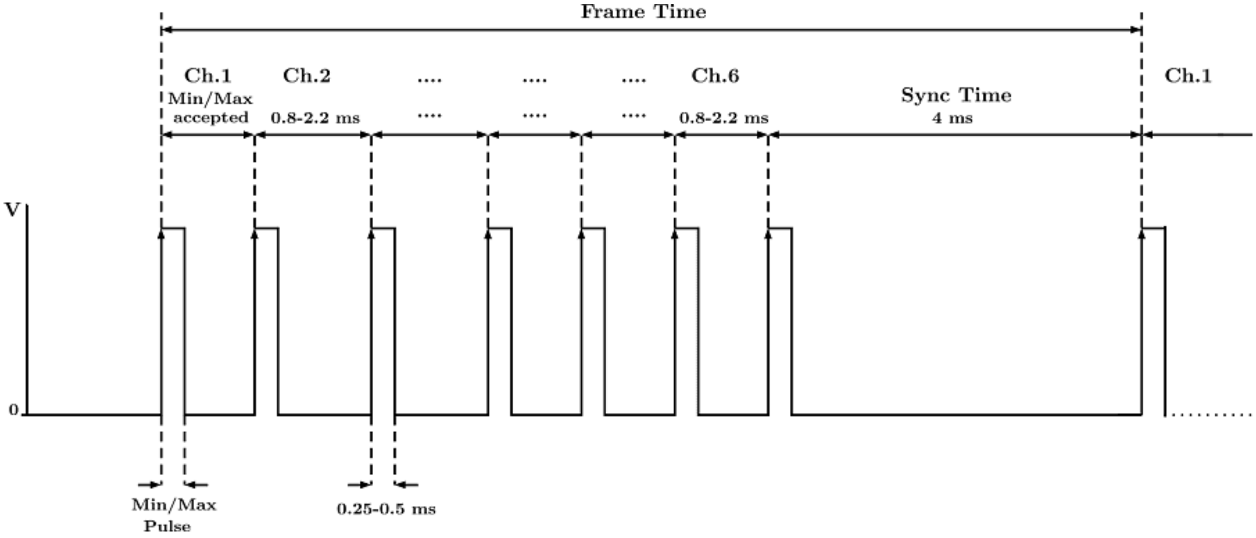
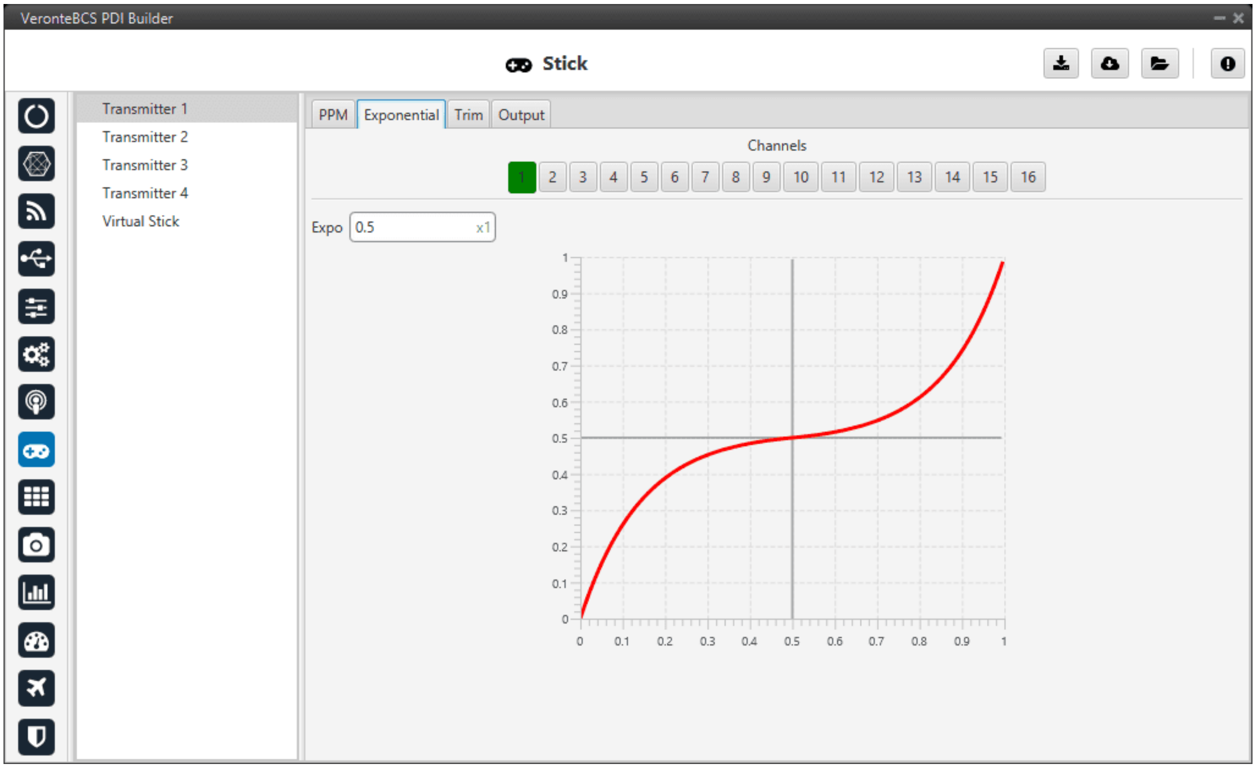


Fig. 119: **PPM signal**

2.8.1.2 Exponential

The second tab allows the user to define an exponential stick response for every channel.

The allowed inputs range from 0 to 1 and there is a graph showing the generated response curve, as can be seen in the figure below.



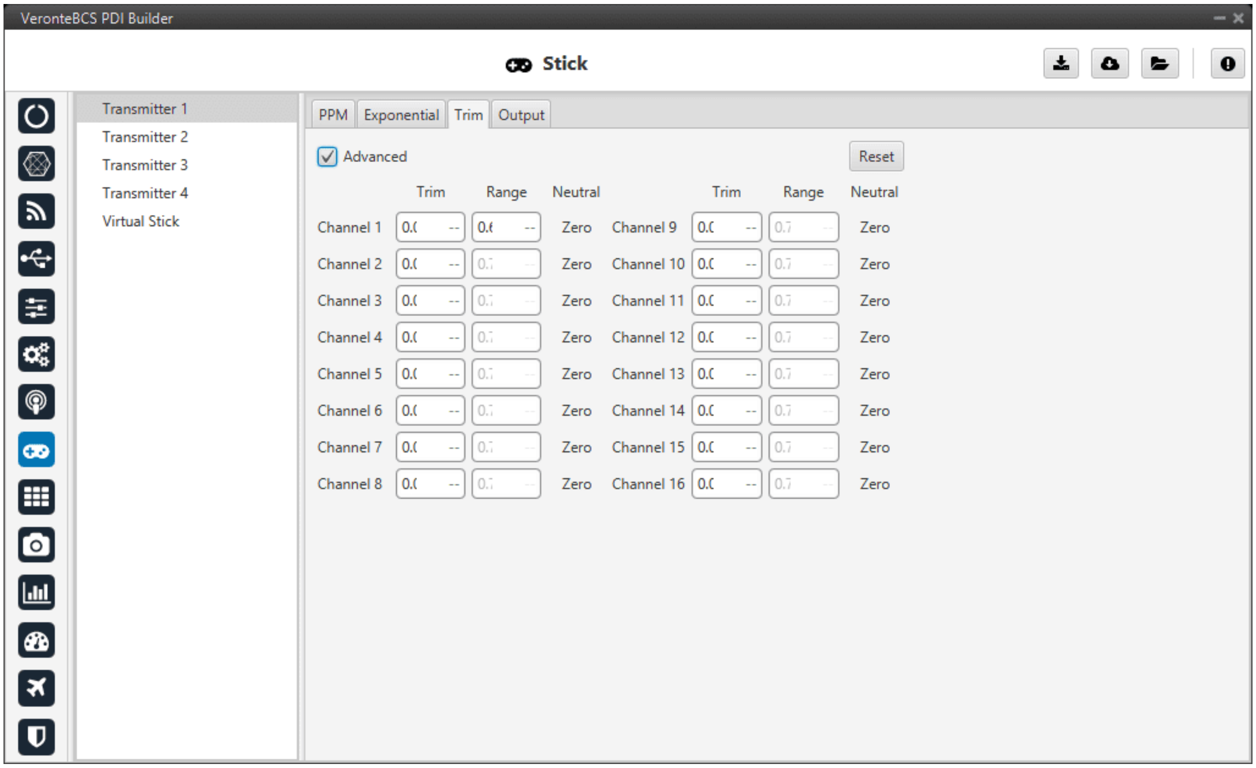
Expo 0.5

Fig. 120: **Stick - Exponential menu**

The **X axis** of the graph corresponds to the stick input and the **Y axis** is the result of applying the exponential function to that stick input.

2.8.1.3 Trim

The third tab available is the Trim option.



☑ Advanced

	Trim	Range	Neutral		Trim	Range	Neutral
Channel 1	0.(--	0.6 --	Zero	Channel 9	0.(--	0.7 --	Zero
Channel 2	0.(--	0.7 --	Zero	Channel 10	0.(--	0.7 --	Zero
Channel 3	0.(--	0.7 --	Zero	Channel 11	0.(--	0.7 --	Zero
Channel 4	0.(--	0.7 --	Zero	Channel 12	0.(--	0.7 --	Zero
Channel 5	0.(--	0.7 --	Zero	Channel 13	0.(--	0.7 --	Zero
Channel 6	0.(--	0.7 --	Zero	Channel 14	0.(--	0.7 --	Zero
Channel 7	0.(--	0.7 --	Zero	Channel 15	0.(--	0.7 --	Zero
Channel 8	0.(--	0.7 --	Zero	Channel 16	0.(--	0.7 --	Zero

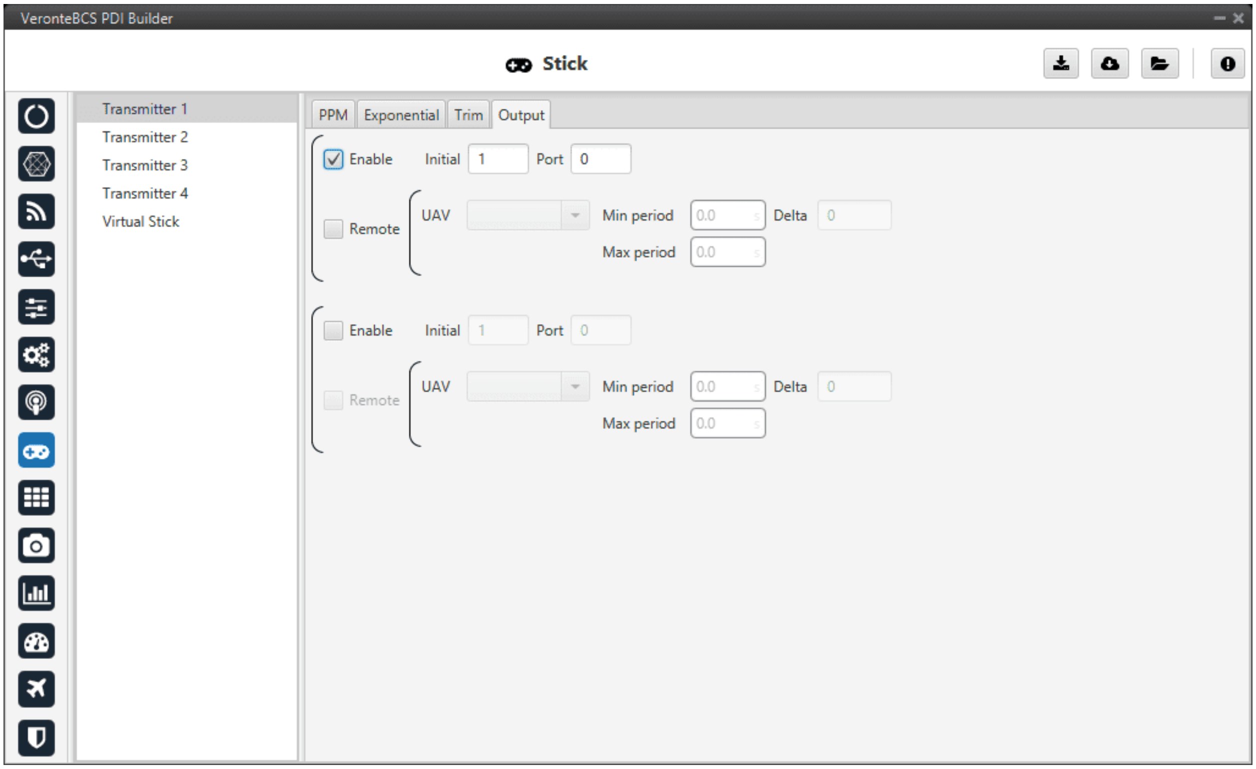
Fig. 121: **Stick - Trim menu**

By enabling the **Avanced** option, the user can set the expected trim values manually. The user should have a deep knowledge on its transmitter if this option is selected.

Finally, on the right hand side, the **Reset** button puts every parameter back to 0.

2.8.1.4 Output

In this menu the user sets the receiving port and process the incoming commands. Once the stick has been configured, the commands that arrive at the control station have to be sent to the air unit.



☑ Enable Initial 1 Port 0
□ Remote UAV Min period 0.0 s Delta 0
Max period 0.0 s

□ Enable Initial 1 Port 0
□ Remote UAV Min period 0.0 s Delta 0
Max period 0.0 s

Fig. 122: **Stick - Output menu**

In this menu, the following parameters can be configured:

- **Enable**.
- **Initial Channel at destination**: The user indicates to which channel of the air autopilot will be sent the first channel received in the ground unit. The channels arrive at the platform in order and without spaces between them.

  For example, if at the GND channels 6,7,8,9 and 10 are enabled, the AIR will receive channels 1,2,3,4 and 5. Therefore channel 6 of the stick will be channel 1 in the AIR configuration.
- **Port**: If more than one transmitter is configured, each transmitter must be configured on a different port. This has to match the port set on the air unit.
- **Remote**: It has to be enabled if the user wants to allow the delivery of the commands to the platform.
  - **UAV**: The address of the UAV that receive the commands has to be indicated. The following options are available:
    - App 2: Veronte Ops address.
    - Broadcast: The commands are sent to all units on the network. **We recommend this option**.
    - Veronte v4.X XXXX: The address of a specific air unit.

- **Min period**: As the period is the inverse of the frequency, this is the **maximum frequency**. Therefore, to give the pilot more control, this is the frequency that is set when the **stick is commanding**. We recommend **0.02s**.
- **Max period**: As the period is the inverse of the frequency, this is the **minimum frequency**. Thus, to free up bandwidth, this is the frequency that is set when the **stick is idle**. We recommend **0.2s**.
- **Delta**: This parameter determines whether the frequency is set to the minimum or maximum period set above.

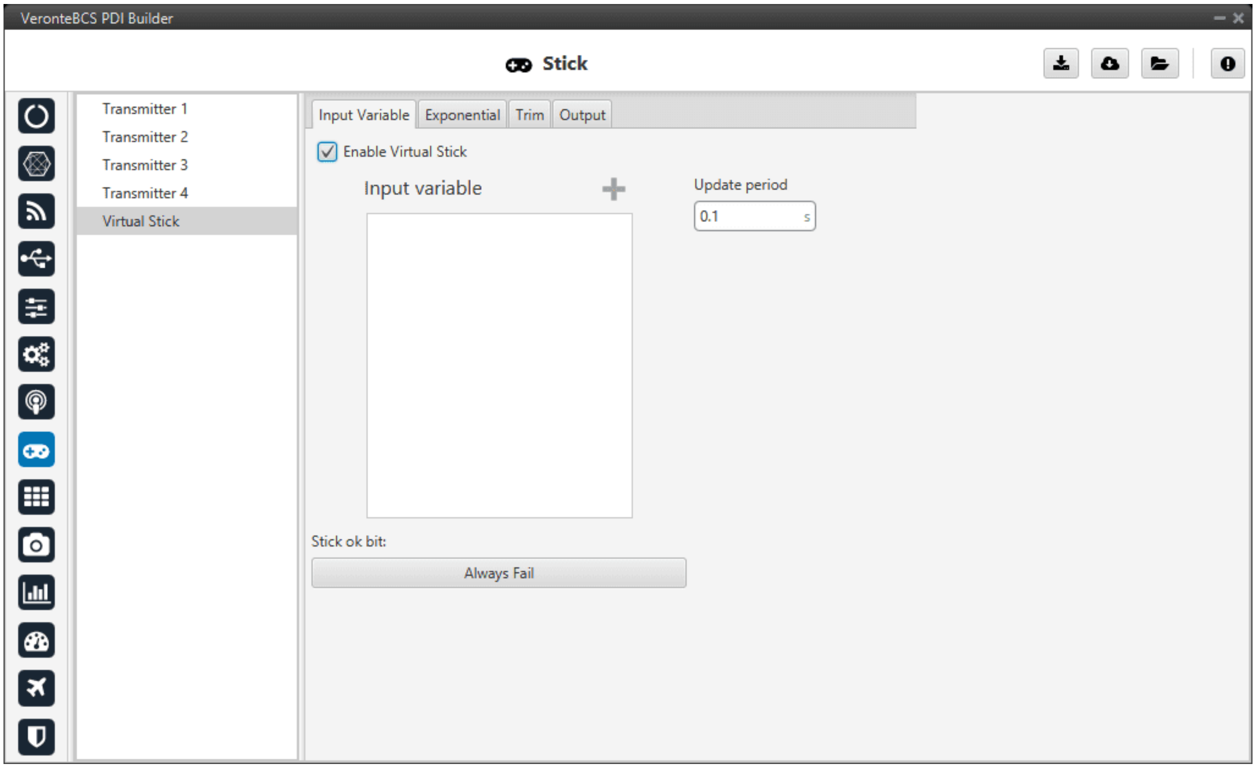
  If **BCS** detects a **change above** the **delta value**, the frequency goes to the **maximum frequency** (minimum period). While if the **changes are less than this value**, it switches to the **minimum frequency** (maximum period). We recommend **10**.

Note: An example of the *Stick integration* can be found in the Integration examples section.

2.8.2 Virtual Stick

In this menu a virtual stick can be defined. It can be useful when:

- Testing the control station configuration (not flight tests).
- Using a USB stick. Please find more information about this in the *USB -> Integration examples section*.
- The information of a stick is received through a different channel than PPM, so that the virtual stick will process that information and input it into the system.



☑ Enable Virtual Stick

Update period: 0.1 s

Stick ok bit: Always Fail

Fig. 123: **Virtual stick menu**

In this menu the user can configure:

- **Enable Virtual Stick**
- **Input variables**: Place here the variables containing the stick information. It is not necessary when using a USB stick or simply virtual stick.
- **Update Period**: Configure the period required. We recommend **0.02 s**.

In addition, a **virtual stick widget** must be configured in **Veronte Ops**. Please find more information in the Stick -> Widgets section of the **Veronte Ops** manual.

The tabs **Trim** and **Output** are the same as the Transmitter ones, so refer to the *Transmitter* menu for more information.

Note: An example of the *Virtual stick integration* can be found in the Integration examples section.

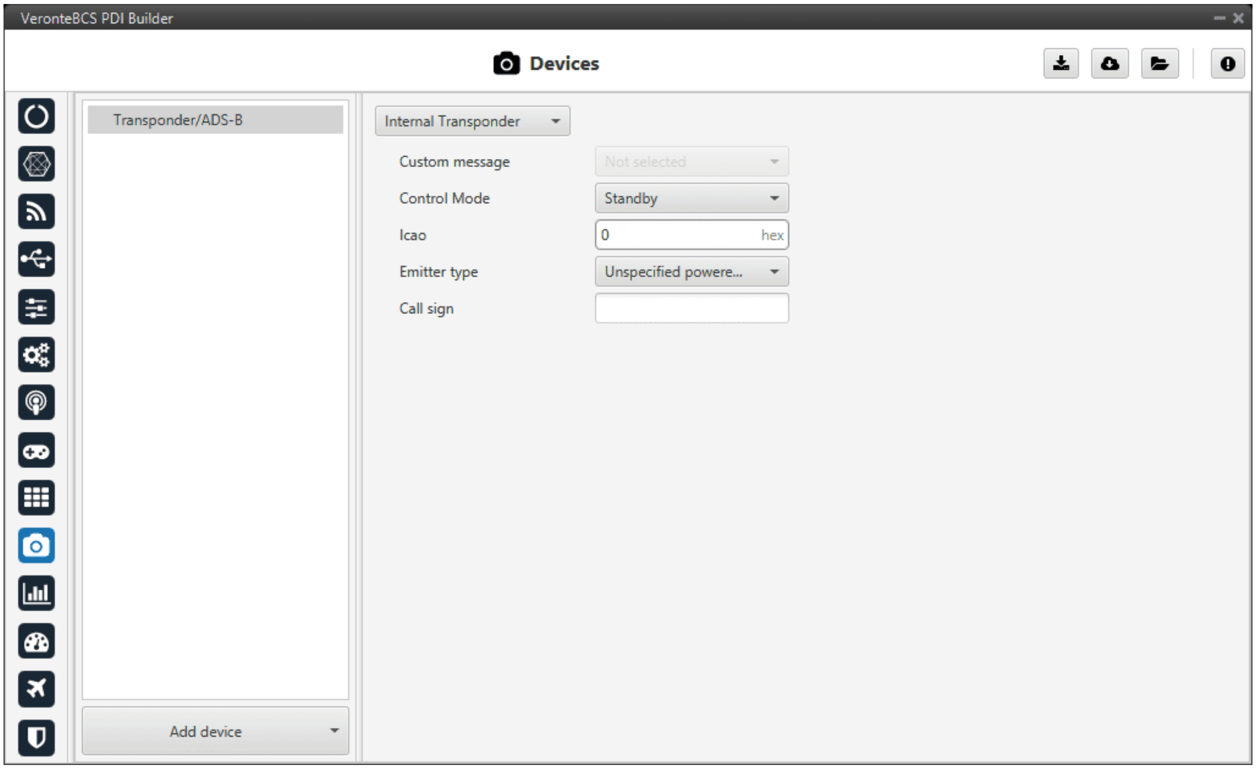
2.9 Block Programs

Block programs are used and configured in the same way as **Autopilot 1x** (which uses 1x PDI Builder). The difference between **1x** and **BCS** lies in the absence of **guidance** and **navigation** blocks for **BCS**.

Hence, block programs are explained in the Block Programs section of **1x PDI Builder manual**.

2.10 Devices

This menu displays the possible payloads/devices that can be configured with **BCS**. Each section will allow the user to configure different parameters from the available variety of payloads.



Internal Transponder

Custom message: Not selected

Control Mode: Standby

Icao: 0 hex

Emitter type: Unspecified powere...

Call sign:

Fig. 124: **Devices menu**

By default, only the *Transponder/ADS-B* device is added to this menu (as shown in the figure above). However, users can added other supported devices by simply by clicking **Add device**:

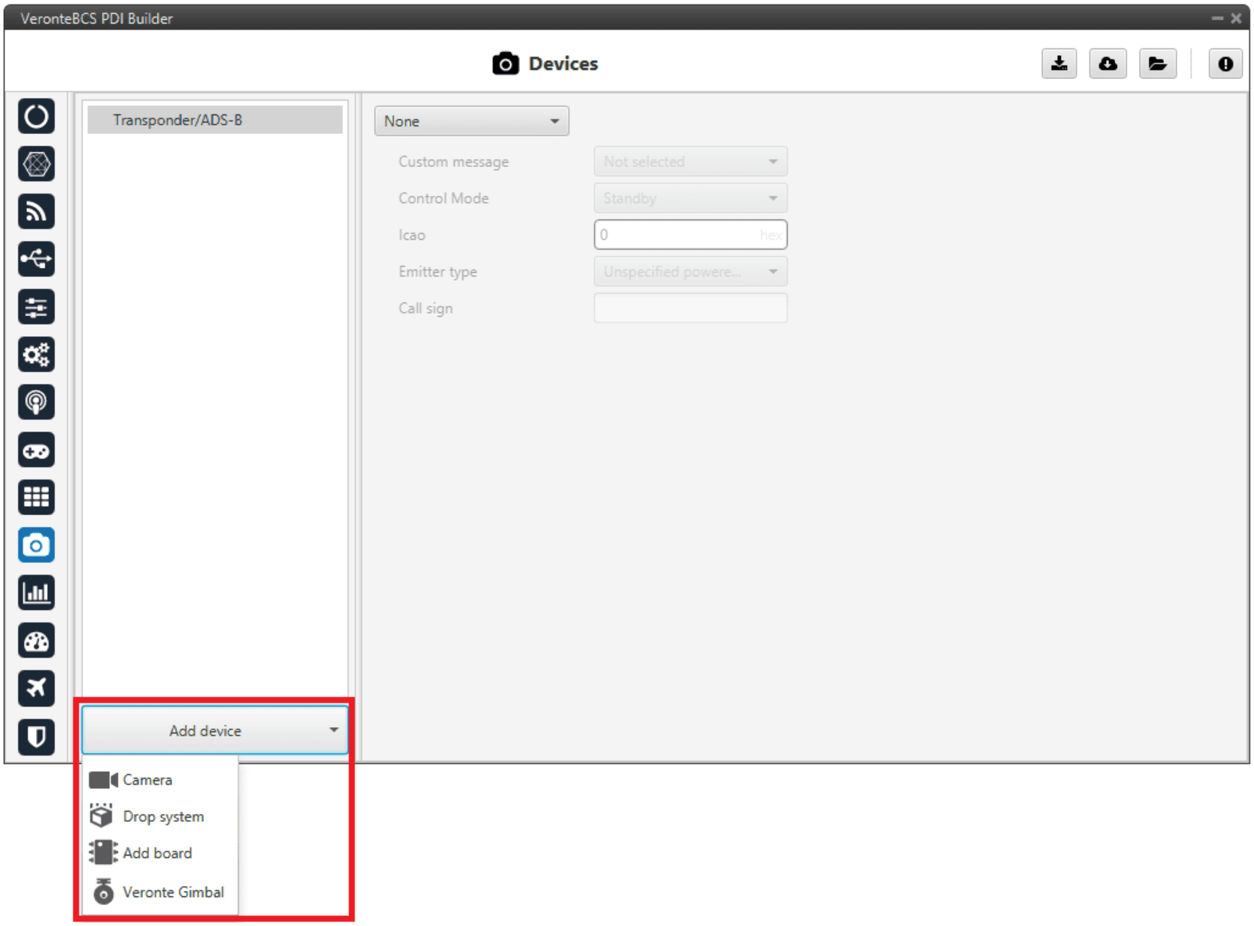
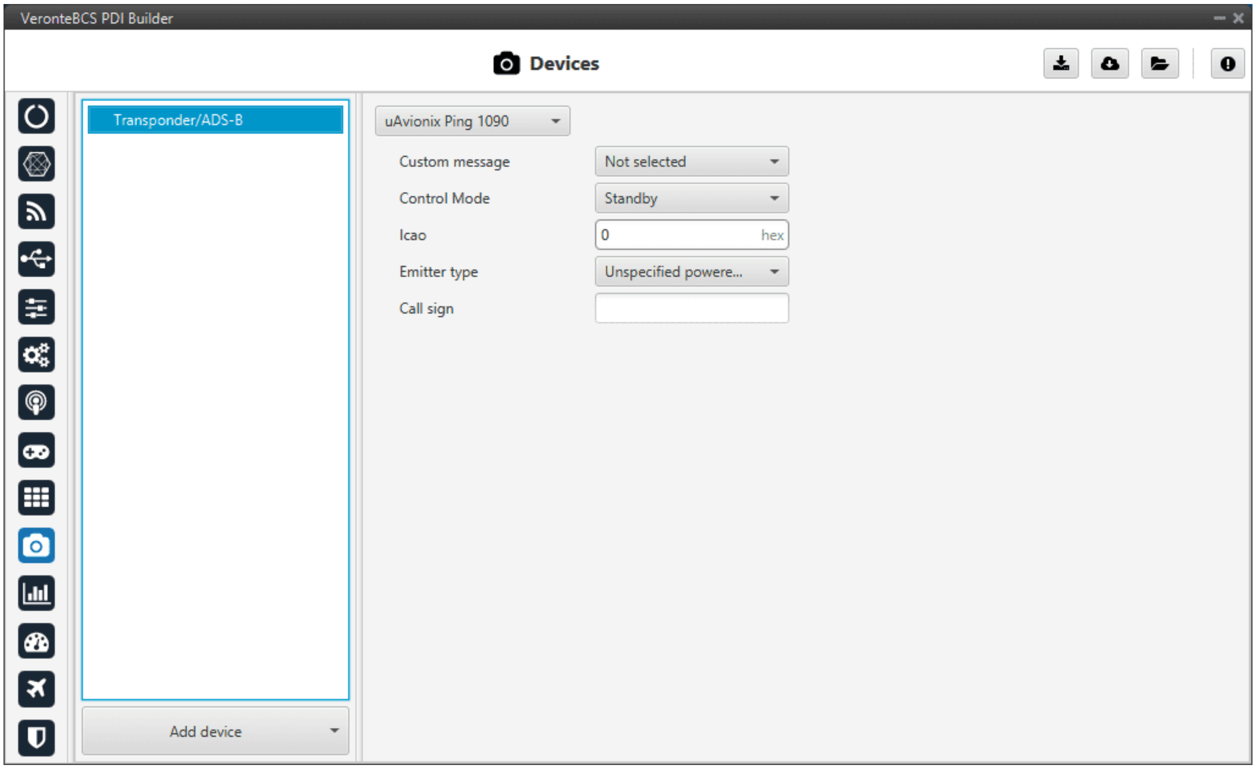


Fig. 125: **Add devices**

2.10.1 Transponder/ADS-B

A transponder is a device that generates or parses **ADS-B messages**. This menu allows the user to use/configure a transponder through **Custom Messages**.

The interface includes different transponder models, depending on the one selected, some configurable parameters are enabled or disabled.



uAvionix Ping 1090

Custom message: Not selected

Control Mode: Standby

Icao: 0 hex

Emitter type: Unspecified powere...

Call sign:

The parameters that appear in this menu and that must be configured by the user depending on the transponder model are explained below:

- **Model**: The options available in **Veronte BCS** are as follows:
  - Internal Transponder
  - uAvionix Ping 20s
  - uAvionix Ping 1090
  - Sagetech MX Series
  - Sagetech XPS-TR
  - Sagetech XPG-TR
  - Sagetech XPS-TRB
  - Daedalean
- **Custom message**: Select the *Custom message 1-3* to be used for the information exchange, as it will be automatically filled with the information required by the transponder/ADS-B.

  Note: This option is **not available** for the **Internal Transponder** model, as it is already integrated in the **BCS**.

- **Control Mode**: The user must configure the transponder type:
  - **Standby**: Transponder will not respond to interrogation.
  - **ADS-B In**: When the transponder sends the detected UAVs.

Note: This option is not admitted for the **uAvionix Ping 20s** model.

- **ADS-B Out**: When the transponder wants to send the position of the UAV.

Note: This option is not admitted for the **Deadelean** model.

- **ADS-B In/Out**: This option is not admitted for **uAvionix Ping 20s** and **Daedalean** models.

- **Icao**: Unique ICAO 24-bit address permanent for the aircraft, which becomes a part of the aircraft's Certificate of Registration.

  It is represented by six **hexadecimal** characters.

  Note:

  - This option is **not available** for **Sagetech** and **Daedalean** models.
  - If the **ADS-B In** control mode has been selected, this option will be disabled.

- **Emitter type**: The user should specify the type/category of the vehicle in which the transponder is located. The available options are:

  Note:

  - This option is **not available** for **Sagetech** and **Daedalean** models.
  - If the **ADS-B In** control mode has been selected, this option will be disabled.

- **Call sign**: Communication call sign assigned as unique identifier to the aircraft.

  Note:

  - This option is **not available** for **Sagetech** and **Daedalean** models.
  - If the **ADS-B In** control mode has been selected, this option will be disabled.

2.10.2 Cameras

Adding a camera will create a default Camera configuration menu.

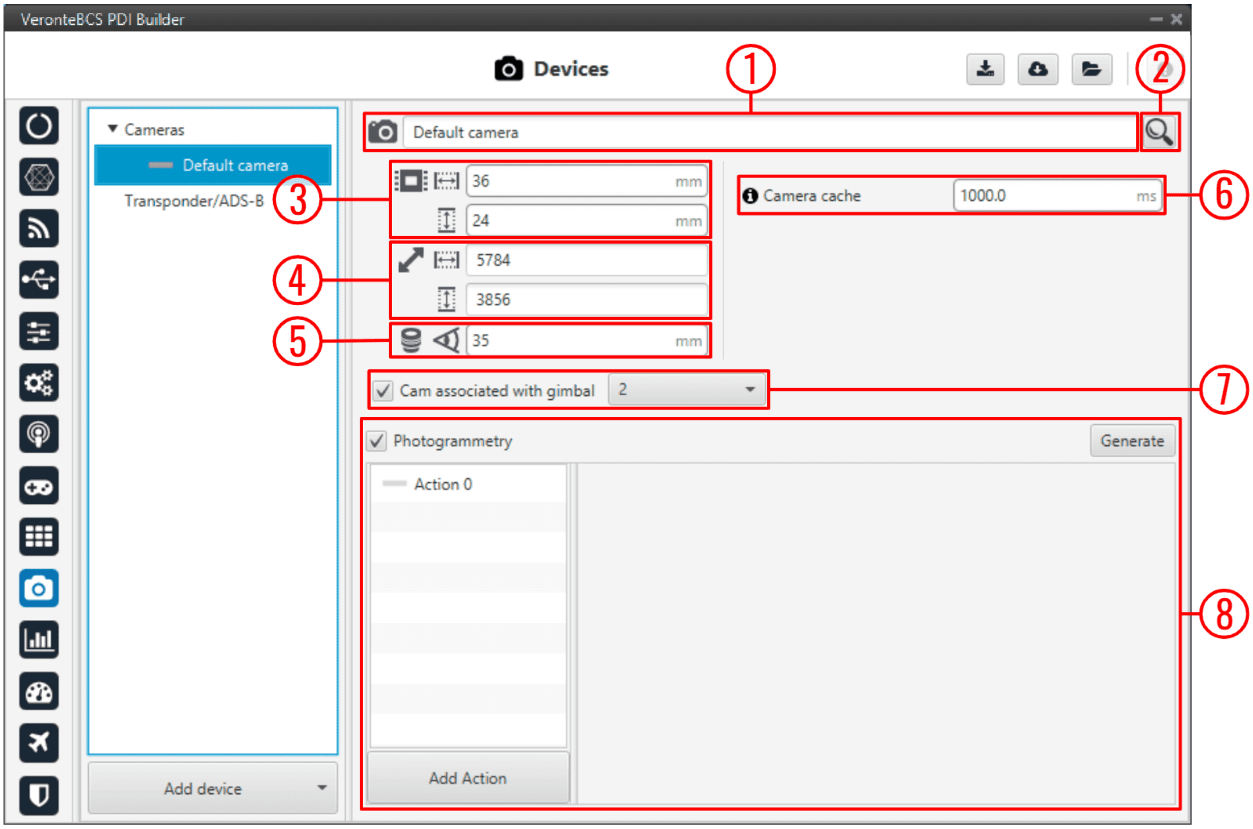


Fig. 126: **Cameras section**

1. **Name**: Give a name to the customized camera.
2. **Search**: Users can also choose a camera from the predefined list of cameras, which will automatically establish the values of the **Sensor**, **Resolution** and **Lens** parameters.
3. **Sensor**: Defines the camera sensor width and height in mm.
4. **Resolution**: Defines the camera resolution width and height.
5. **Lens**: Defines the focal length from the camera in mm.
6. **Camera cache**: Defines the **cache time used to play the selected camera on the gimbal widget**.
   - A **higher** cache might increase the **video delay**.
   - A **lower cache** might cause **video artifacts or disconnections**.

   Tip: 333 miliseconds should be enough for a 1080p video.

7. **Cam associated with gimbal**: If the camera is **from a Gimbal device**, it is important to configure this field and select the **the Gimbal block number** that is related to this camera.
8. **Photogrammetry**: This allows the creation of Photogrammetry actions.

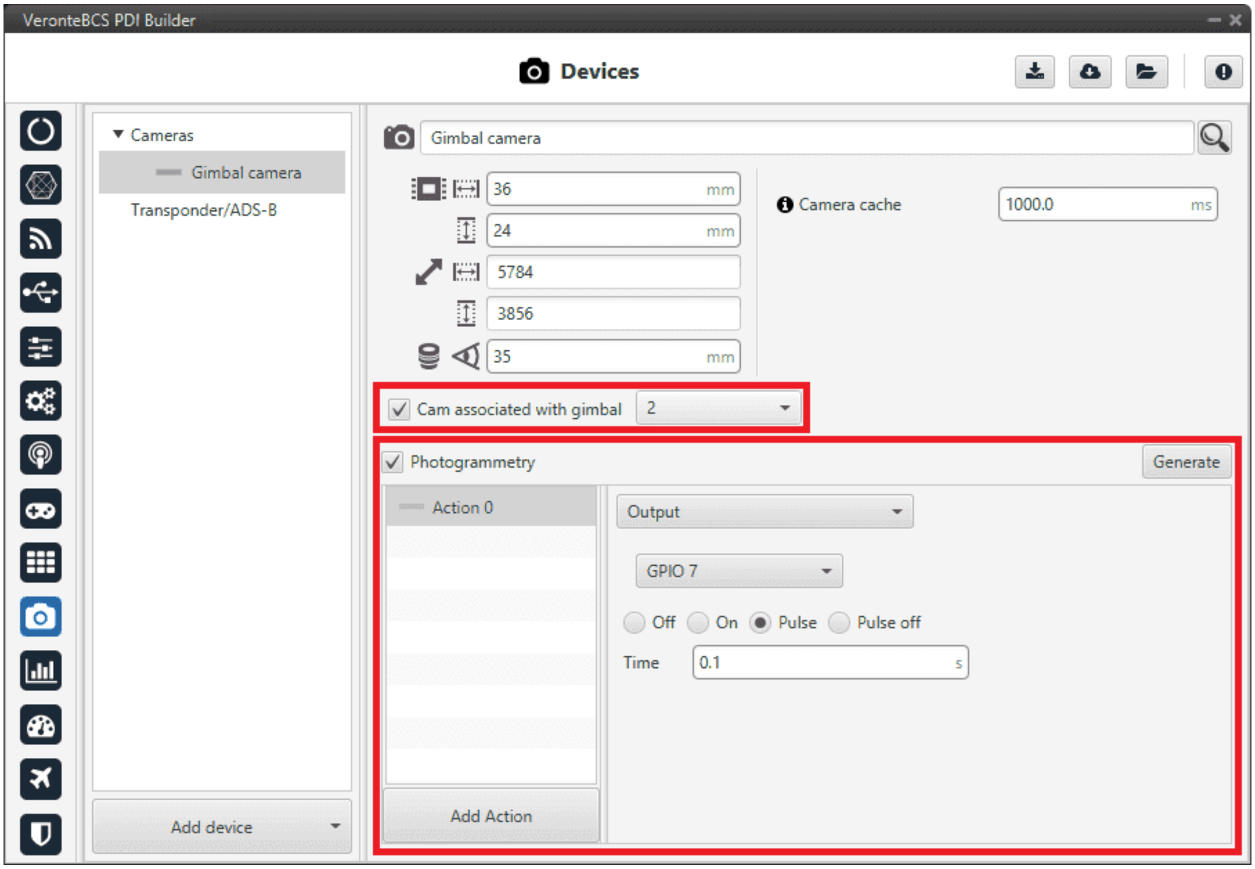
   The actions performed in a Photogrammetry mission can be defined here, following the same possibilities as in *Actions - Automations*.

   - **Add Action**: Will add a new action.

> **Warning:** A **maximum of 4 Actions** can be defined (*Actions 0-3*).

- **Generate**: Clicking on 'Generate' will create the automation 'Photogrammetry' with a Button as event and with the actions defined here.

An example is given below:



Gimbal camera

36 mm

24 mm

5784

3856

35 mm

Camera cache 1000.0 ms

☑ Cam associated with gimbal 2

☑ Photogrammetry

Output

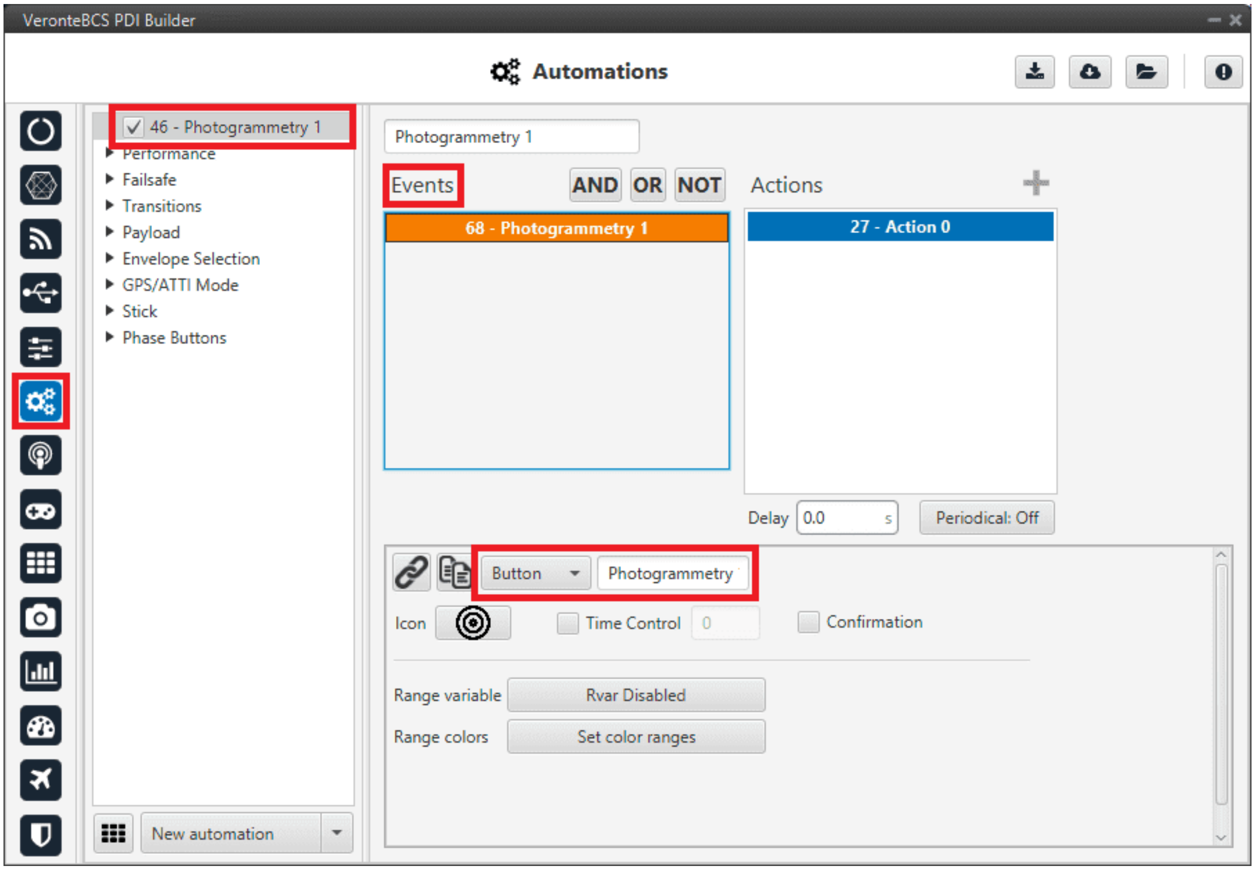
GPIO 7

○ Off ○ On ● Pulse ○ Pulse off

Time 0.1 s

Fig. 127: **Cameras - Example**

The automation created for Photogrammetry is shown below:



☑ 46 - Photogrammetry 1

Photogrammetry 1

68 - Photogrammetry 1

27 - Action 0

Delay 0.0 s

Periodical: Off

Button | Photogrammetry

□ Time Control 0

□ Confirmation

Range variable Rvar Disabled

Range colors Set color ranges

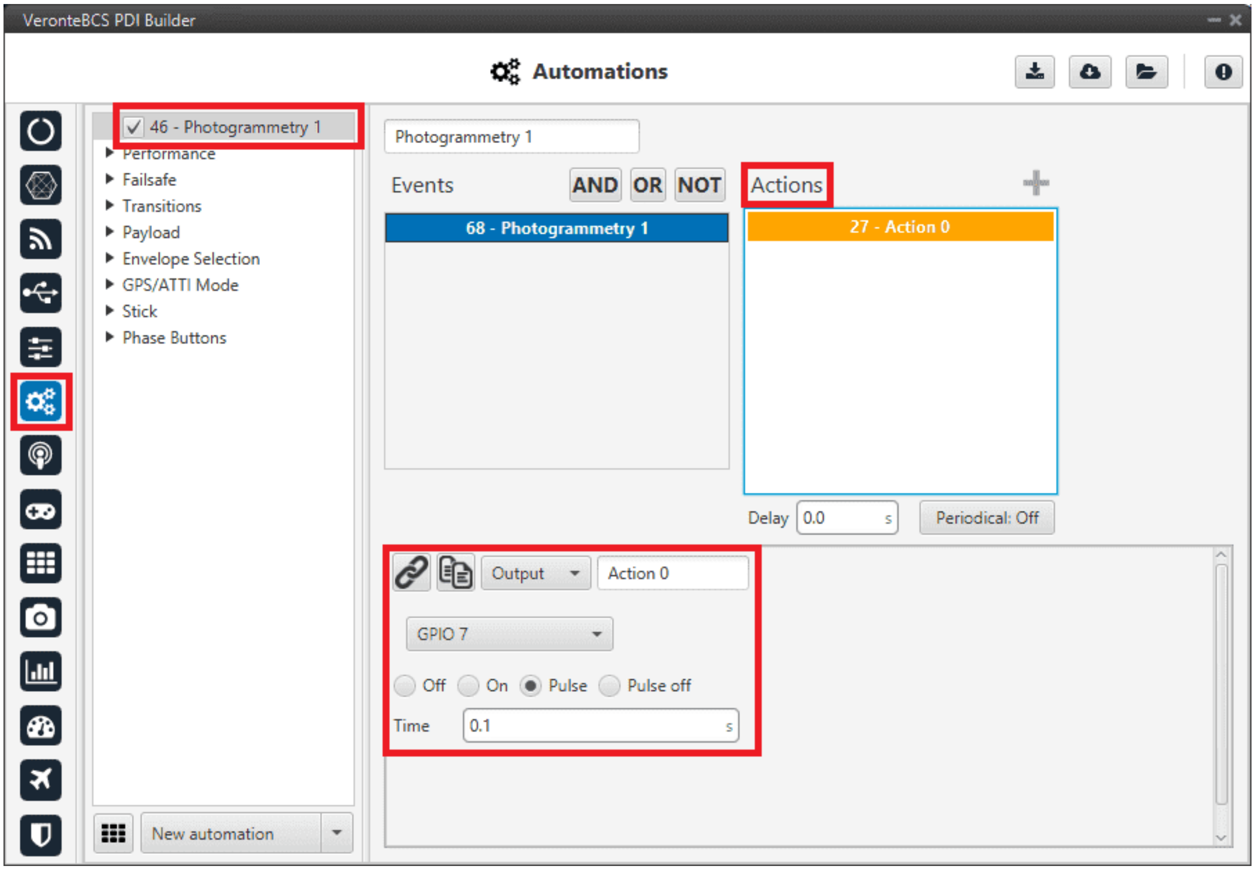
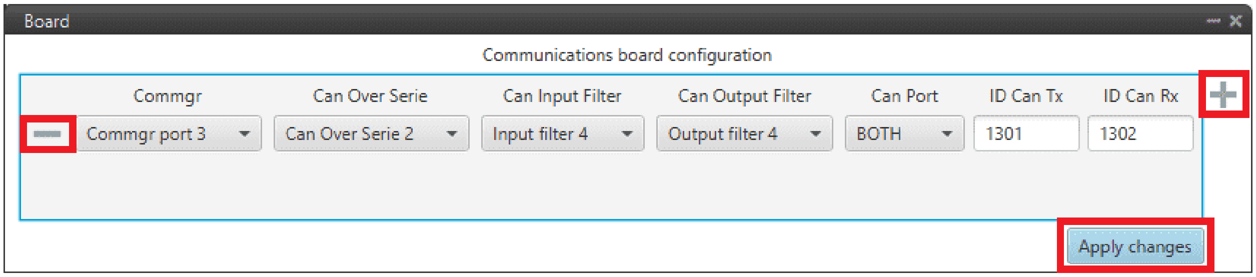


Fig. 128: **Cameras - Automation example**

2.10.3 Board

This board menu allows the user to configure **communicate via CAN**, in **BCS PDI Builder**, with another device such as *CEX*, *MEX*, *MC01*, etc., in **only 1 step**, so in only 1 interface window. Instead of doing it in several steps, as is explained in *CEX - Integration examples* or *MC01 - Integration examples*.



Commgr	Can Over Serie	Can Input Filter	Can Output Filter	Can Port	ID Can Tx	ID Can Rx
Commgr port 3	Can Over Serie 2	Input filter 4	Output filter 4	BOTH	1301	1302

Fig. 129: **Board section**

- Click on the + icon to add a new field. The following parameters must be defined:
  - **Commgr port**: Select the desired COM Manager port: *Commgr port 1-6*.
  - **Can Over Serie**: Select the desired **Serial to CAN / CAN to serial**: *Serial to CAN 1-2/CAN to Serial 1-2*.
  - **Can Input Filter**: Select the desired input filter to use: *Input Filter 1-2*.

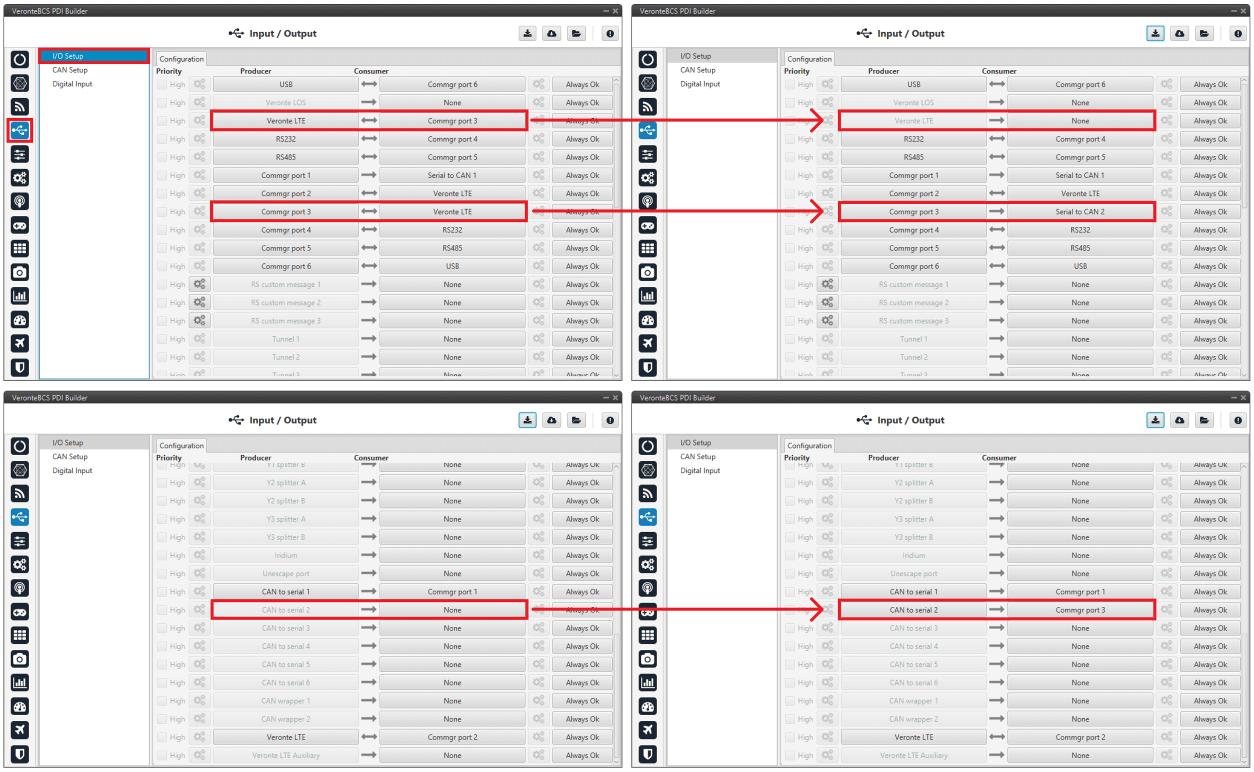
- **Can Output Filter**: Select the output filter the user wishes to use: *Output Filter 1-2*.
- **Can Port**: *CAN A*, *CAN B* or *BOTH* can be selected.
- **ID Can Tx**: Enter the ID of the CAN message to be sent.
- **ID Can Rx**: Enter the ID of the CAN message to be received.

> **Warning:** Be careful not to select a producer/consumer that is being used for another purpose, as the configuration defined here has "priority" and will be changed to this.

For more information on these parameters, see *Input/Output section* of this manual.

- Clicking the ▬ icon will remove the field and display a confirmation warning message.
- By clicking on '**Apply changes**', all these CAN communication settings are applied to the **BCS** configuration.

Below is an example of before/after when changes are applied:



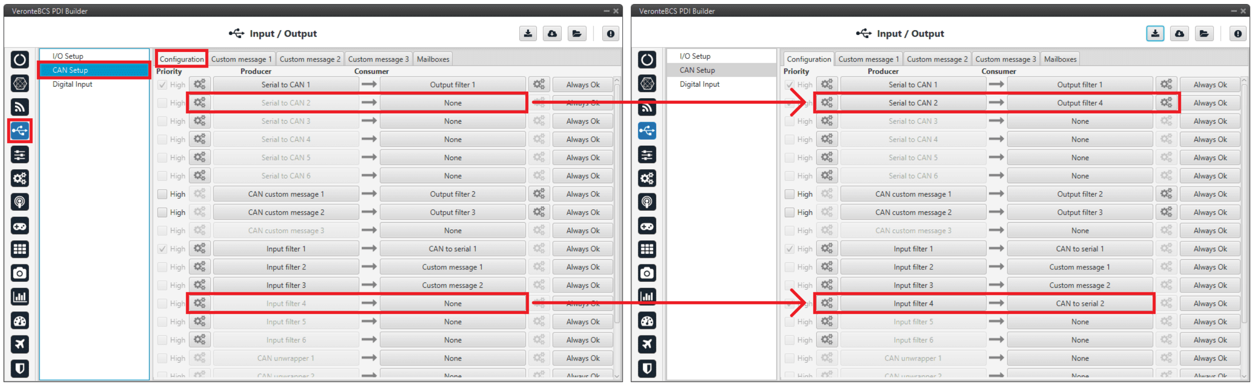
Priority	Producer		Consumer	
High	USB	⟷	Commgr port 6	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	⟷	Commgr port 3	Always Ok
High	RS232	⟷	Commgr port 4	Always Ok
High	RS485	⟷	Commgr port 5	Always Ok
High	Commgr port 1	→	Serial to CAN 1	Always Ok
High	Commgr port 2	⟷	Veronte LTE	Always Ok
High	Commgr port 3	⟷	Veronte LTE	Always Ok
High	Commgr port 4	⟷	RS232	Always Ok
High	Commgr port 5	⟷	RS485	Always Ok
High	Commgr port 6	⟷	USB	Always Ok
High	RS custom message 1	→	None	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Priority	Producer		Consumer	
High	USB	⟷	Commgr port 6	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	→	None	Always Ok
High	RS232	⟷	Commgr port 4	Always Ok
High	RS485	⟷	Commgr port 5	Always Ok
High	Commgr port 1	→	Serial to CAN 1	Always Ok
High	Commgr port 2	⟷	Veronte LTE	Always Ok
High	Commgr port 3	→	Serial to CAN 2	Always Ok
High	Commgr port 4	⟷	RS232	Always Ok
High	Commgr port 5	⟷	RS485	Always Ok
High	Commgr port 6	⟷	USB	Always Ok
High	RS custom message 1	→	None	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Priority	Producer		Consumer	
High	Y1 splitter B	→	None	Always Ok
High	Y2 splitter A	→	None	Always Ok
High	Y2 splitter B	→	None	Always Ok
High	Y3 splitter A	→	None	Always Ok
High	Y3 splitter B	→	None	Always Ok
High	Iridium	→	None	Always Ok
High	Unescape port	→	None	Always Ok
High	CAN to serial 1	→	Commgr port 1	Always Ok
High	CAN to serial 2	→	None	Always Ok
High	CAN to serial 3	→	None	Always Ok
High	CAN to serial 4	→	None	Always Ok
High	CAN to serial 5	→	None	Always Ok
High	CAN to serial 6	→	None	Always Ok
High	CAN wrapper 1	→	None	Always Ok
High	CAN wrapper 2	→	None	Always Ok
High	Veronte LTE	⟷	Commgr port 2	Always Ok
High	Veronte LTE Auxiliary	→	None	Always Ok

Priority	Producer		Consumer	
High	Y1 splitter B	→	None	Always Ok
High	Y2 splitter A	→	None	Always Ok
High	Y2 splitter B	→	None	Always Ok
High	Y3 splitter A	→	None	Always Ok
High	Y3 splitter B	→	None	Always Ok
High	Iridium	→	None	Always Ok
High	Unescape port	→	None	Always Ok
High	CAN to serial 1	→	Commgr port 1	Always Ok
High	CAN to serial 2	→	Commgr port 3	Always Ok
High	CAN to serial 3	→	None	Always Ok
High	CAN to serial 4	→	None	Always Ok
High	CAN to serial 5	→	None	Always Ok
High	CAN to serial 6	→	None	Always Ok
High	CAN wrapper 1	→	None	Always Ok
High	CAN wrapper 2	→	None	Always Ok
High	Veronte LTE	⟷	Commgr port 2	Always Ok
High	Veronte LTE Auxiliary	→	None	Always Ok

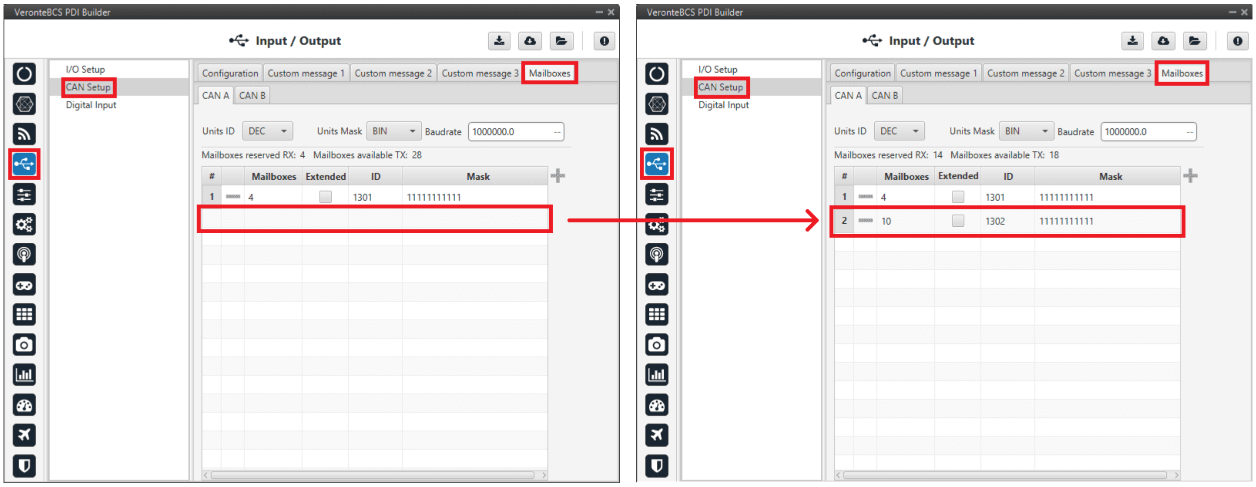
Fig. 130: **Board - I/O Setup configuration**



Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
□ High	Serial to CAN 2	→	None	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message 1	→	Output filter 2	Always Ok
□ High	CAN custom message 2	→	Output filter 3	Always Ok
□ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
□ High	Input filter 2	→	Custom message 1	Always Ok
□ High	Input filter 3	→	Custom message 2	Always Ok
□ High	Input filter 4	→	None	Always Ok
□ High	Input filter 5	→	None	Always Ok
□ High	Input filter 6	→	None	Always Ok
□ High	CAN unwrapper 1	→	None	Always Ok
□ High	CAN unwrapper 2	→	None	Always Ok

Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
☑ High	Serial to CAN 2	→	Output filter 4	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message 1	→	Output filter 2	Always Ok
□ High	CAN custom message 2	→	Output filter 3	Always Ok
□ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
□ High	Input filter 2	→	Custom message 1	Always Ok
□ High	Input filter 3	→	Custom message 2	Always Ok
☑ High	Input filter 4	→	CAN to serial 2	Always Ok
□ High	Input filter 5	→	None	Always Ok
□ High	Input filter 6	→	None	Always Ok
□ High	CAN unwrapper 1	→	None	Always Ok
□ High	CAN unwrapper 2	→	None	Always Ok

Fig. 131: **Board - CAN Setup configuration**



Units ID DEC Units Mask BIN Baudrate 1000000.0

Mailboxes reserved RX: 4 Mailboxes available TX: 28

#	Mailboxes	Extended	ID	Mask
1	4	□	1301	11111111111

Units ID DEC Units Mask BIN Baudrate 1000000.0

Mailboxes reserved RX: 14 Mailboxes available TX: 18

#	Mailboxes	Extended	ID	Mask
1	4	□	1301	11111111111
2	10	□	1302	11111111111

Fig. 132: **Board - Mailboxes configuration**

2.11 Telemetry

In this section there are 2 options available: **Telemetry** and **Sniffer**.

2.11.1 Telemetry

Telemetry controls permit to configure data to be stored or transmitted on the system. There are 4 main items that can be configured within this panel:

Type	Description
Data to VApp	Configures the variables to send throughout the data link channel.
Onboard Log	Sets the variables to be stored on system Log. (on SD Card)
User Log	User Log for custom applications.
Fast Log	Saves data at the maximum frequency available on the system. Recording time depends on the selected variables.

Configuration display permits to enable the desired variables for each telemetry file and to set the maximum and minimum values together with precision for each one.

2.11.1.1 Data to VApp

This menu contains the variables sent **between BCS and Veronte Ops**. By default, the system provides one Data link that represents the connection between the air autopilot and the software (Veronte Ops).

1x autopilot air unit sends the variables to **BCS**, being processed when they arrive there by Veronte Ops. The variables indicated in **red** in a Data to Vapp are **required for correct operation** of Veronte Ops.

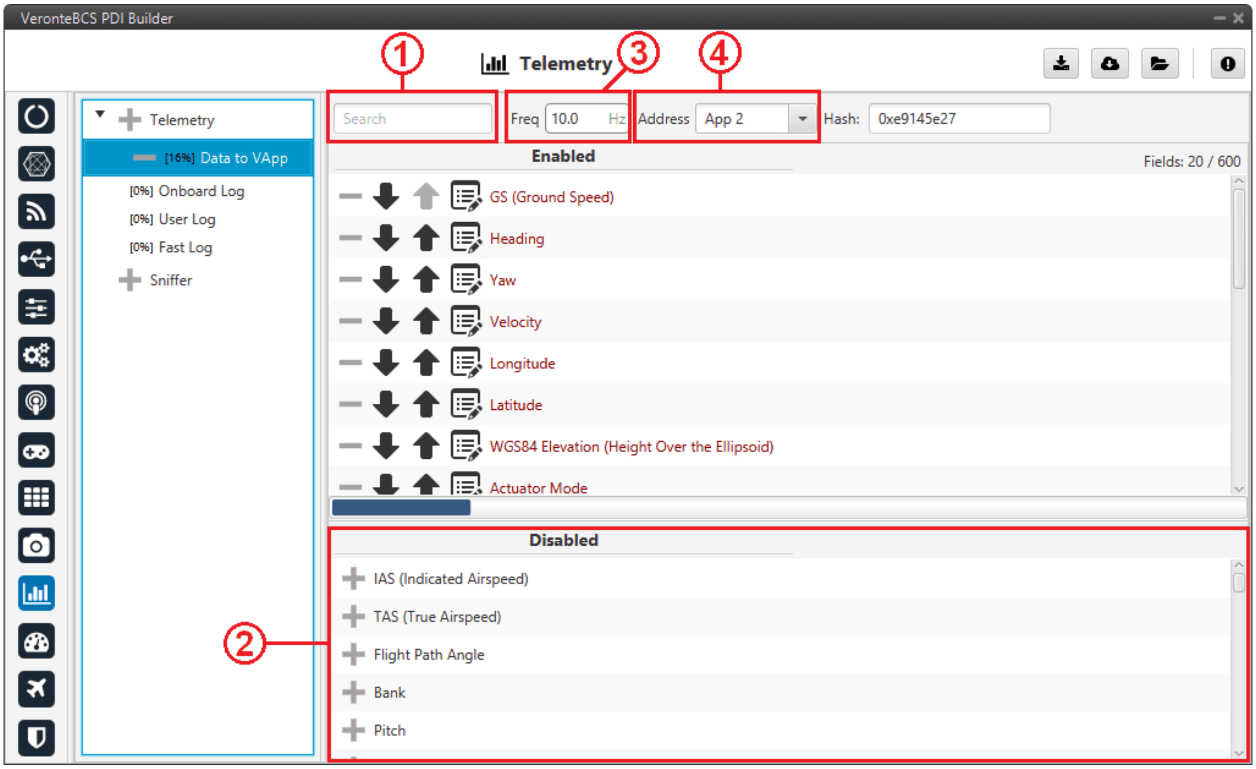


Fig. 133: **Data to VApp section**

In order to configure the variables sent, users have to:

1. **Search**: Search the desired variables into the **Disabled** panel.
2. **Disabled**: When the desired variables are found, add them to the **Enabled** panel by dragging and dropping them into it or simply by clicking on the + button.
3. **Freq**: Specify the sending rate. 10 Hz usually works well, this frequency depends on the bandwidth of the radio.
4. **Address**: Select the corresponding address, the option available are *App 2* (Veronte Ops address), *Broadcast* (all units on the network) and *Veronte v4.X XXXX* (to a specific unit). Usually **App 2**.

Note: **Hash** parameter is not configurable, it is automatically calculated by the system based on the telemetry vector configured by the user.

It is a hexadecimal representation of the CRC of the fieldset.

BCS PDI Builder allows the creation of more Data links, the user can add it by simply pressing in the "**+**" icon next to 'Telemetry'.

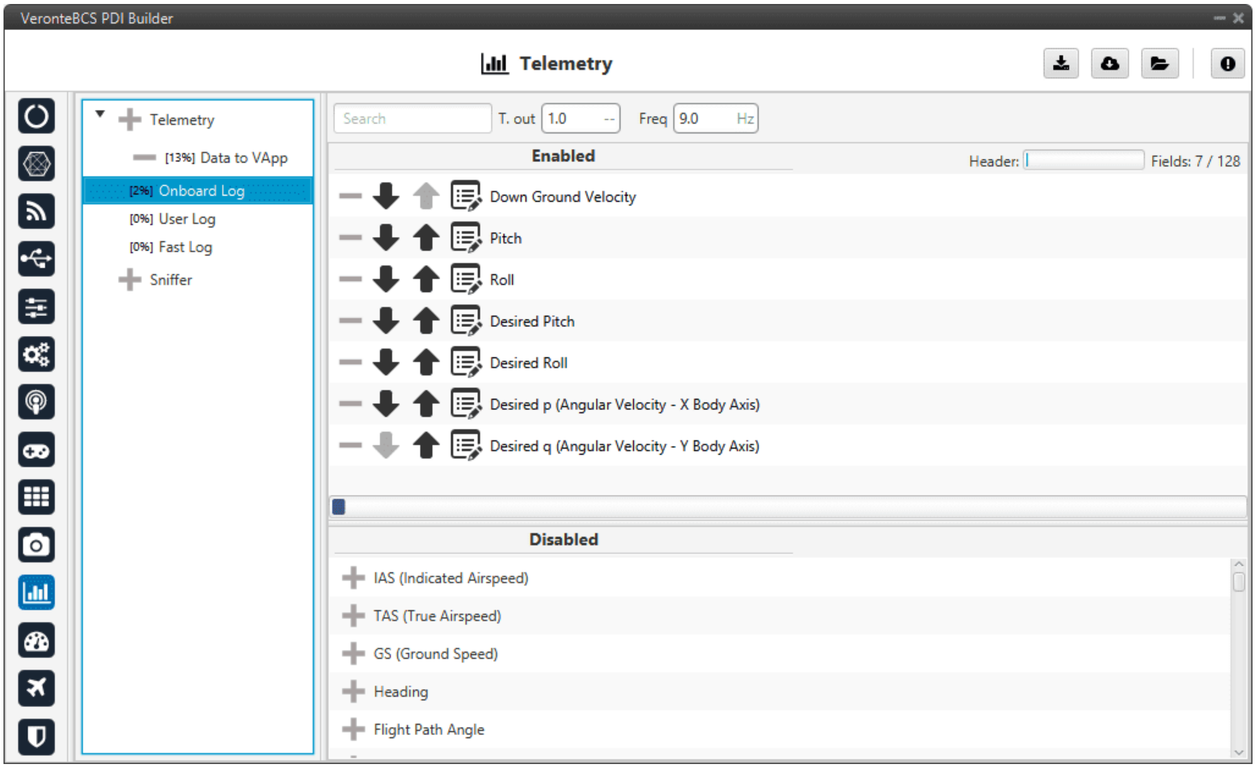
Warning: If the number of variables enabled for telemetry communication are higher than the maximum supported by the system, the latest variables will not be sent, so they will display a zero value if shown in the workspace.

Note: It is possible to create more than one data link associated to the same receiver address, and they can also have different sending rates. It could be useful in case one of the data links is almost full.

2.11.1.2 Onboard Log

The Onboard Log determines the variables that are being **stored** on the **autopilot SD Card**. In this case, there are not sending/receiving units, so the only thing to configure here is the list of variables that will be saved on the autopilot internal memory for a further download and processing, as well as the writing frequency.

The log starts writing once the autopilot is turned on and **does not stop logging until the autopilot is turned off**.



T. out 1.0 -- Freq 9.0 Hz

Header: Fields: 7 / 128

Enabled:
- Down Ground Velocity
- Pitch
- Roll
- Desired Pitch
- Desired Roll
- Desired p (Angular Velocity - X Body Axis)
- Desired q (Angular Velocity - Y Body Axis)

Disabled:
- IAS (Indicated Airspeed)
- TAS (True Airspeed)
- GS (Ground Speed)
- Heading
- Flight Path Angle

Fig. 134: **Onboard Log section**

> **Warning:** This is a circular log, which means that if the SD card memory is full, **Veronte BCS** will delete the oldest logs automatically so it can continue logging.
>
> **BCS** has **2.5 GB** of memory reserved for these logs. Hence the registered time can be calculated, for example:
>
> - If 10 variables are stored with 4 bytes each one, then each log will occupy 40 bytes
> - With a frequency of 10 Hz, the writing speed will be 400 bytes/s
> - $\frac{2.5\,GB}{400\,bytes/s} = 6710886\,s = 1864\,h = 77.7\,days$

2.11.1.3 User Log

The user log contains the variables that are stored according to an automation created by the user.

Considering an example, in a photogrammetry mission it is important to record the aircraft location when the photo is taken, so a user log could be used to record a certain set of variables (position, speed, direction, . . .) each time a photo is taken.

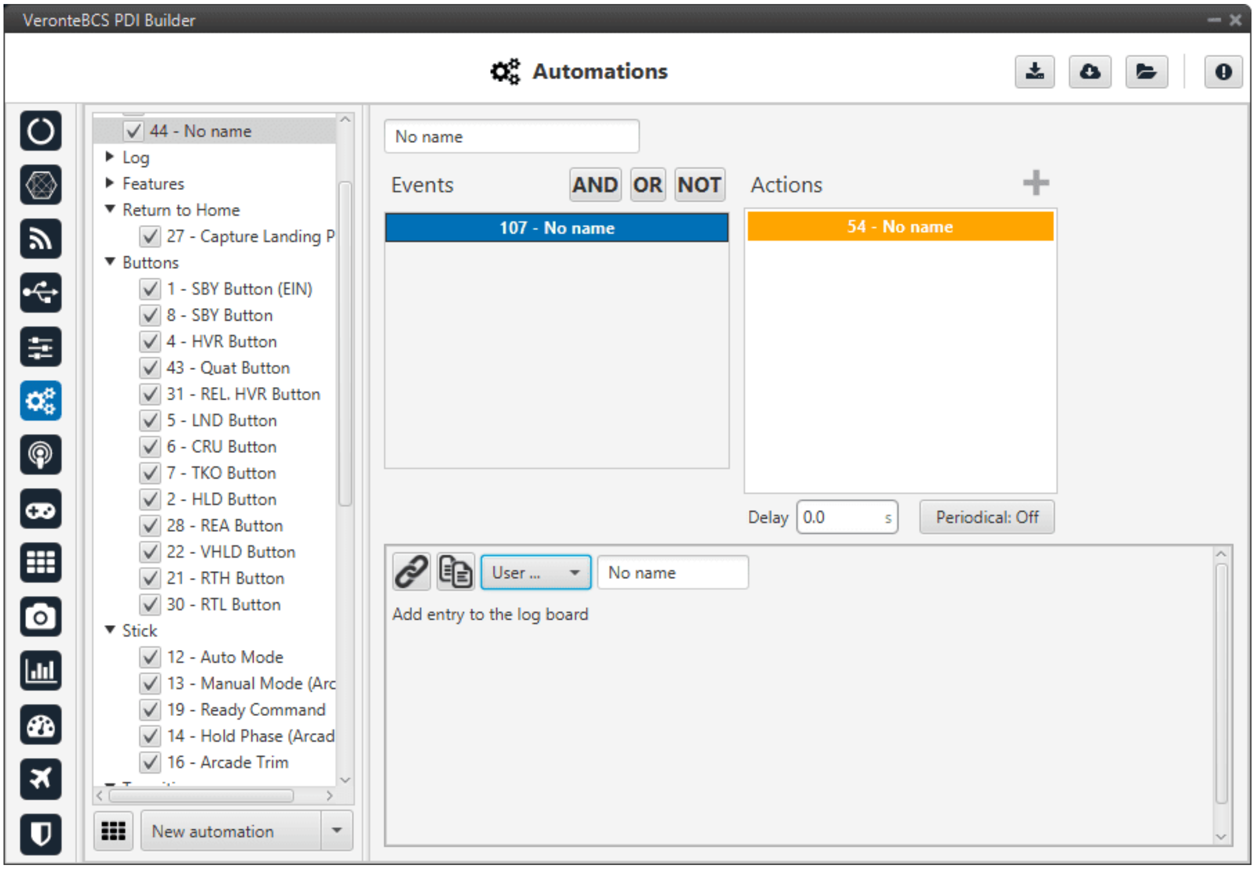


Fig. 135: **User Log section**

In order to create a **User Log action** where an entry is added to the log when a certain set of events are accomplished check *Actions* section of the Automations menu.

2.11.1.4 Fast Log

The fast log store the specified variables at the maximum rate available on the system. This tool could be used to save information in an operation that happens extremely fast, such as missile launching. The time that this logging process lasts depends on the number of variables being saved.

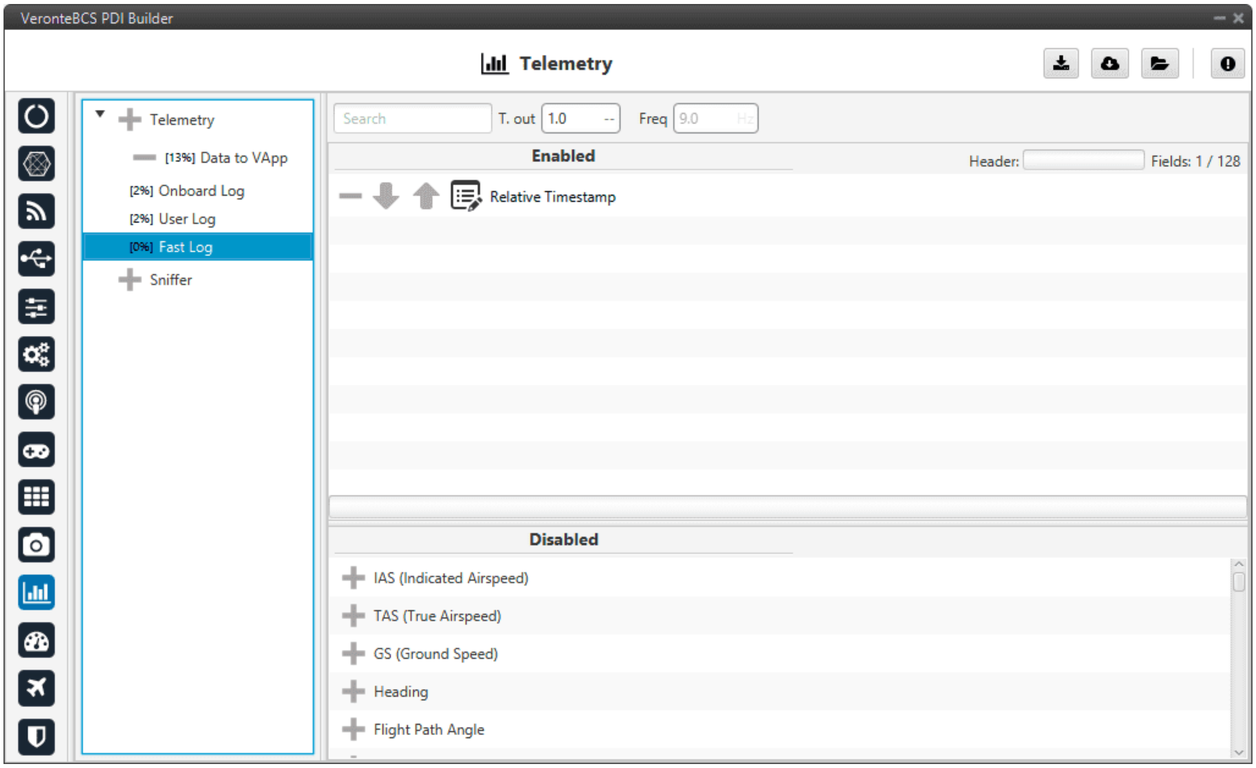


Fig. 136: **Fast Log section**

The Fast Log can be activate when the user want, but must be done in Veronte Ops. For more information about Fast Log, click on the Veronte Ops manual.

The downloading of the information of an operation depends on how it has been stored, i.e depends on the type of log (data link, onboard, user or fast). Visit FDR manual for information related to **Onboard Log**, **User Log** and **Fast log** downloading. And Veronte Link manual for information about **Data to VApp**.

Besides, **BCS** includes some compression tools that may be useful for increasing the amount of information transmitted in a certain bandwith or stored in a log. Each variable can be compressed separately in each log.

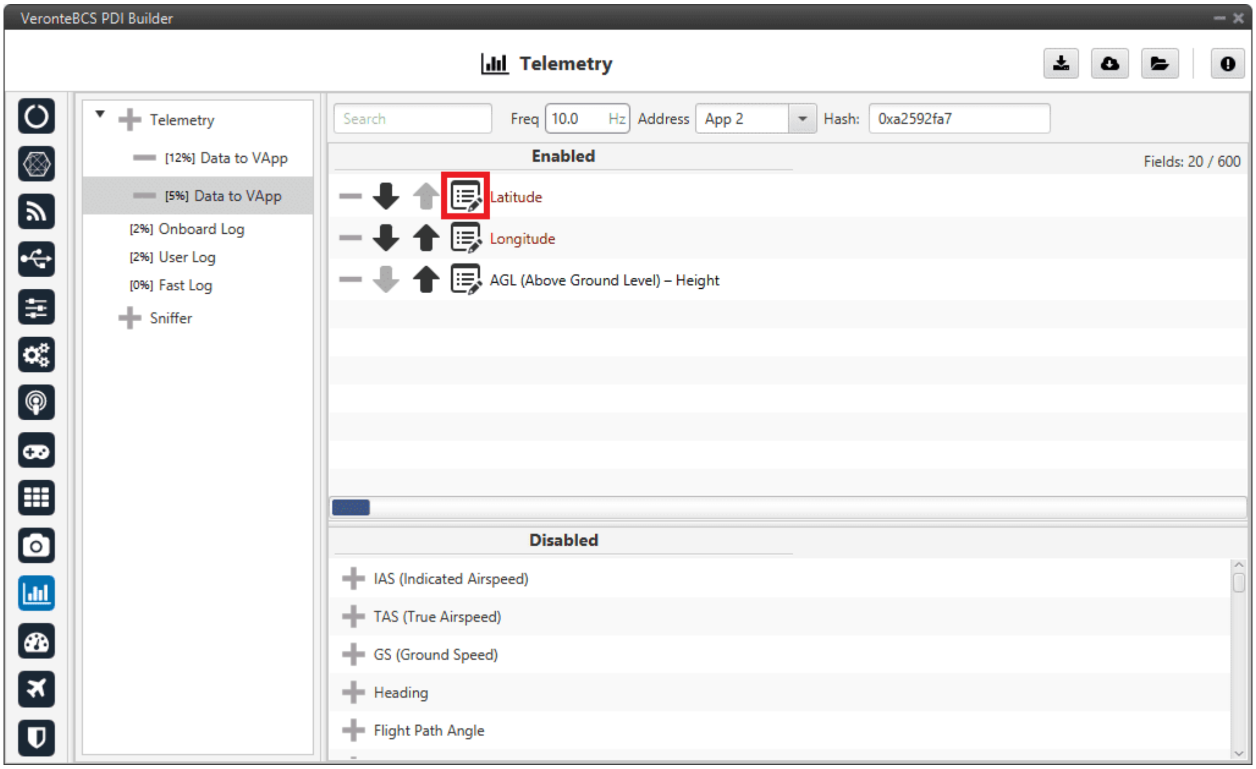
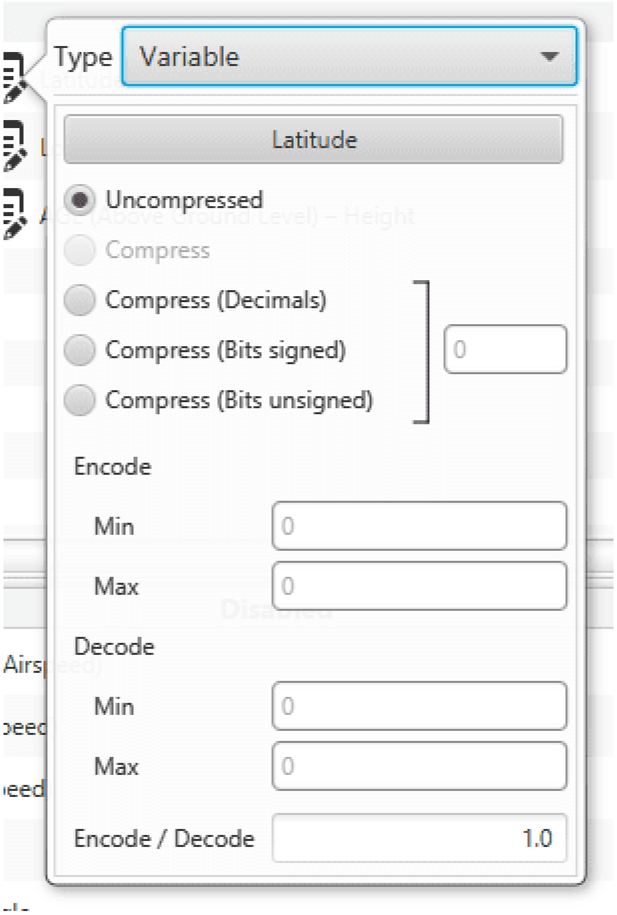


Fig. 137: **Compression options**

There are different types of compression available:



Type: Variable

● Uncompressed

○ Compress

○ Compress (Decimals)

○ Compress (Bits signed)

○ Compress (Bits unsigned)

0

Encode

Min: 0

Max: 0

Decode

Min: 0

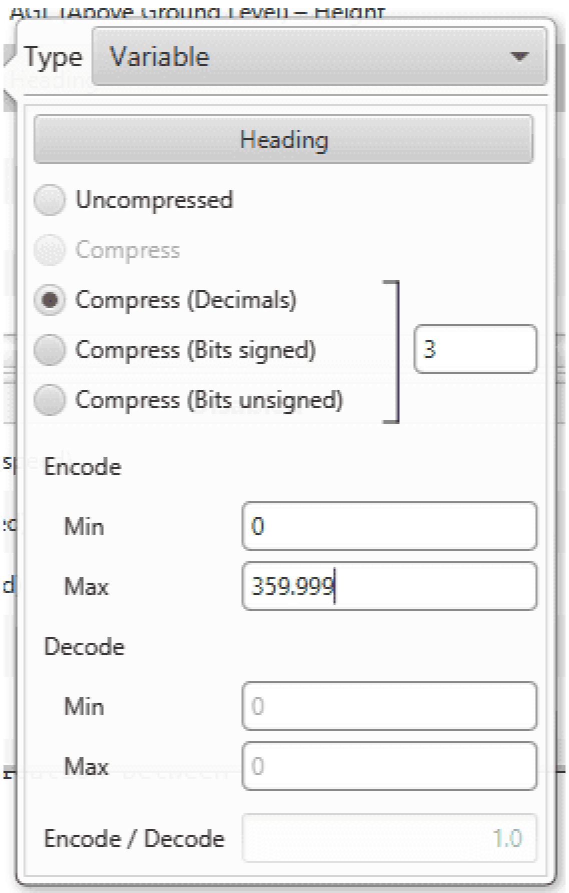
Max: 0

Encode / Decode: 1.0

Fig. 138: **Compression options panel**

- **Uncompressed**: The variable is taken in its full length, with no value modification.
- **Compress (Bits signed)**: Specify the number of bits to be compressed to (**negative values accepted**). It is necessary that the user configures **Encode/Decode** options.
- **Compress (Bits unsigned)**: Specify the number of bits to be compressed to (**no negative values accepted**). It is necessary that the user configures **Encode/Decode** options.
- **Compress (Decimals)**: The variable is compressed according to the number of decimals specified and the range specified (max and min values). The resultant compression (number of bits) follows the relation $(max - min) \cdot 10^{decimals}$, which yields the encoding of the maximum value of the range (and the number of bits necessary for that). The range needs to be specified on the **Encode - Min/Max** field.
- **Encode/Decode**: These values are used to apply a scaling factor after the transformation from binary to decimal value, or before the transformation from decimal to binary value.

In the example shown below, the Heading variable with 3 decimals will be compressed, so instead of using 32 bits, it will only require 19 bits.



Type: Variable

Heading

○ Uncompressed
○ Compress
● Compress (Decimals)
○ Compress (Bits signed)
○ Compress (Bits unsigned)

3

Encode

Min: 0
Max: 359.999

Decode

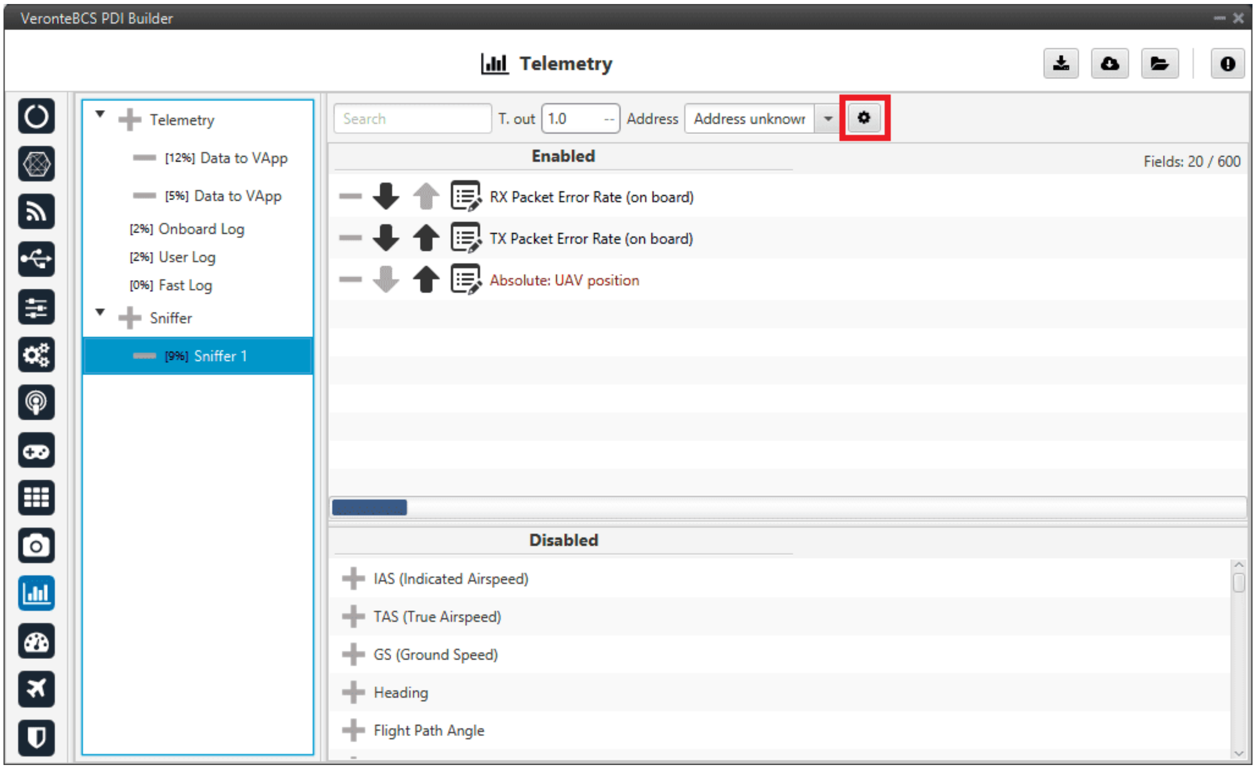
Min: 0
Max: 0

Encode / Decode: 1.0

Fig. 139: **Compression example**

2.11.2 Sniffer

This menu is used to establish a telemetry communication between autopilot and **BCS**. The **BCS** being configured will "listen" the variables indicated in the window **Enabled**, from another autopilot whose address is indicated in **Address**. The sniffer is commonly used to make the **BCS** listen the position of the aircraft and the link quality.



T. out 1.0
Address: Address unknowr
Fields: 20 / 600

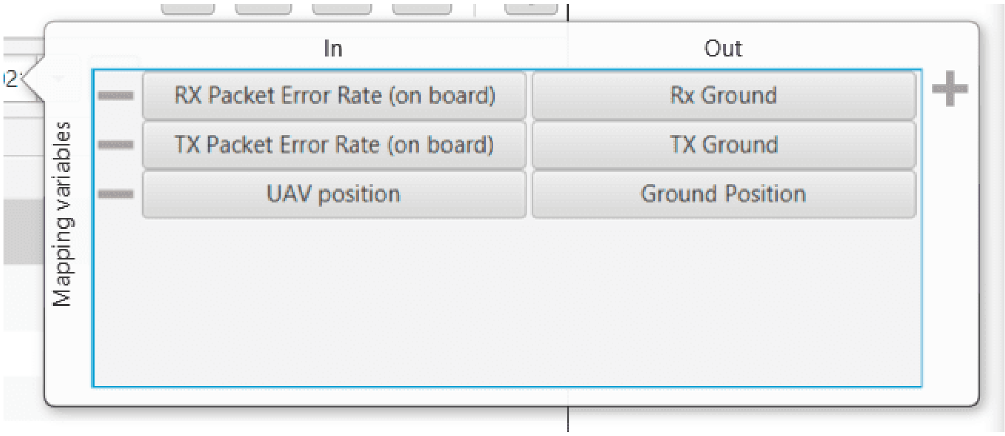
Enabled:
- RX Packet Error Rate (on board)
- TX Packet Error Rate (on board)
- Absolute: UAV position

Disabled:
- IAS (Indicated Airspeed)
- TAS (True Airspeed)
- GS (Ground Speed)
- Heading
- Flight Path Angle

Fig. 140: **Sniffer menu**

The **1x autopilot** unit that sends the data has to be configured as well (**1x** air unit), in the *Telemetry* section. That unit will send telemetry through a **Data Link**.

By clicking on ⚙, the user can access the **Mapping Variables** configuration. Here, the variables sent by the ground unit are indicated in the columm **In**, and they are stored in the variables indicated in **Out** for its later use.



In	Out
RX Packet Error Rate (on board)	Rx Ground
TX Packet Error Rate (on board)	TX Ground
UAV position	Ground Position

Fig. 141: **Mapping variables option**

An example of the configuration required for communication between 1x ground and air units can be found in *Data transmission between Veronte 1x Autopilots -> Integration examples section* of this manual.

2.12 UI

In this menu, the user can manage operation elements and system variables.

2.12.1 Operation elements

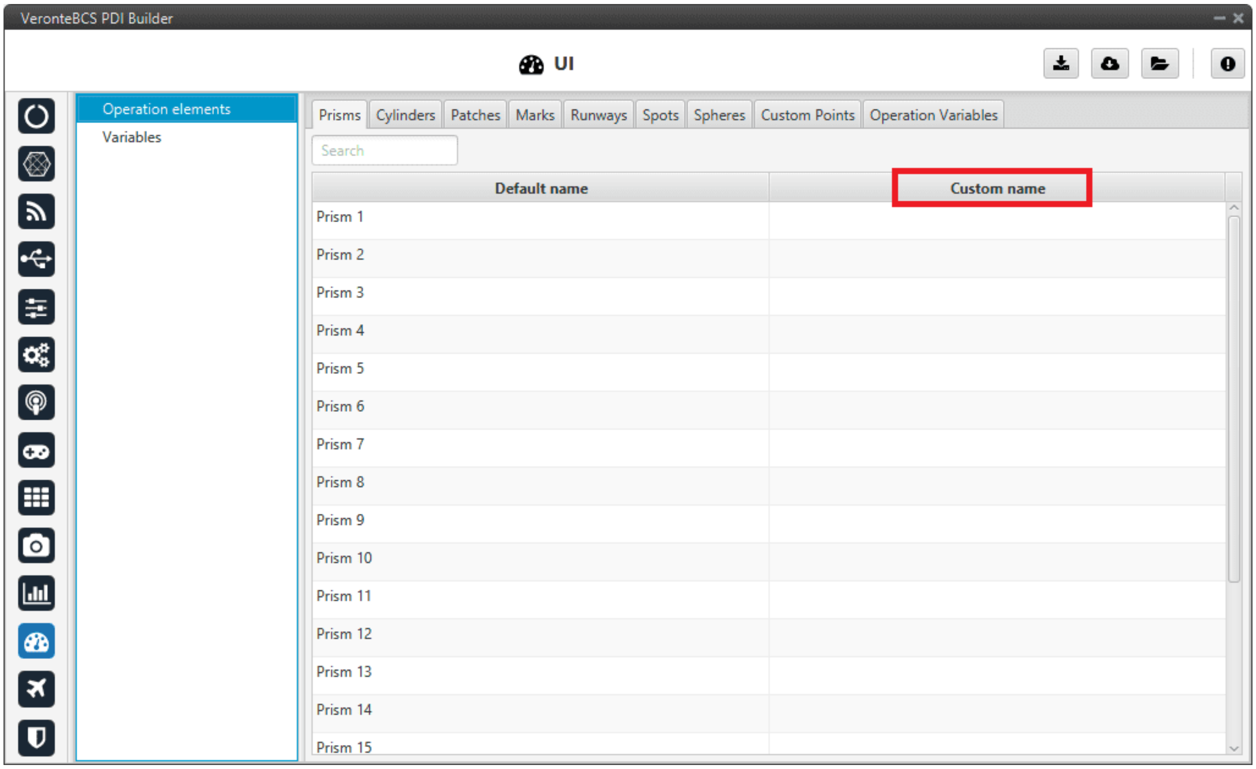
In this section, the user can rename operation elements. These are variables that have no value until an operation is performed, e.g. cruise speed and altitudes, marks to initiate landing, the route, etc.

In addition, they must first be renamed in this section, so that they can then be referenced in the setup, as in Automations (see *Automations* section), and then defined in the operation, in **Veronte Ops** (visit Veront Ops manual).

Operation elements are divided into 9 different types:

- **Custom Points**: An operation custom point is a waypoint, a position variable (x,y,z) that can be used as a reference.
- **Patches**: A patch establishes a path that the UAV can fly to, they make up the route. Therefore patches include waypoints, segments, arcs and orbits.
- **Marks**: A mark is a reference that is placed in a patch and when the uav reaches it, an action takes place.
- **Prisms**: A prism is a detection area.
- **Cylinders**: A cylinder is a circular prism.
- **Runways**: These are the runways used during the take-off and landing phases.
- **Spots**: A spot is a kind of runway with an initial point, direction and azimuth.
- **Spheres**: A sphere is a detection area.
- **Operation Variables**: An operation guidance point is a value of the operation, such as the cruise speed.

  In addition to assigning a specific name, the user can also select the unit of this variable as well as its maximum and minimum values (in SI units).



Default name	Custom name
Prism 1	
Prism 2	
Prism 3	
Prism 4	
Prism 5	
Prism 6	
Prism 7	
Prism 8	
Prism 9	
Prism 10	
Prism 11	
Prism 12	
Prism 13	
Prism 14	
Prism 15	

Fig. 142: **Operation elements section**

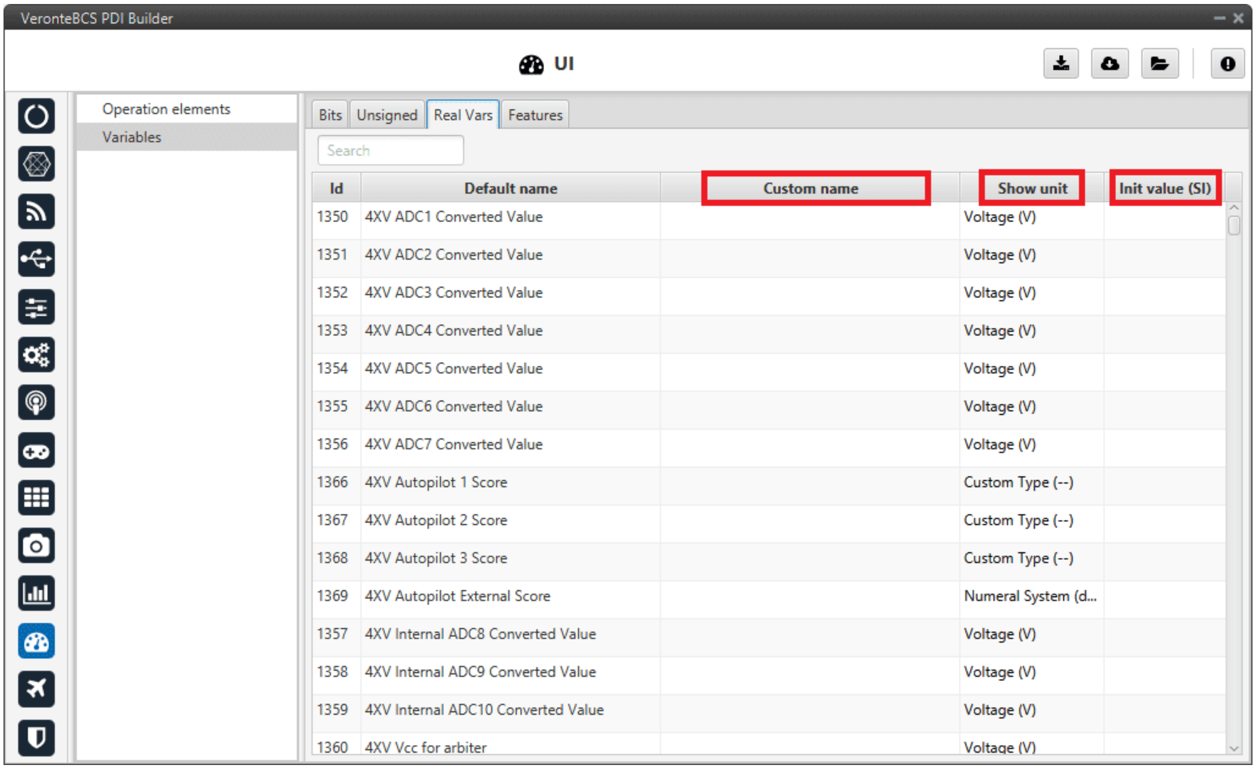
2.12.2 Variables

In this section, the user can find the name of all system variables, as well as their units and initial values. This is very useful, for example, in the case of ‘User Variables’.

Variables are divided into 4 different tabs:

- **Bits**: Bits variables, 1 bit.
- **Unsigned**: Unsigned integer variables, 16 bits.
- **Real Vars**: Real Variables, 32 bits.
- **Features**: Features variables, 64 bits.

Note: There are 300 User variables available for each class (Bits, Unsignet and Real Vars).



Id	Default name	Custom name	Show unit	Init value (SI)
1350	4XV ADC1 Converted Value		Voltage (V)	
1351	4XV ADC2 Converted Value		Voltage (V)	
1352	4XV ADC3 Converted Value		Voltage (V)	
1353	4XV ADC4 Converted Value		Voltage (V)	
1354	4XV ADC5 Converted Value		Voltage (V)	
1355	4XV ADC6 Converted Value		Voltage (V)	
1356	4XV ADC7 Converted Value		Voltage (V)	
1366	4XV Autopilot 1 Score		Custom Type (--)	
1367	4XV Autopilot 2 Score		Custom Type (--)	
1368	4XV Autopilot 3 Score		Custom Type (--)	
1369	4XV Autopilot External Score		Numeral System (d...	
1357	4XV Internal ADC8 Converted Value		Voltage (V)	
1358	4XV Internal ADC9 Converted Value		Voltage (V)	
1359	4XV Internal ADC10 Converted Value		Voltage (V)	
1360	4XV Vcc for arbiter		Voltage (V)	

Fig. 143: **Variables section**

To set a custom name for one of the system variables:

1. Click on the **Custom Name** cell of the desired variable and **introduce the new name** for it.
2. When the name is introduced press **Enter** to store the name on the system.
3. Press **Save** to save all changes.

Error: In this version, there is a maximum amount of characters that the user can use to rename variables, i.e. there is no limit per se but there is a maximum size of the configurable (xml size).

Besides changing their name, the user can also configure the measurement units of the variables, as well as the initial value (expressed in SI units) they will have each time the system (re)starts, using the Show Units and the Initial Value (SI) cells.

By right-clicking on the **Show Units** cell, the user can select the desired units.

The table below shows all the available units in **BCS PDI Builder**.

Variable Type	Units
Velocity	[m/s] [kt] [km/h] [mph] [ft/s] [mm/s] [ft/m]
Length	[m] [km] [mi] [NM] [yd] [ft] [in] [cm] [mm]
Angle	rad[-;] °[-180;180] °[0;360] [° ‘ “] [rad] rad[0;2] °
Acceleration	[m/s^2] [ft/s^2] [in/s^2] [g]
Temperature	[K] [°C] [°F]
Magnetic Flux Density	[T] [mG] [gauss] [nT]

continues on next page

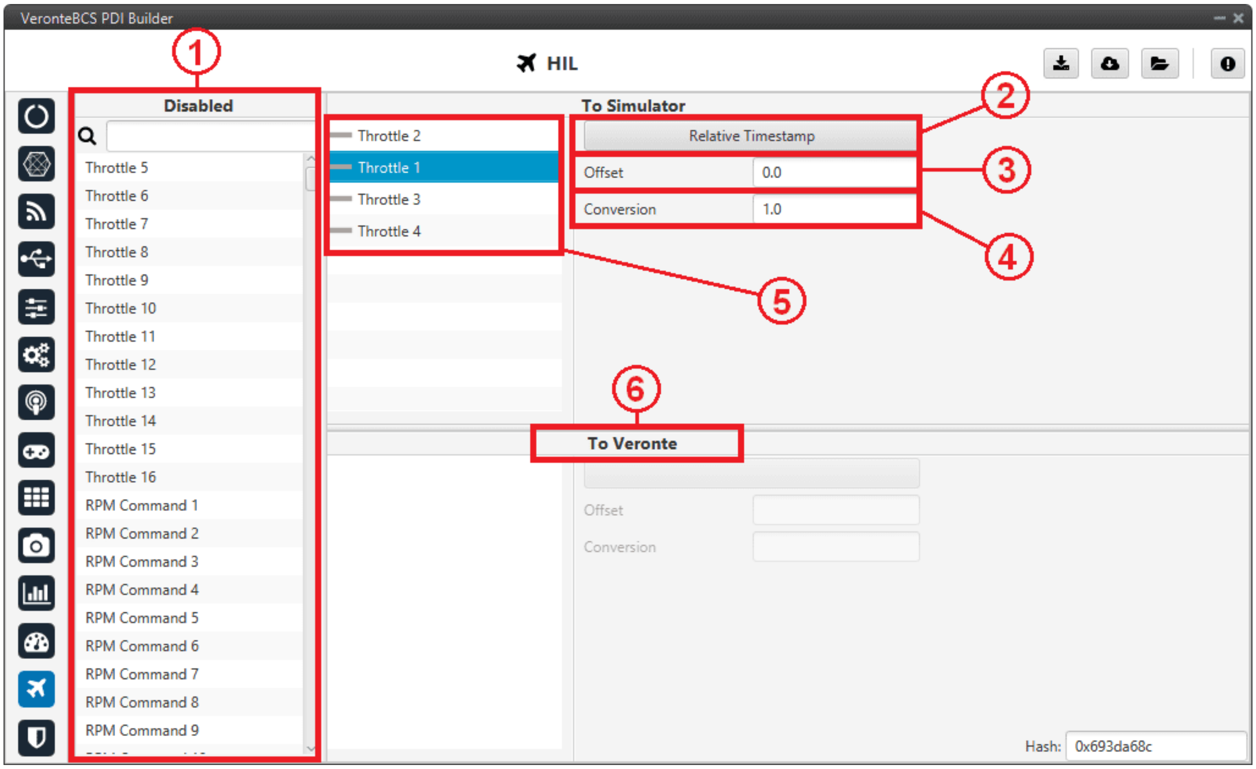
Table 1 – continued from previous page

Variable Type	Units
Voltage	[V] [mV]
Current	[A] [mA]
Pressure	[Pa] [kPa] [bar] [mbar] [psi] [mmHg] [at] [atm]
Time	[s] [min] [h] [s] [ms] [Time]
Angular Velocity	[rad/s] [rad/m] [rad/h] [rps] [rpm] [rph] [°/s]
Flow Rate	[m^3/s] [gal/s] [gal/h] [l/s] [l/h]
Custom Type	[- -]
Percentage	[x1] [%]
Transfer	[pkts/s]
Frequency	[Hz] [mHz] [kHz]
Area	[m^2] [cm^2] [mm^2] [km^2] [mile^2] [ft^2] [yd^2]
Data	[bit] [byte] [kB] [GB] [bytes/s]
Mass	[kg] [g] [tonnes] [lbs] [oz]
Force	[N] [kN] [lbf] [pdl]
Angular Acceleration	[rpm/s] [rad/s^2] [rad/m^2] [rad/h^2] [°/s^2] [°/m^2] [°/h^2]
Baudrate	[Bd] [kBd] [MBd]
Pressure Variance	[Pa^2]
Magfield Variance	[T^2]
Velocity Variance	[(m/s)^2] [(cm/s)^2] [(mm/s)^2]
Numeral System	[bin] [octal] [dec] [hex]
Pressure Square Error Rate	[Pa^2/s]
Centimeters/Pixels	[cm/pixel]
Jerk	[m/s^3]
Power	[W] [kW] [kgm/s] [erg/s] [CV]
Resistence	[]
Inductance	[H]
Volume	[m^3] [dm^3] [mm^3] [L] [mL]
Decibel	[dB]
Density	[kg/m^3]

2.13 HIL

Professional Hardware In the Loop (HIL) Simulator package is a powerful tool for **BCS** and **1x** integration, development and operator training; allowing to extensively operate the system in a safe environment, prior to conducting real flight operations.

The user can link the variables on **BCS** with the corresponding ones in the simulator. In this panel, simulator variables are available on the left side (**Disables**). In addition, it can be seen 2 section more, **To Simulator** and **To Veronte**.



Offset 0.0

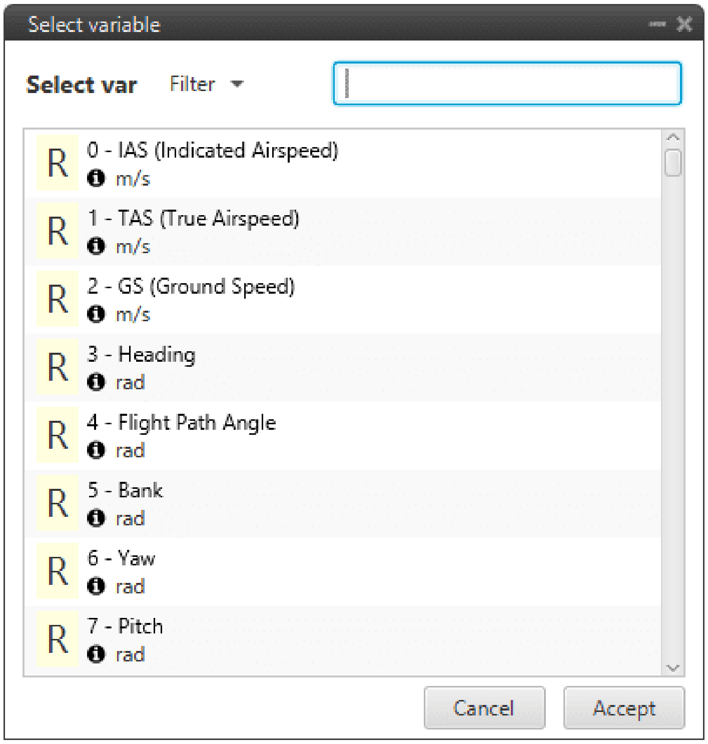
Conversion 1.0

Hash: 0x693da68c

Fig. 144: **HIL menu**

In order to configure the simulation variables, users have to:

1. **Disabled**: Select the simulator variables that have been configured in the aircraft model. Just drag and drop them into **To Simulator** section.
2. Select the actuator variable (Control Output) of **BCS** that matches with the one in the simulator. A new window will be displayed for each variable.



R 0 - IAS (Indicated Airspeed)
m/s

R 1 - TAS (True Airspeed)
m/s

R 2 - GS (Ground Speed)
m/s

R 3 - Heading
rad

R 4 - Flight Path Angle
rad

R 5 - Bank
rad

R 6 - Yaw
rad

R 7 - Pitch
rad

Fig. 145: **BCS variables**

3. **Offset**: Set an offset, if it is necessary.
4. **Conversion Factor**: Set a conversion factor, if it is necessary. It multiplies the **BCS** output signal and can be used in case units on **BCS** and the simulator do not match. For example, in X-Plane simulator, the unit of angles is radians.

   Note: To be sure of which units the simulator has, please refer to the relevant simulator manual.

   Warning: Always make sure that surfaces are moving in the right direction and with the correct deflection angle.

5. Here users can see all the variables selected and sent to the simulator. Select the one to be configured.

   To remove a variable from the list, simply click on the “**-**” icon next to it.

6. **To Veronte**: The user can also select variables to be sent from the simulator to **BCS**. An interesting variable could be the RPM of the motor.

2.14 Safety

In this menu the user can create checklists for each phase, avoid changing certain parameters, settings or programs and define safety bits lists.

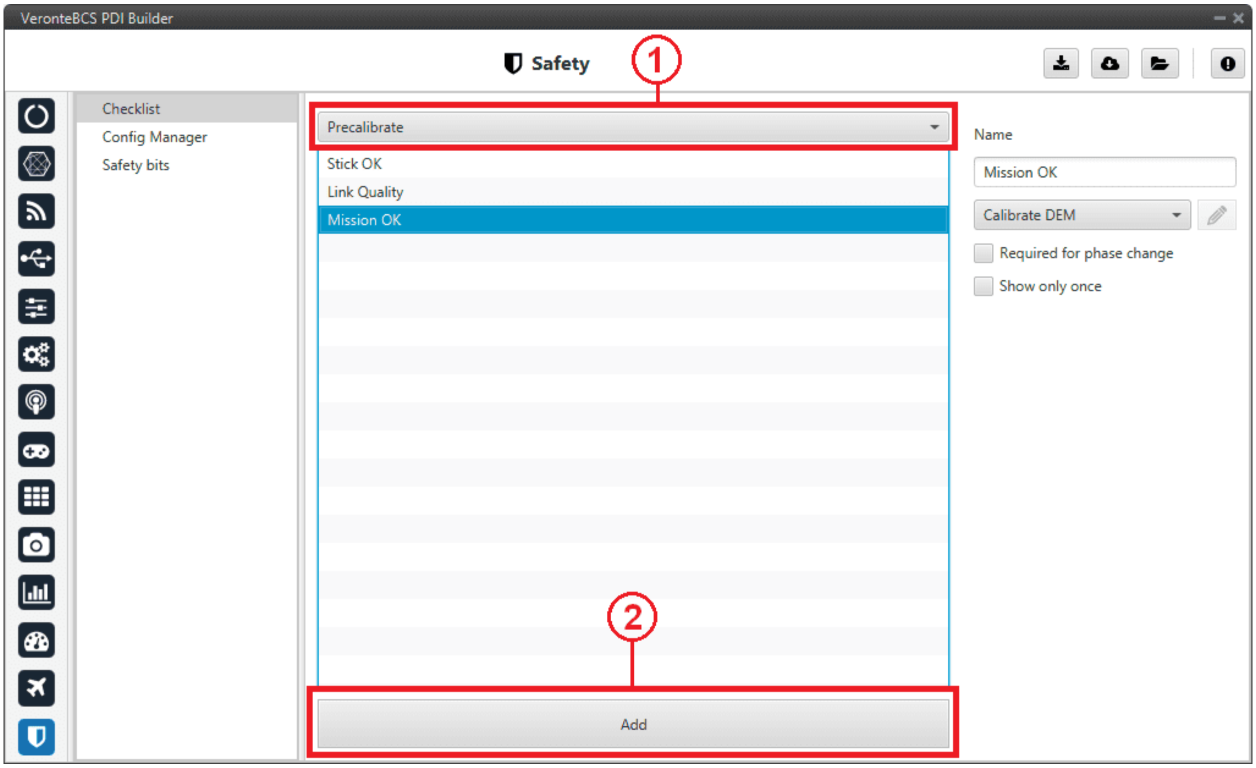
2.14.1 Checklist

This feature is used to make sure that some requirements have been accomplished, for example, prior to a phase change or to avoid a possible malfunction.

These checklists will appear in a panel called **Checklist** of **Veronte Ops** (for more information about this, visit Veronte Ops manual).

Note: There are 3 different types of checks:

- Checks that are performed automatically by **BCS**, such as "**In Range check**".
- Checks that need a command to **BCS**, e.g. "**Calibrate Atmosphere**".
- Checks for operator information only, which are performed with type "**None**".



Precalibrate

Stick OK
Link Quality
Mission OK

Name: Mission OK
Calibrate DEM
□ Required for phase change
□ Show only once

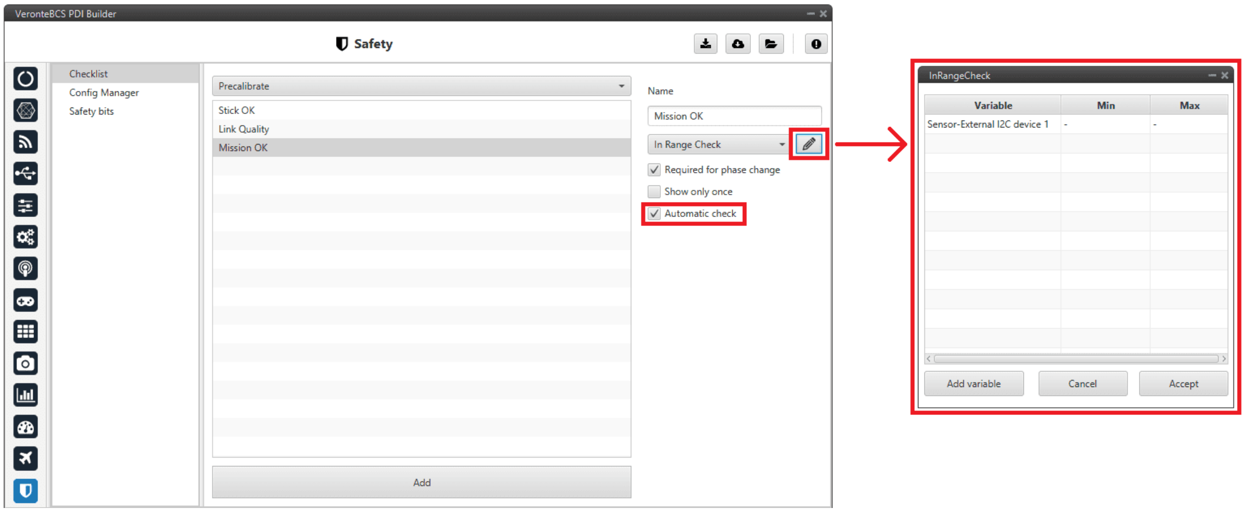
Fig. 146: **Checklist section**

In (1), the user will find all the phases configured for the operation. In each one of them, new elements for the checklist can be added with the button **Add** (2). The user can modify the checklist order of the phase by selecting and dragging elements in the list to the desired position.

The configurable parameters for each element are:

- **Name**: The name that will identify the element.
- **Type**: The element chosen from the checklist can be one of the following types:
  - **Calibrate Atmosphere**: The user can request the calibration of the atmosphere model.
  - **Calibrate DEM**: The user can request the calibration of the DEM.
  - **Command Position**: Send to the UAV a position.
  - **Command Yaw**: Send to the UAV a yaw angle.
  - **Enter Wind Information**: Enter initial values for wind state to the UAV.
  - **In Range Check**: Allows checking if a variable is between the range selected.
  - **None**: Any action is performed, been just a check for the user to do something external.
  - **Trim arcade**: The user can request the stick calibration for arcade commands.
- **Required for phase change**: If **enabled**, the element must be checked to switch to another phase.
- **Show only once**: If **enabled**, the check will only appear the first time its phase is executed.
- **Automatic check**: This option is only available when **'In Range check'** is selected.

An example of '**In Range Check**' can be shown below:



Name: Mission OK

In Range Check

☑ Required for phase change

□ Show only once

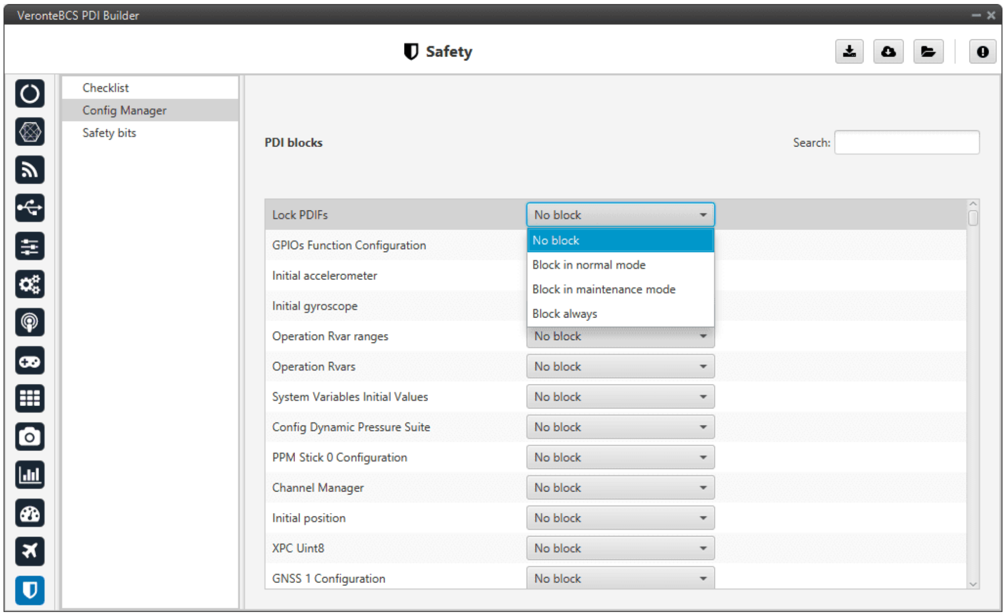
☑ Automatic check

Variable	Min	Max
Sensor-External I2C device 1	-	-

Fig. 147: **Checklist example**

2.14.2 Config Manager

Config manager prevents the user changing certains parameters, settings or programs of **BCS**. It is shown in the picture below:



Lock PDIFs	No block
GPIOs Function Configuration	
Initial accelerometer	
Initial gyroscope	
Operation Rvar ranges	No block
Operation Rvars	No block
System Variables Initial Values	No block
Config Dynamic Pressure Suite	No block
PPM Stick 0 Configuration	No block
Channel Manager	No block
Initial position	No block
XPC Uint8	No block
GNSS 1 Configuration	No block

Fig. 148: **Config manager section**

The user can choose between:

- No block
- Block in normal mode
- Block in maintenance mode
- Block always

2.14.3 Safety bits

In this section the user can configure 3 different safety bits lists.

The bits included in these lists are added to the set of default system bits that trigger the **System error** variable, and therefore trigger the **FTS**. The user can refer to this list of default system bits in the Activation System Error bits section of the **1x Software Manual**.

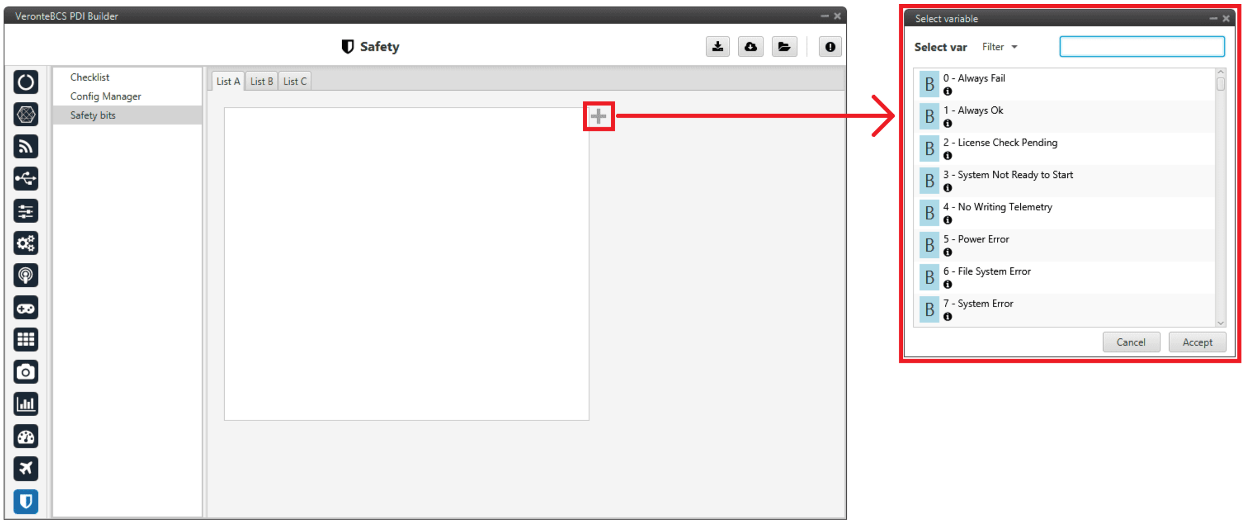


Fig. 149: **Safety bits section**

By default, there is no bit defined in any Safety bits list. To add them, just press "**+**" icon and select the desired bits. A common user bit to add to these lists is the '**Sensors error' bit**, so that if one of the sensors fails, the FTS is triggered.

In addition, the user can switch between the different lists with an action, see *Actions* of the Automations menu.

Once the installation is finished, open **BCS PDI Builder** and select the unit.

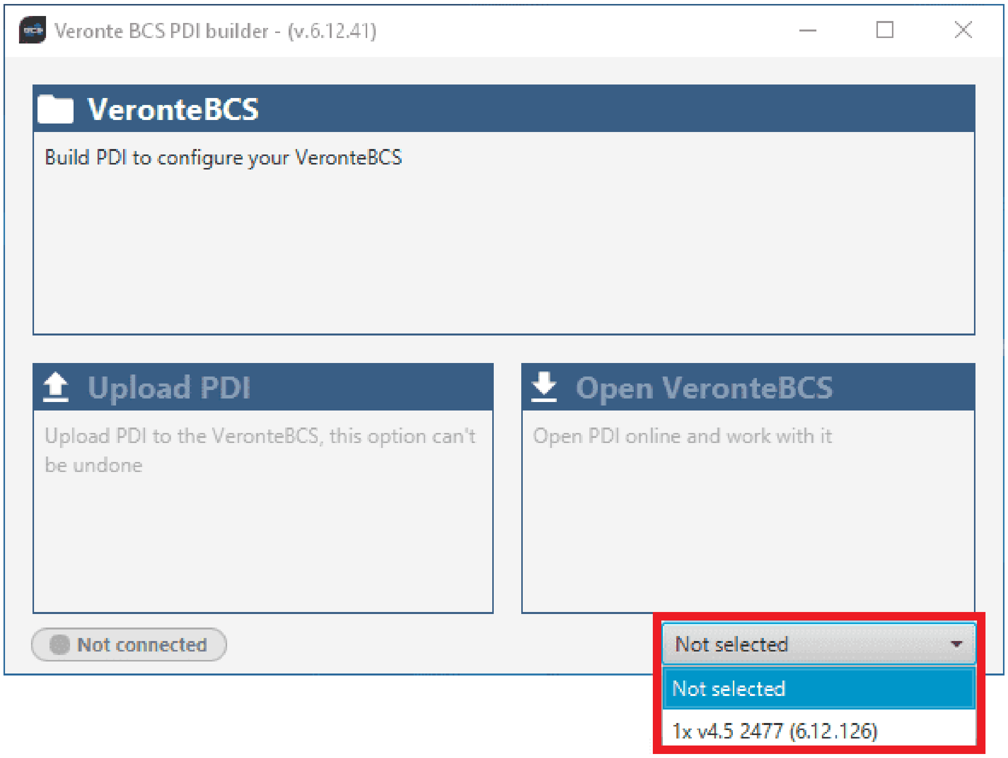
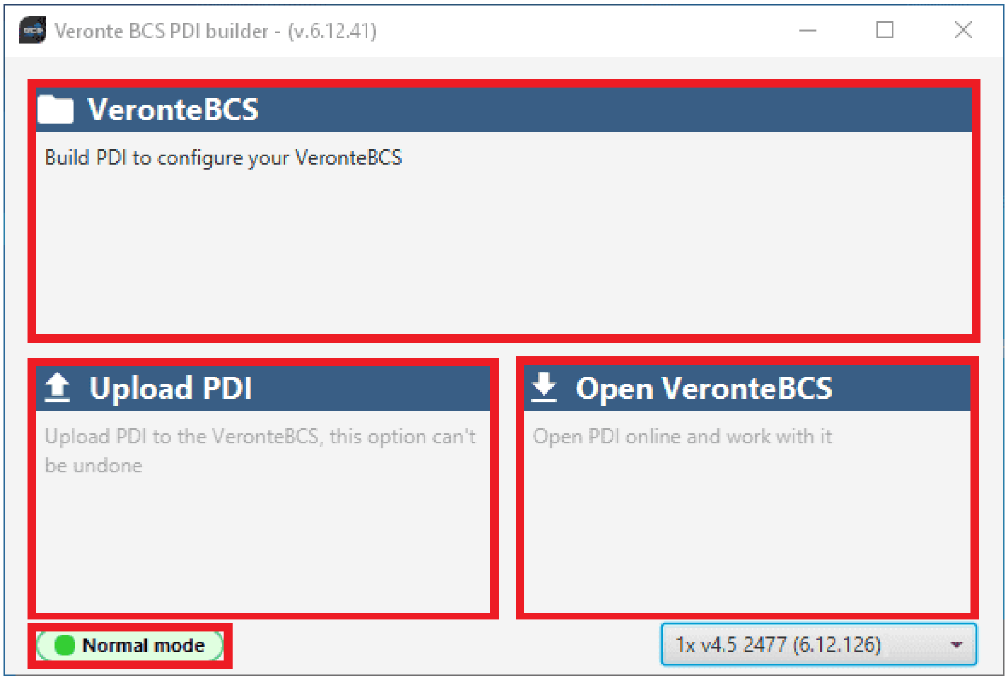

Veronte BCS PDI builder - (v.6.12.41)
VeronteBCS
Build PDI to configure your VeronteBCS
Upload PDI
Upload PDI to the VeronteBCS, this option can't be undone
Open VeronteBCS
Open PDI online and work with it
Not connected
Not selected
Not selected
1x v4.5 2477 (6.12.126)

Fig. 150: **BCS selection**

If it is correctly connected, it should appear in **Normal mode**, as shown in the figure below, or **Maintenance mode**.



Build PDI to configure your VeronteBCS

Upload PDI to the VeronteBCS, this option can't be undone

Open PDI online and work with it

Fig. 151: **BCS PDI Builder - Normal mode**

BCS unit can also appears as: Maintenance mode (loaded with errors) or Normal mode - Disconnected.

Note: Maintenance mode (loaded with errors) appears when something is wrong in the configuration. For more information, see *Troubleshooting section* of this manual.

The user can access now to 3 configuration options:

- **Veronte BCS**: It allows the user to work with **offline** configurations. A previously exported **BCS** PDI can be opened and modified or it is possible to build a new one from the default configuration.
- **Upload PDI**: A previously exported **BCS** PDI configuration can be imported to the linked **BCS**.
- **Open Veronte BCS**: By clicking on this option, **BCS PDI Builder** configuration menu opens with the configuration (the PDI files) loaded in the connected **BCS**. Then, the user can modify it online.

Note: PDI files are configuration files. These files allow for modular control with improved version management. These PDI files are split in two folders. Each folder hold several .xml files:

- **Operation**: This folder holds all the files related with the operations defined, such as waypoints, routes, operative parameters, runaways, etc.
- **Setup**: This contains the configuration of the vehicle. All the control loops and their parameters, the definition of the flight phases and guidance commands, and the automations defined are stored here.

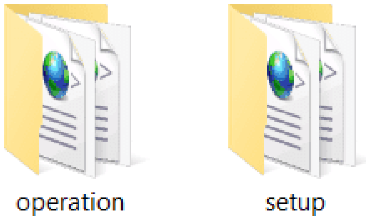


Fig. 152: **PDI files**

Finally, click on 'Open Veronte BCS' to open the configuration and start editing. The different 'buttons' that can be seen in the initial menu of the **BCS PDI builder** are explained below.

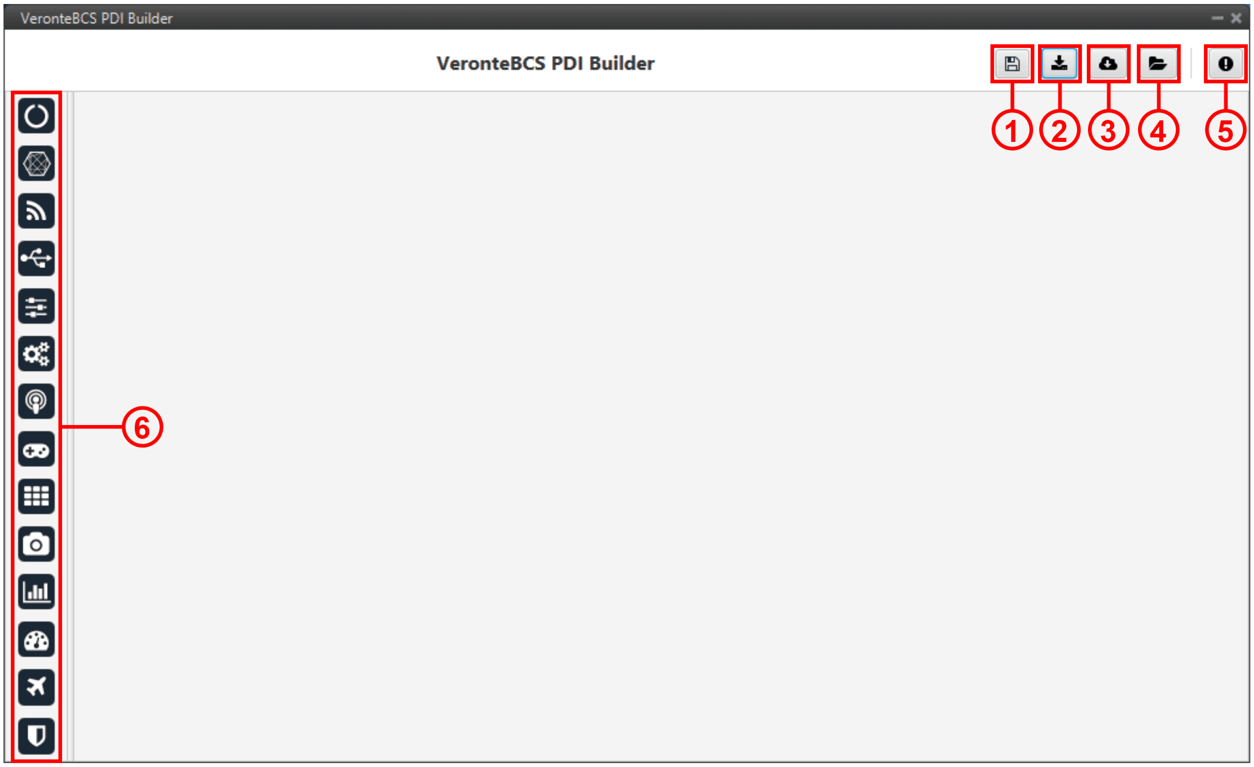


Fig. 153: **Initial menu**

1. **Save PDI**: After changes are done, press on the save button to apply the changes.

   While saving, a percentage of the progress of the saving process is displayed:

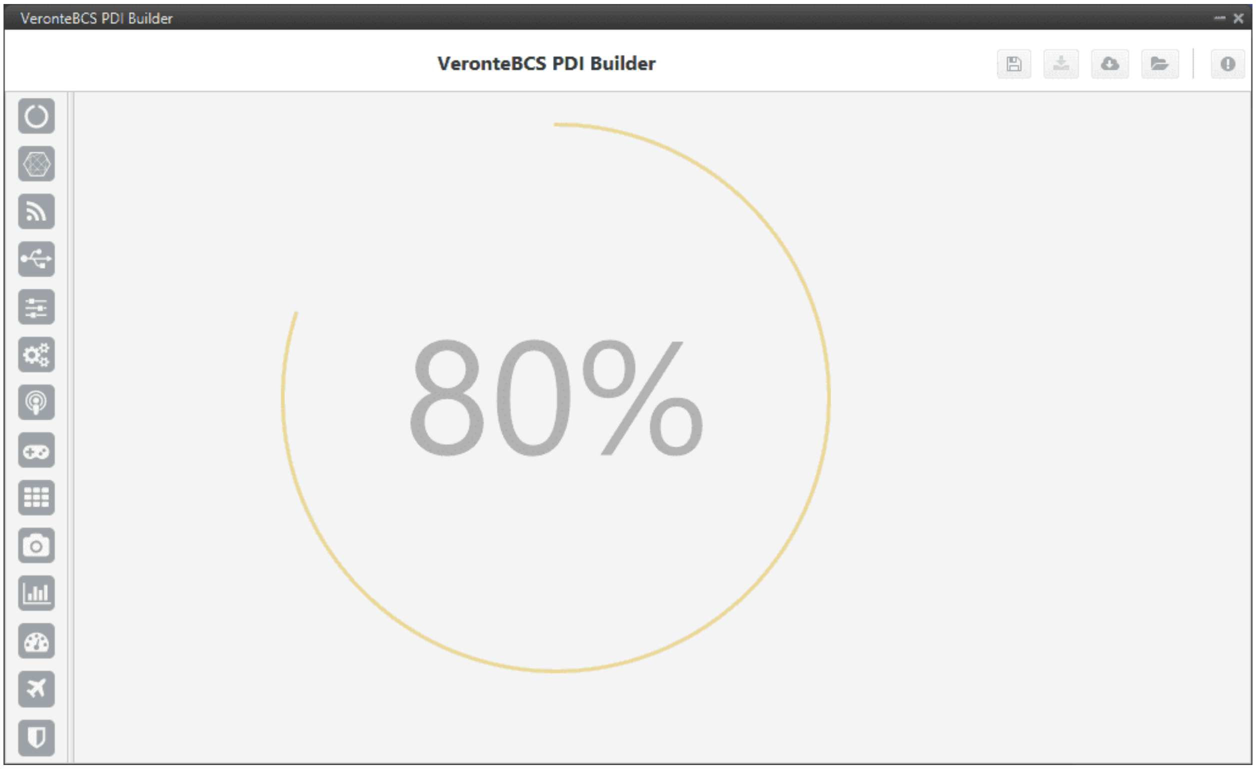


Fig. 154: **Save PDI**

After saving any changes, **BCS** will **RESET** and the **BCS PDI Builder** software will **close**.

Danger: As **Veronte BCS** is **reset**, it is **not advisable to save changes during flight tests**.

Note: This button will only appear if a **BCS** is connected, i.e. when working offline this button will not be available.

2. **Export PDI**: After modifying a configuration, press the export button to store the configuration in the local storage. Users can store this configuration in an empty folder or in the folder where the previously imported configuration is stored. With the latter option, the "original" configuration will be overwritten by the one with the new changes. The user can choose between:

   - **Download PDI**: With this option the 2 folders with the PDI files are downloaded.
   - **Download VER file**: Download a **.ver file** with the configuration in **binary**. This option is only available 'online', otherwise it will be disabled.

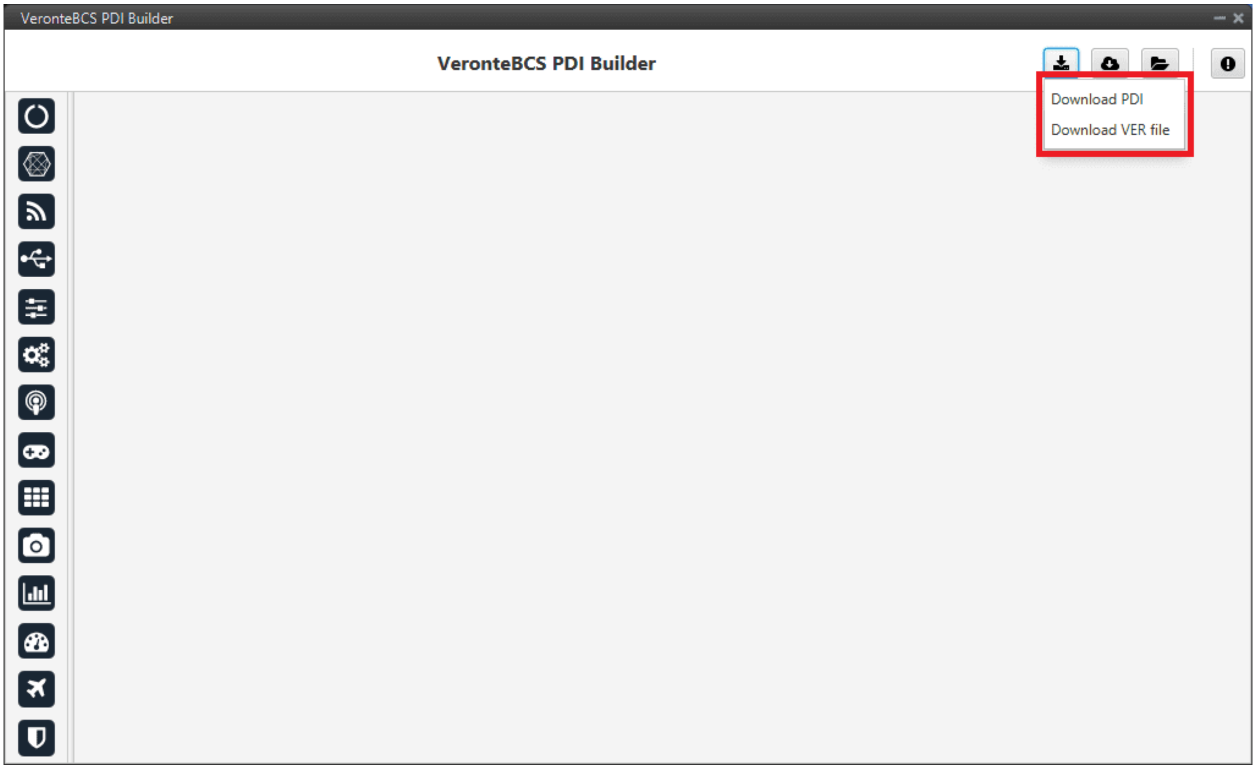


Fig. 155: **Download option**

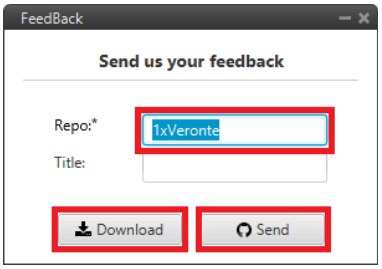
3. **Import PDI from repo**: The user can import a configuration file from the repo and modify it. After that, if the save button is pressed, this configuration will be uploaded on the **BCS**.
4. **Import PDI from local storage**: The user can import a configuration file from the local storage and modify it. After that, if the save button is pressed, this configuration will be loaded into the **BCS**.

   > **Warning:** If users want to upload a configuration in this way (either from the repo or from local storage), note that only the configuration 'setup' folder is uploaded, the 'operation' folder is not.

   This means that the operation of the new configuration uploaded to the **BCS** will not be saved, but the previous operation will remain.

5. **Feedback**: Users can report a problem they have encountered by **creating an issue in their own 'Joint Collaboration Framework'**. The 'Download' button downloads a zipped folder with the current **BCS** configuration and more information needed for Embention to resolve the issue. It is advisable to attach this folder when creating the issue.

   > **Note:** The user's '**Joint Collaboration Framework**' is simply a **own Github repository for each customer**.
   >
   > If the user has any questions about this Joint Collaboration Framework, please see Joint Collaboration Framework user manual or contact sales@embention.com.



Repo:* 1xVeronte

Title:

Fig. 156: **Feedback**

6. These are the different functions of **BCS** control station. Each option will be explained in detail in the next sections.

Icon	Item	Description
○	*Veronte*	Introduce BCS information
⬡	*Connections*	Configure I/O connections on BCS
⌇	*Sensors*	Configure parameter sensors
⭠	*Input/Output*	Configure external sensors/devices and I/O signals
☰	*Control*	Introduce Phases, Modes and Arcade axis configuration
⚙	*Automations*	Configure automatic actions on event detection (go home, change phase…)
◉	*Communications*	Configure alternative communication channels, statistics and routing
🎮	*Stick*	Cusomize transmitter configuration
▦	*Block Programs*	Customize algorithms executed by BCS
📷	*Devices*	Configure any connected devices: servo, radio, camera…
📊	*Telemetry*	Customize traffic: log, telemetry…
◔	*UI*	Customize variable names
✈	*HIL*	Configure parameters for hardware simulation
🛡	*Safety*	Customize checklist, block user control in PDI configuration and safety bits

CHAPTER

THREE

INTEGRATION EXAMPLES

In this section, a series of examples will be presented so that the user knows how to configure certain customizations in the **BCS PDI Builder**. In addition, some examples of integration between **BCS** and external devices are explained.

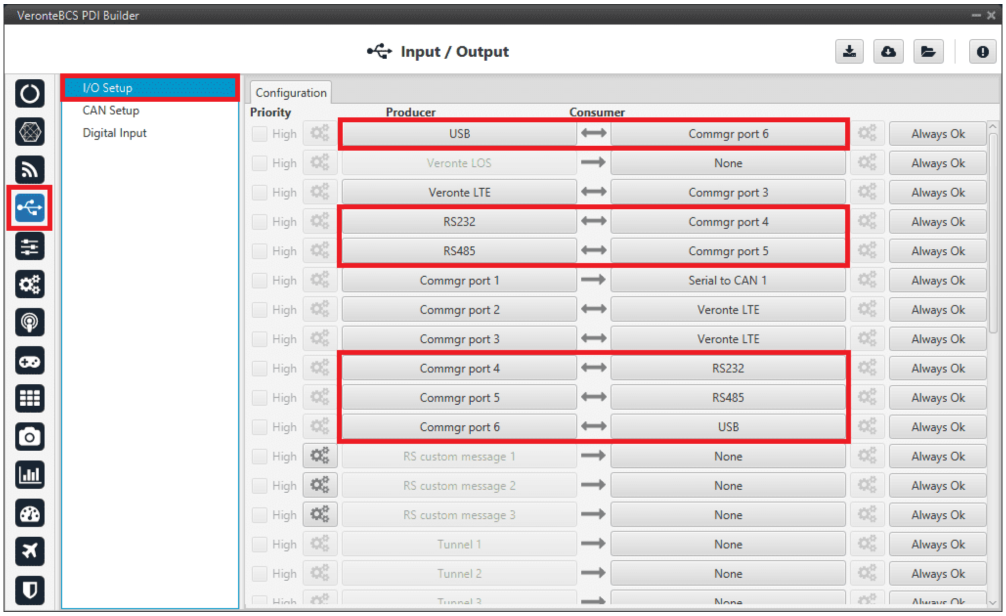
3.1 Communication with PC

Since **BCS** can be connected to a computer via a USB or serial interface, the configuration for both connections is already set by default in **BCS PDI Builder**.

However, users should check that this configuration has not been modified to ensure a correct communication via both ways in case one of them is lost. To verify this:

Go to Input/Output menu → **I/O Setup section**. **Each USB, RS232 and RS485 Producers must be bidirectionally connected to a Commgr port**:

Important: Users should also check that Commgr ports to which USB and serial ports are connected are not routed. For more information on Routing, see *Ports section* of this manual.



Priority	Producer		Consumer	
High	USB	⟷	Commgr port 6	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	⟷	Commgr port 3	Always Ok
High	RS232	⟷	Commgr port 4	Always Ok
High	RS485	⟷	Commgr port 5	Always Ok
High	Commgr port 1	→	Serial to CAN 1	Always Ok
High	Commgr port 2	⟷	Veronte LTE	Always Ok
High	Commgr port 3	⟷	Veronte LTE	Always Ok
High	Commgr port 4	⟷	RS232	Always Ok
High	Commgr port 5	⟷	RS485	Always Ok
High	Commgr port 6	⟷	USB	Always Ok
High	RS custom message 1	→	None	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Fig. 1: **USB/RS232/RS485 ↔ Commgr port**

3.2 ArcTrim Button

The ArcTrim button allows the user to trim the stick signal directly from the stick position, before the operation, by simply clicking on it. In addition, this button is considered as an 'action button' that can be embedded in the **Veronte Panel**.

To do this, read the following steps:

1. Go to the **Block Programs menu**.
   - Create a program to make the necessary connection to the *Arc Trim* block.

     Usually the user has a **Stick program** where the blocks that are related to the stick are implemented.
   - Add the *Arc Trim* block and connect the input and output variables to it.

     Usually the **input** variables are **Stick Input u1-u4** and the **output** variables **Stick Input d1-d4**.
   - Finally, **enable the block to be commanded** by simply clicking on .

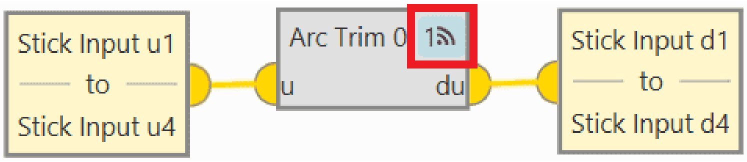
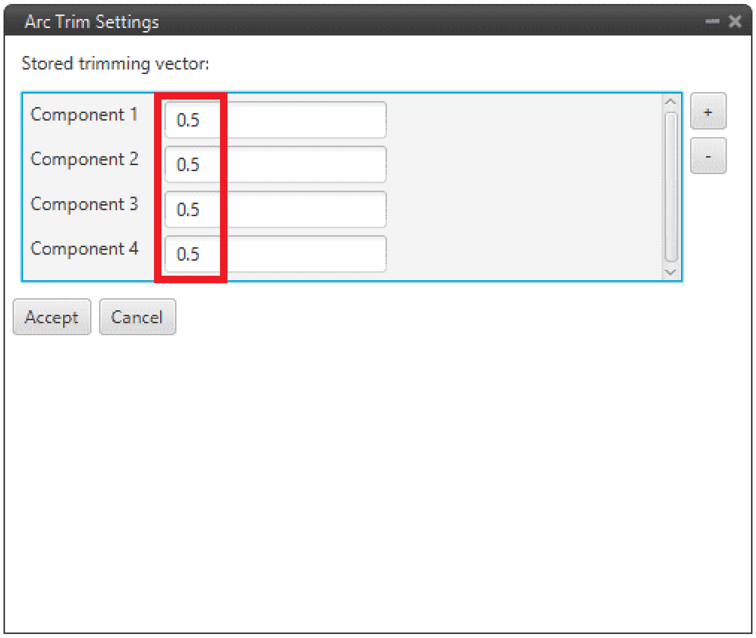


Fig. 2: **Arc Trim block**

2. **Configure the trim vector** of the *Arc Trim block*.

   Depending on the range of the signal, the following values are recommended:
   - If the signal ranges from **0 to 1** ⇒ **0.5**.
   - If the signal ranges from **-1 to 1** ⇒ **0**.

   In this example, the signal is in the range from 0 to 1, then 0.5 is set:



Arc Trim Settings

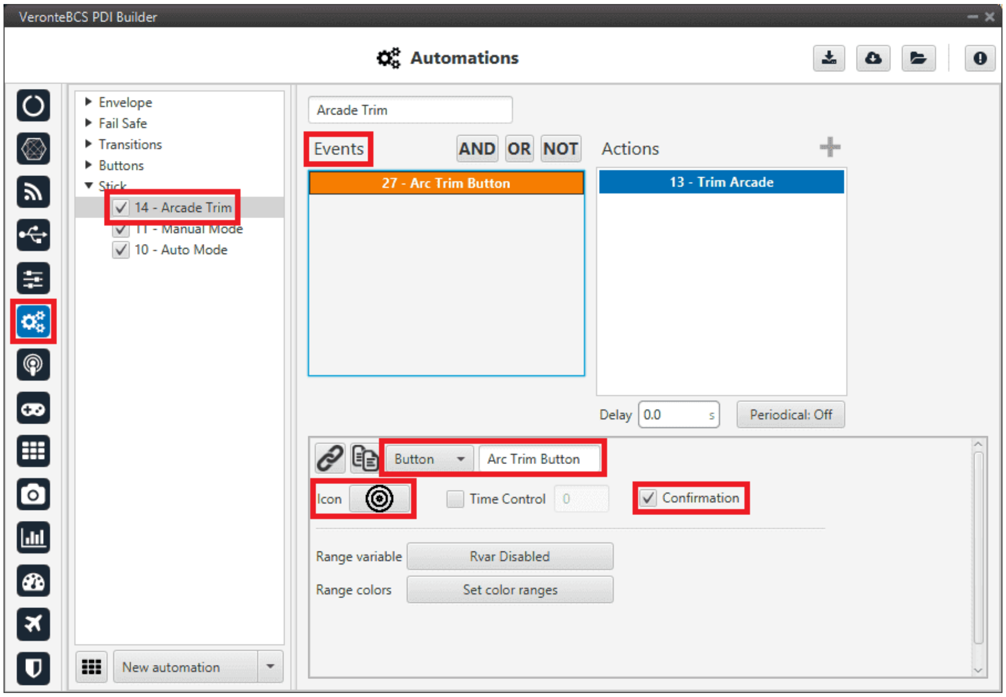
Stored trimming vector:

Component 1: 0.5
Component 2: 0.5
Component 3: 0.5
Component 4: 0.5

Fig. 3: **Arc Trim block configuration**

3. Go to **Automations menu** → create a **New Automation** → **Events**.

   Select the **Button** option and choose the desired icon for this button. In addition, it is recommended to activate the **Confirmation** checkbox, to prevent trimming the stick by mistake.



VeronteBCS PDI Builder

Automations

- Envelope
- Fail Safe
- Transitions
- Buttons
- Stick
  - ☑ 14 - Arcade Trim
  - ☑ 11 - Manual Mode
  - ☑ 10 - Auto Mode

Arcade Trim

Events: 27 - Arc Trim Button

Actions: 13 - Trim Arcade

Delay 0.0 s
Periodical: Off

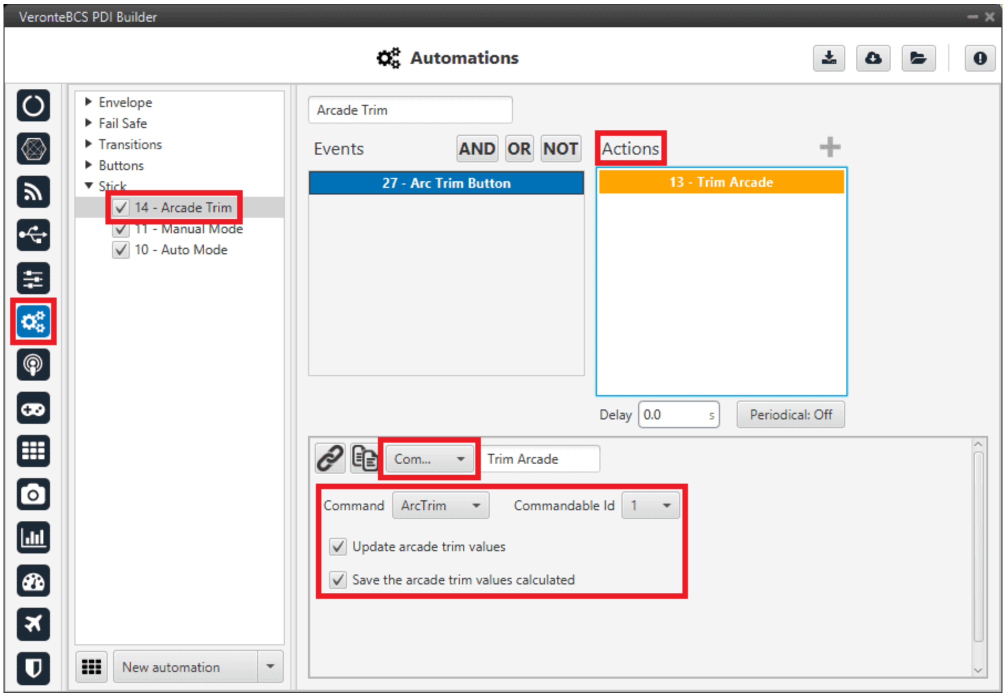
Button: Arc Trim Button
Icon
□ Time Control 0
☑ Confirmation

Range variable: Rvar Disabled
Range colors: Set color ranges

Fig. 4: **Arcade trim automation - Events**

4. In the created automation, go to **Actions**.

- Add the **Command block** action.
- Select **Arc Trim** block to command and choose the commandable **Id**.
- Finally, it is recommended to activate both checkboxes:



- ☑ 14 - Arcade Trim
- ☑ 11 - Manual Mode
- ☑ 10 - Auto Mode

Arcade Trim

Delay 0.0 s

Com... Trim Arcade

Command ArcTrim Commandable Id 1

☑ Update arcade trim values

☑ Save the arcade trim values calculated

Fig. 5: **arctrim_automation_action.png**

5. In **Veronte Ops**, this button will appear embedded in the **Veronte Panel**.

Note: This action button will only appear on the Veronte Panel if the action buttons have been enabled to be shown on it. For more information on this, see Veronte Panel section on the **Veronte Ops** user manual.

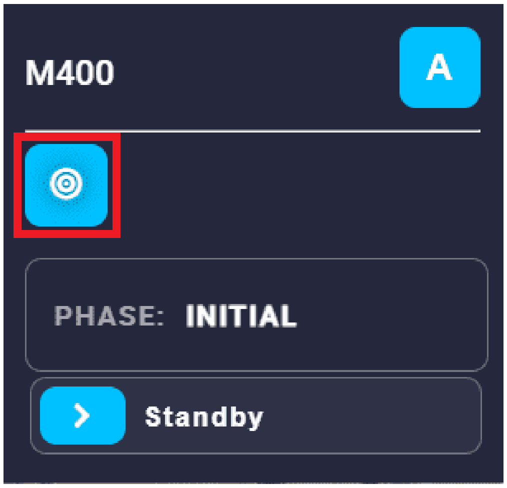
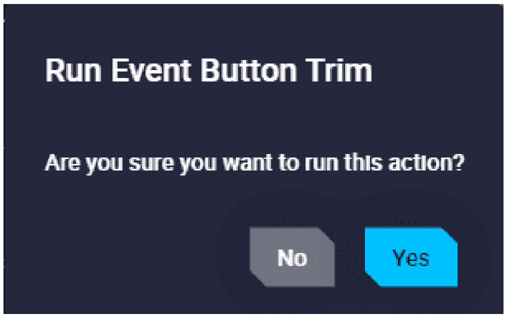


Fig. 6: **Arc Trim button - Veronte Panel**

When clicking on it, the following confirmation message will be displayed (as the confirmation checkbox has been activated in the automation):



Run Event Button Trim

Are you sure you want to run this action?

Fig. 7: **Arc Trim button - Veronte Panel**

Now, the stick is trimmed.

3.3 CAN communication

This section describes how to correctly receive and transmit CAN messages:

CAN messages reception

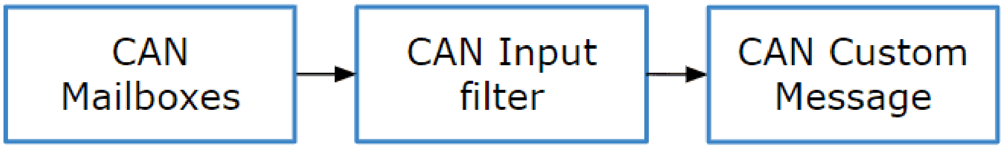
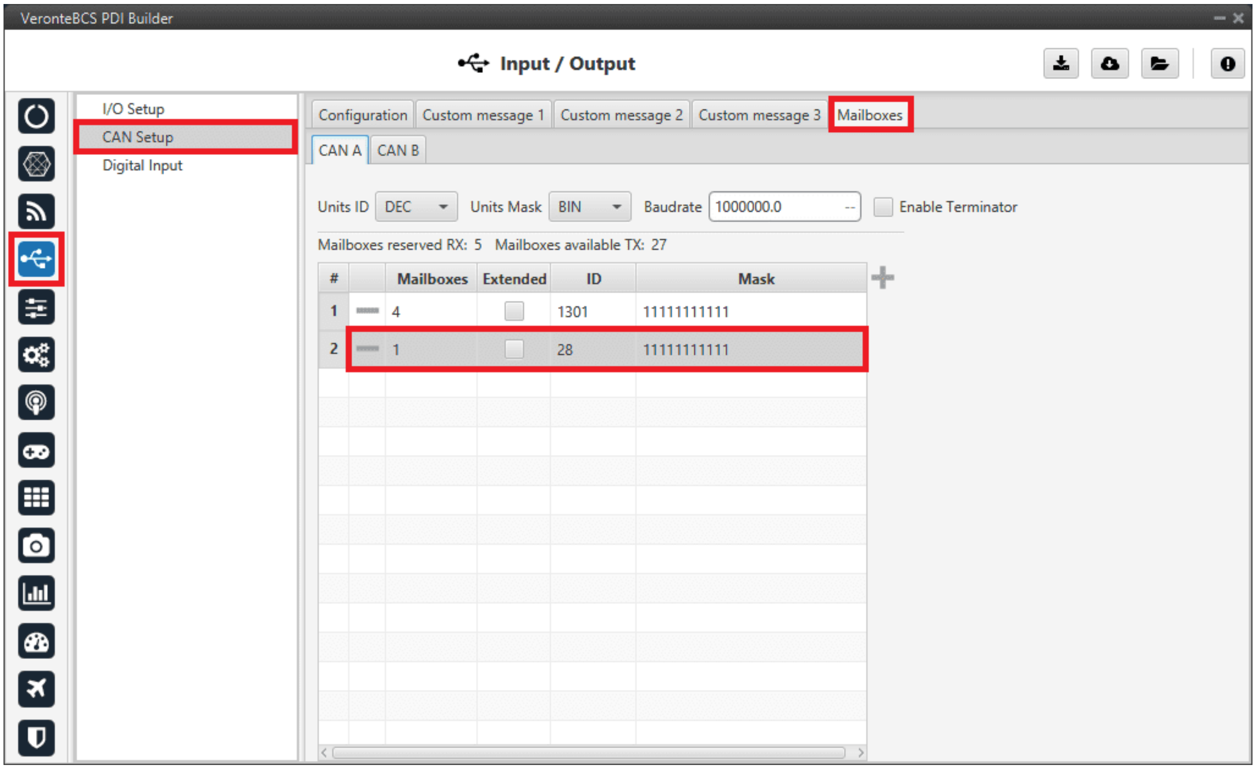


Fig. 8: **CAN messages reception diagram**

1. Go to Input/Output menu → CAN Setup section → **Mailboxes tab**.

   Configure the mailbox to receive a message with the appropiate **ID** (in this example ID 28 has been configured):



Units ID DEC Units Mask BIN Baudrate 1000000.0 -- □ Enable Terminator

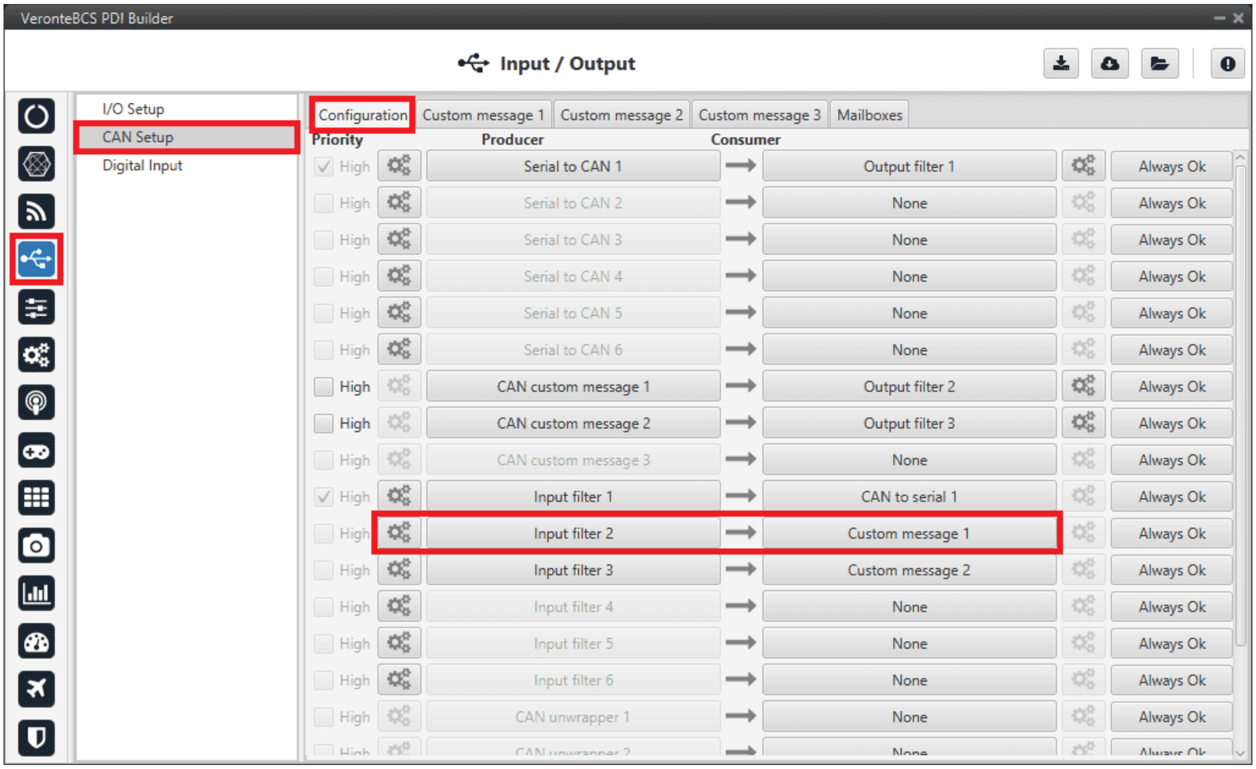
Mailboxes reserved RX: 5 Mailboxes available TX: 27

#		Mailboxes	Extended	ID	Mask
1	—	4	□	1301	11111111111
2	—	1	□	28	11111111111

Fig. 9: **Mailbox configuration**

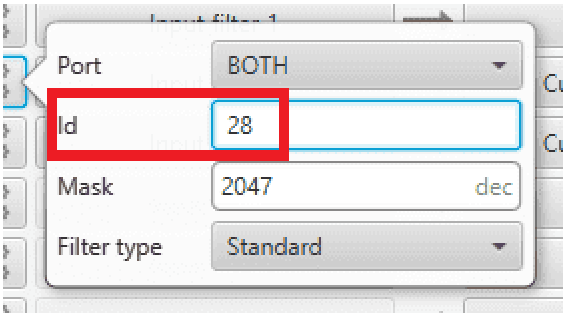
2. Go to Input/Output menu → CAN Setup section → **Configuration tab**.

   Connect an **Input filter** with the **right CAN ID** to a **Custom message consumer**:



Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
☐ High	Serial to CAN 2	→	None	Always Ok
☐ High	Serial to CAN 3	→	None	Always Ok
☐ High	Serial to CAN 4	→	None	Always Ok
☐ High	Serial to CAN 5	→	None	Always Ok
☐ High	Serial to CAN 6	→	None	Always Ok
☐ High	CAN custom message 1	→	Output filter 2	Always Ok
☐ High	CAN custom message 2	→	Output filter 3	Always Ok
☐ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
☐ High	Input filter 2	→	Custom message 1	Always Ok
☐ High	Input filter 3	→	Custom message 2	Always Ok
☐ High	Input filter 4	→	None	Always Ok
☐ High	Input filter 5	→	None	Always Ok
☐ High	Input filter 6	→	None	Always Ok
☐ High	CAN unwrapper 1	→	None	Always Ok
☐ High	CAN unwrapper 2	→	None	Always Ok

Fig. 10: **Input filter**



Port: BOTH

Id: 28

Mask: 2047 dec

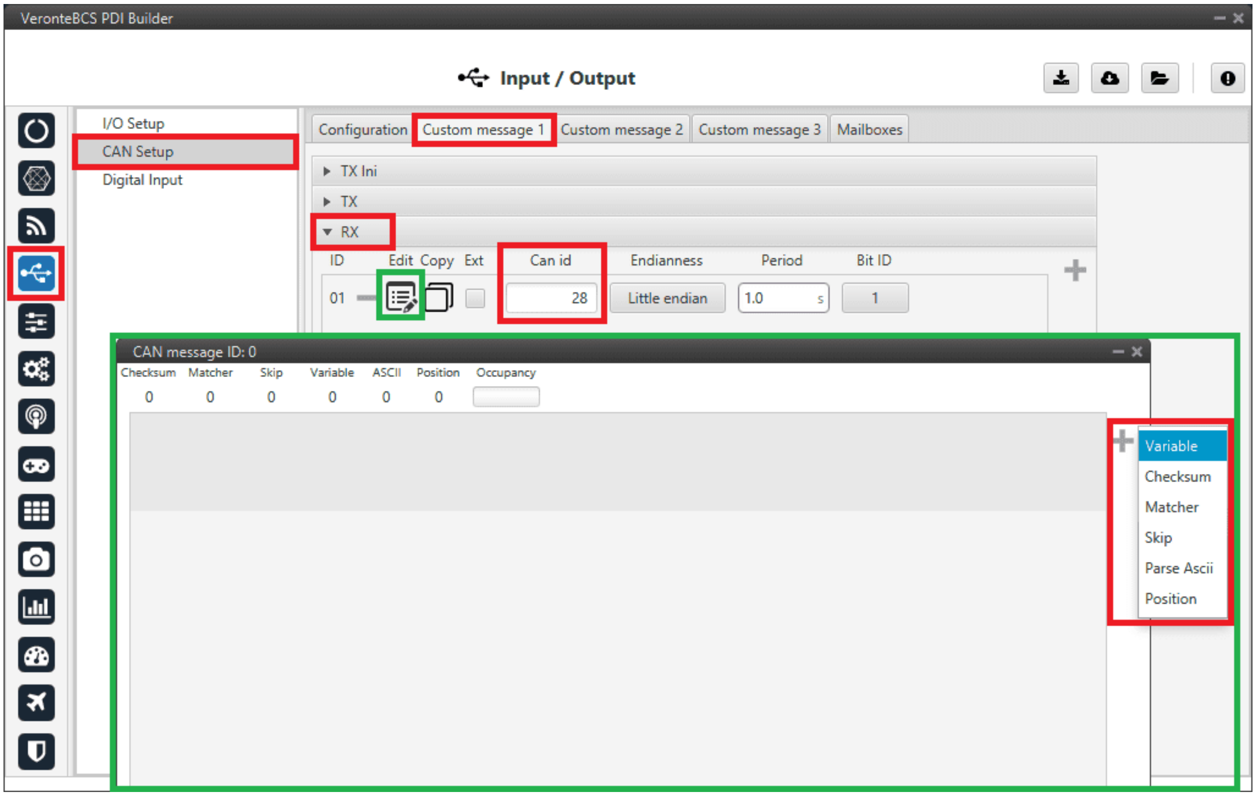
Filter type: Standard

Fig. 11: **Input filter configuration**

3. Go to Input/Output menu → CAN Setup section → **Custom message 1 tab** (as Custom Message **1** has been selected as consumer).

   Configure the message reading as desired in the **RX tab** by setting the correct CAN ID.

   The different options and parameters to be configured are explained in the *RX messages -> Custom messages section* of this manual.



ID	Edit	Copy	Ext	Can id	Endianness	Period	Bit ID
01			□	28	Little endian	1.0 s	1

CAN message ID: 0

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	0	0	0	0	0	

Fig. 12: **Custom Message configuration**

CAN messages transmission

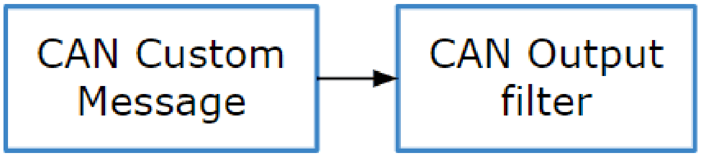


Fig. 13: **CAN messages transmission diagram**

1. Go to Input/Output menu → CAN Setup section → **Custom message 1 tab**.

   Select the fields to send in the **TX** or TX Ini section, as it is a Producer. More information on the configuration of CAN messages can be found in the *TX messages -> Custom messages section* of this manual.

   For example, a CAN messsage set to ID 12:

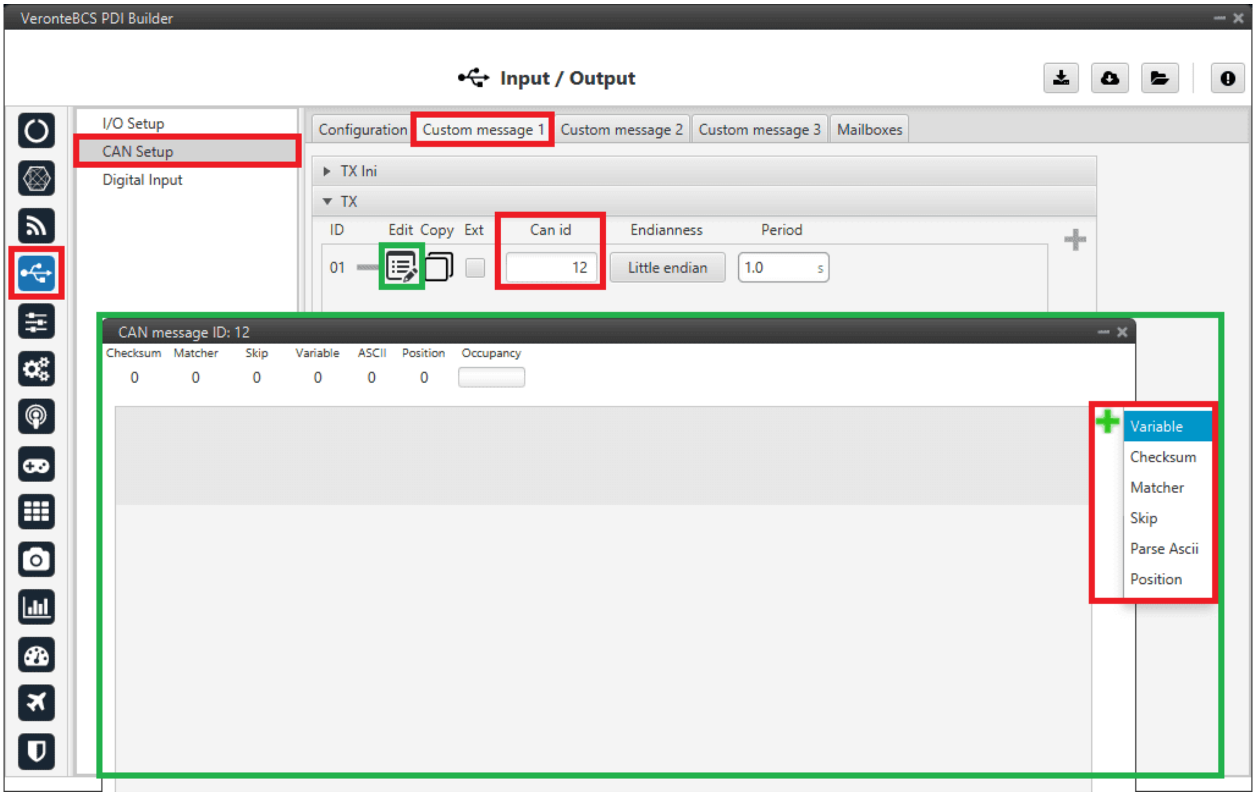
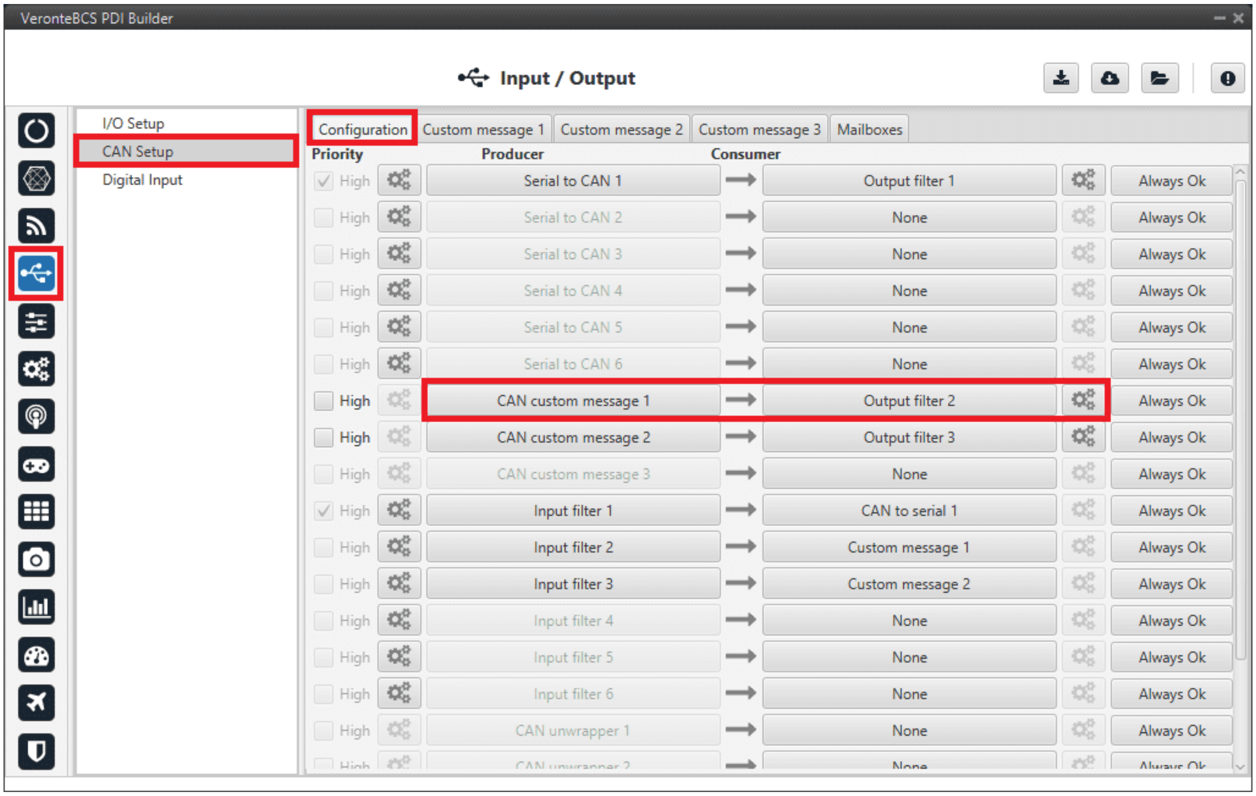


Fig. 14: **Custom Message configuration**

2. Go to Input/Output menu → CAN Setup section → **Configuration tab**.

   Connect **CAN custom message 1** producer (as the message has been configured in the Custom Message **1** tab) to an **Output Filter** as follows:



Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
□ High	Serial to CAN 2	→	None	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message 1	→	Output filter 2	Always Ok
□ High	CAN custom message 2	→	Output filter 3	Always Ok
□ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
□ High	Input filter 2	→	Custom message 1	Always Ok
□ High	Input filter 3	→	Custom message 2	Always Ok
□ High	Input filter 4	→	None	Always Ok
□ High	Input filter 5	→	None	Always Ok
□ High	Input filter 6	→	None	Always Ok
□ High	CAN unwrapper 1	→	None	Always Ok
□ High	CAN unwrapper 2	→	None	Always Ok

Fig. 15: **Output filter**

Warning: Remember that it is necessary to have **at least 1 free mailbox for TX messages**.

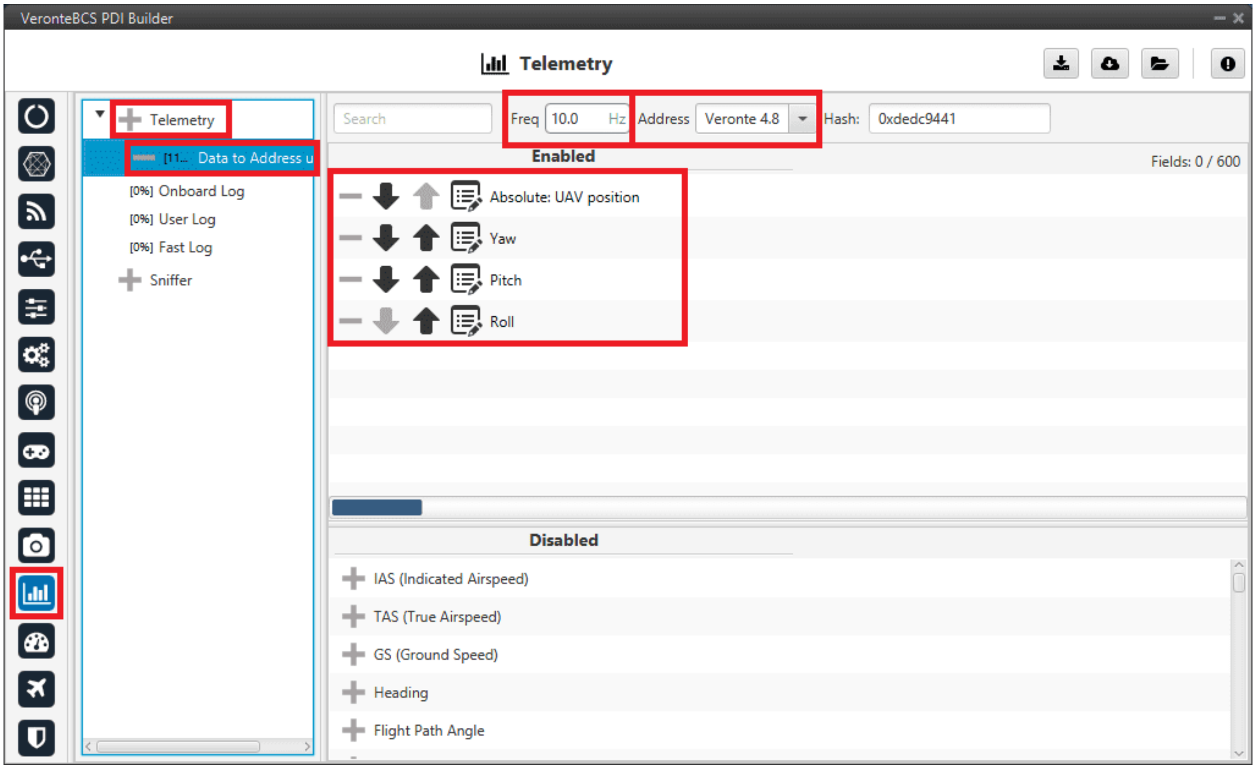
3.4 Data transmission between BCS and 1x Autopilot

To stablish proper communication between ground and air units, the **telemetry** and **sniffer** menus must be configured, respectively.

A simple example of use between a ground unit (**BCS**) and an air unit (**1x**) is shown below:

In the **BCS**:

1. Go to Telemetry menu → **Telemetry** section → **Data link to VApp** tab (see *Telemetry section*, for more information about this).
2. Add the variables: *Absolute: UAV position*, *Yaw*, *Pitch* and *Roll*.
3. Set a **Frequency**, it is recommended to set it to **10 Hz**.
4. On **Address**, point to the **1x air unit** (it is needed to have both units connected through the radio in order to be able to see them on the menu).



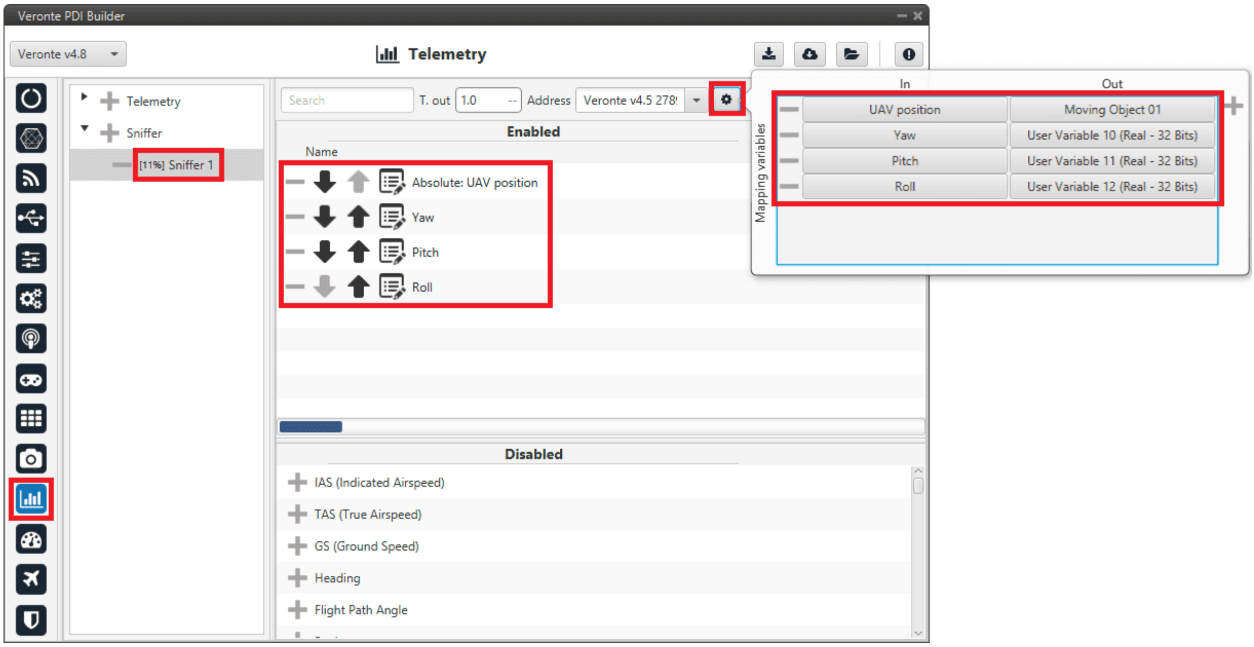
Freq 10.0 Hz Address Veronte 4.8 Hash: 0xdedc9441

Enabled: Absolute: UAV position, Yaw, Pitch, Roll

Fig. 16: **BCS ground unit - Telemetry**

For the **1x air unit** use **1x PDI Builder**:

1. Go to Telemetry menu → **Sniffer** section (for more information about this, see *Sniffer section*).
2. Add a new Sniffer.
3. Configure the **same variables** (keeping the **same order**) than in the ground unit.
4. On **Address**, point to the **BCS ground unit**.
5. In the gear next to it, configure the 4 incoming variables as System Variables: assign UAV Position to **Moving Object** and the 3 variables from attitude to 3 different **User Variables** (keeping the **same order** as well).



T. out 1.0

Address Veronte v4.5 278

In	Out
UAV position	Moving Object 01
Yaw	User Variable 10 (Real - 32 Bits)
Pitch	User Variable 11 (Real - 32 Bits)
Roll	User Variable 12 (Real - 32 Bits)

Fig. 17: **1x air unit - Sniffer**

3.5 External devices

The step-by-step instructions for the following external devices will be explained in detail in the following sections:

- *Altimeters*
- *Radios*
- *Servos*
- *Stick*
- *Veronte products*

3.5.1 Altimeters

3.5.1.1 Lidar

The integration between **Veronte BCS** and a lidar is performed using a variety of interfaces depending on the lidar device. The most common interfaces are I2C or analog although serial or CAN bus can also be used if the lidar is compatible.

3.5.1.1.1 I2C lidar

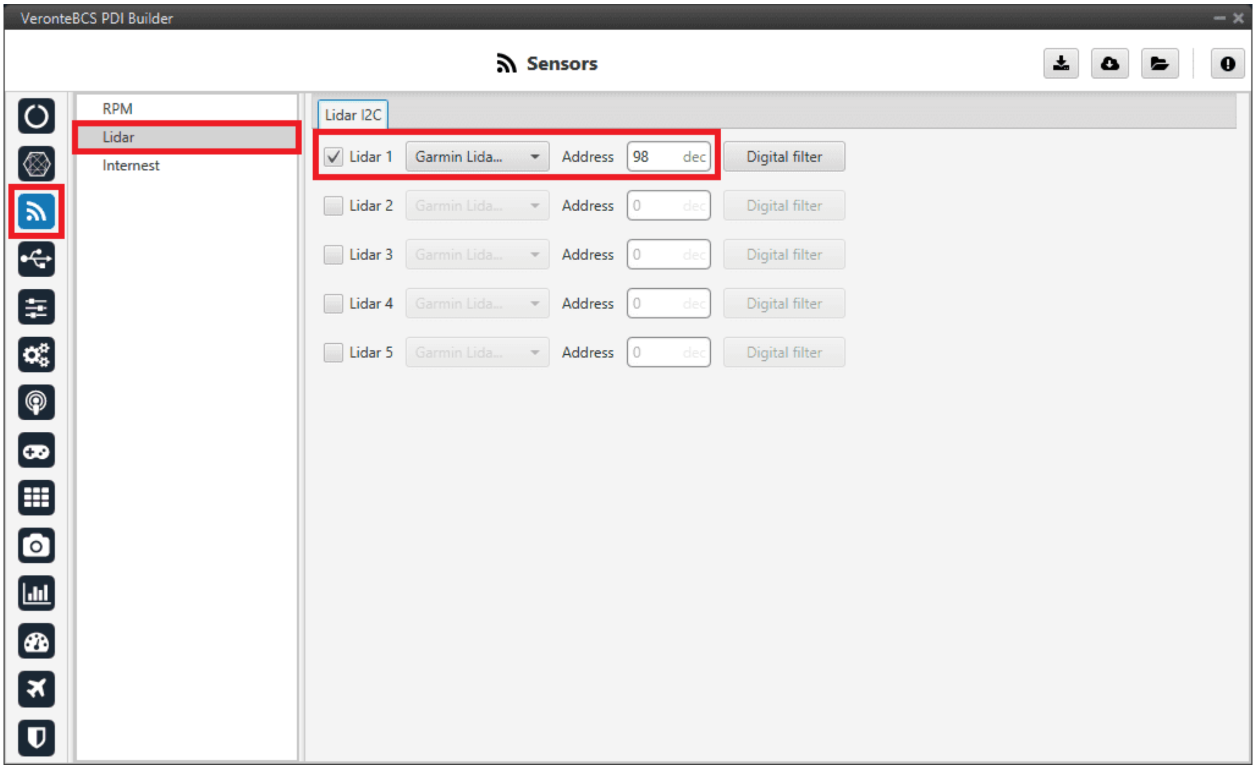
Connect the lidar following the pinout provided by the manufacturer and connect it to the **BCS I2C bus** following the Hardware installation - Pinout section of the **BCS Hardware Manual**.

In this case it is not needed to transform the lidar readings, the readings will be reported directly in the selected lidar distance variable.

Go to Sensors menu → **Lidar section**.

- Enable Lidar 1
- Select the desired Lidar from the drop-down menu
- Set the I2C address

More information on the available lidar options can be found in the *Lidar section* of this manual.



☑	Lidar 1	Garmin Lida...	Address	98 dec	Digital filter
□	Lidar 2	Garmin Lida...	Address	0 dec	Digital filter
□	Lidar 3	Garmin Lida...	Address	0 dec	Digital filter
□	Lidar 4	Garmin Lida...	Address	0 dec	Digital filter
□	Lidar 5	Garmin Lida...	Address	0 dec	Digital filter

Fig. 18: **I2C Lidar**

> **Warning:** I2C address will be different for different devices make sure to define it properly by checking the manufacturer documentation.

3.5.1.1.2 Using lidar readings

Once the information provided by a lidar sensor is stored in a system variable as *Lidar Distance* via an ADC reading, I2C, serial or CAN, the user has to set how this data will be considered. Common uses are: to consider the lidar data as external sensor or to trigger an action based on a predefined event.

- **Altimeter configuration**: The following operation must be configured in the *Block Programs* menu to consider the lidar measurement as an EKF input.

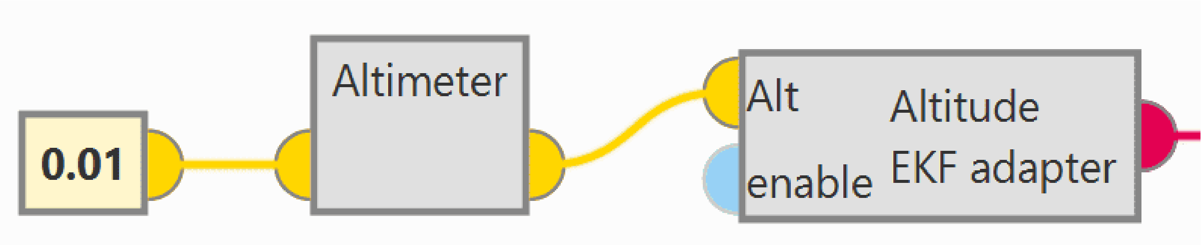
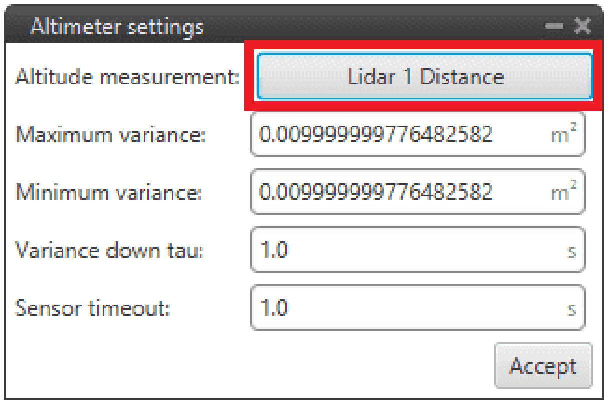


Fig. 19: **Altimer connection - Block Programs**

The *Lidar Distance* variable where the lidar measurement is stored must be selected. In this example, *Lidar 1 Distance* has been used:



Altitude measurement: Lidar 1 Distance

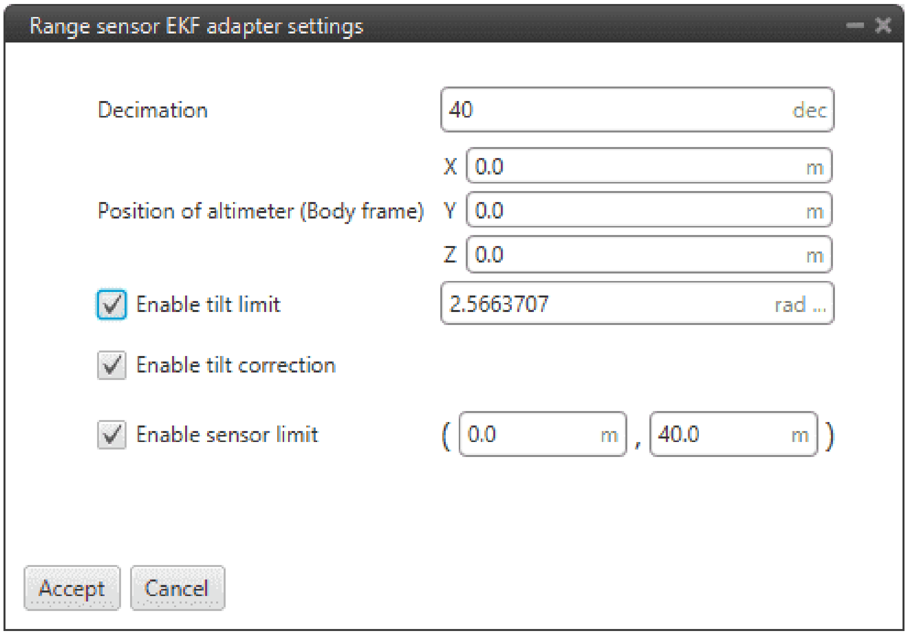
Maximum variance: 0.009999999776482582 m²

Minimum variance: 0.009999999776482582 m²

Variance down tau: 1.0 s

Sensor timeout: 1.0 s

Fig. 20: **Altimer sensor configuration - Block Programs**



Range sensor EKF adapter settings

Decimation 40 dec

Position of altimeter (Body frame)
X 0.0 m
Y 0.0 m
Z 0.0 m

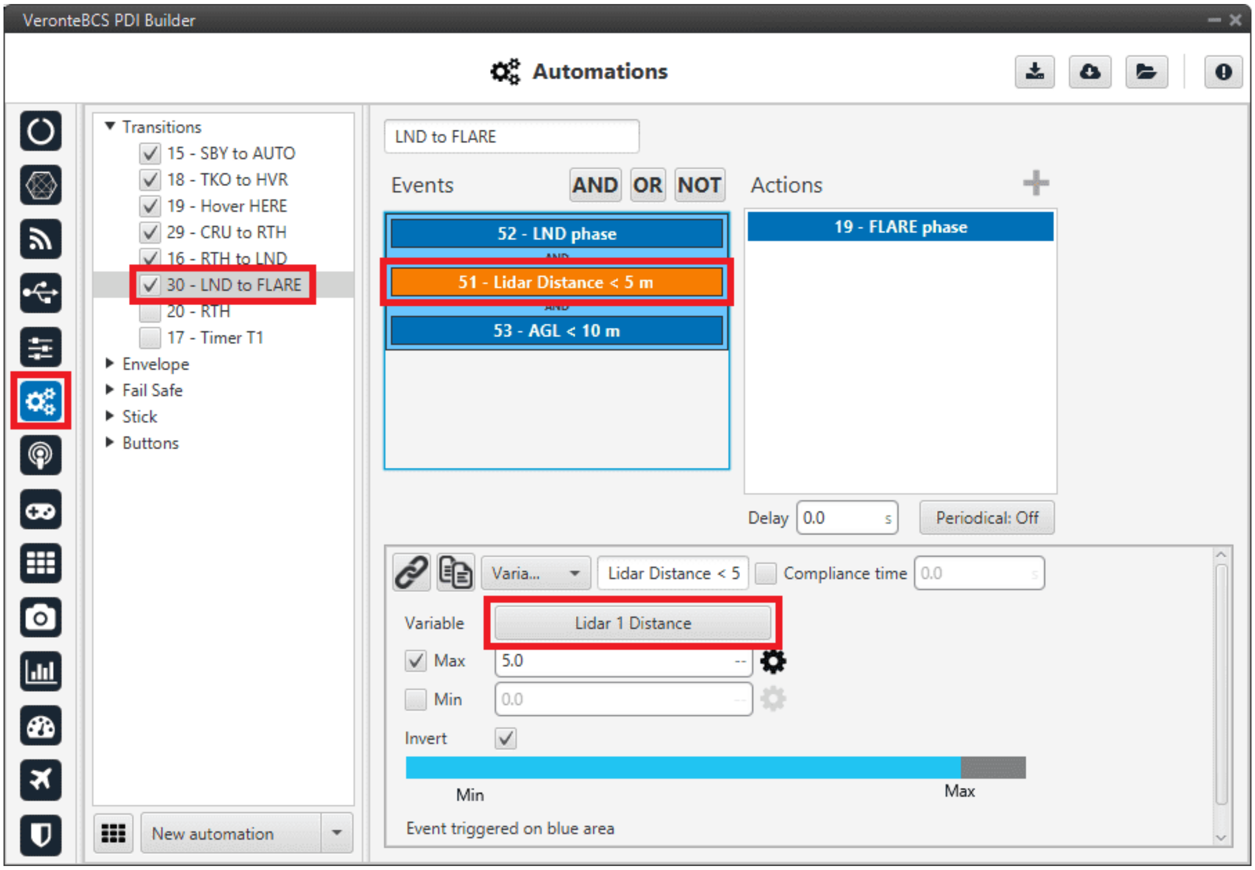
☑ Enable tilt limit 2.5663707 rad ...

☑ Enable tilt correction

☑ Enable sensor limit (0.0 m , 40.0 m)

Fig. 21: **Altitude EKF adapter configuration - Block Programs**

- **Automation**: This automation will trigger a change of phase, Flare phase, when the aircraft is landing and at 5 m AGL.



LND to FLARE

Events: 52 - LND phase AND 51 - Lidar Distance < 5 m AND 53 - AGL < 10 m

Actions: 19 - FLARE phase

Delay 0.0 s

Periodical: Off

Varia... Lidar Distance < 5 □ Compliance time 0.0 s

Variable: Lidar 1 Distance

☑ Max 5.0

□ Min 0.0

Invert ☑

Event triggered on blue area

Fig. 22: **Lidar automation example**

For more information on automations, see *Automations section*.

3.5.1.2 Radar

Radar altimeters are common devices on aircrafts.

3.5.1.2.1 Smartmicro CAN Radar

The following explanation corresponds to the integration of the **Smartmicro CAN Radar**. For more information on the Smartmicro Radar - Altimeter datasheet, the user can go to:

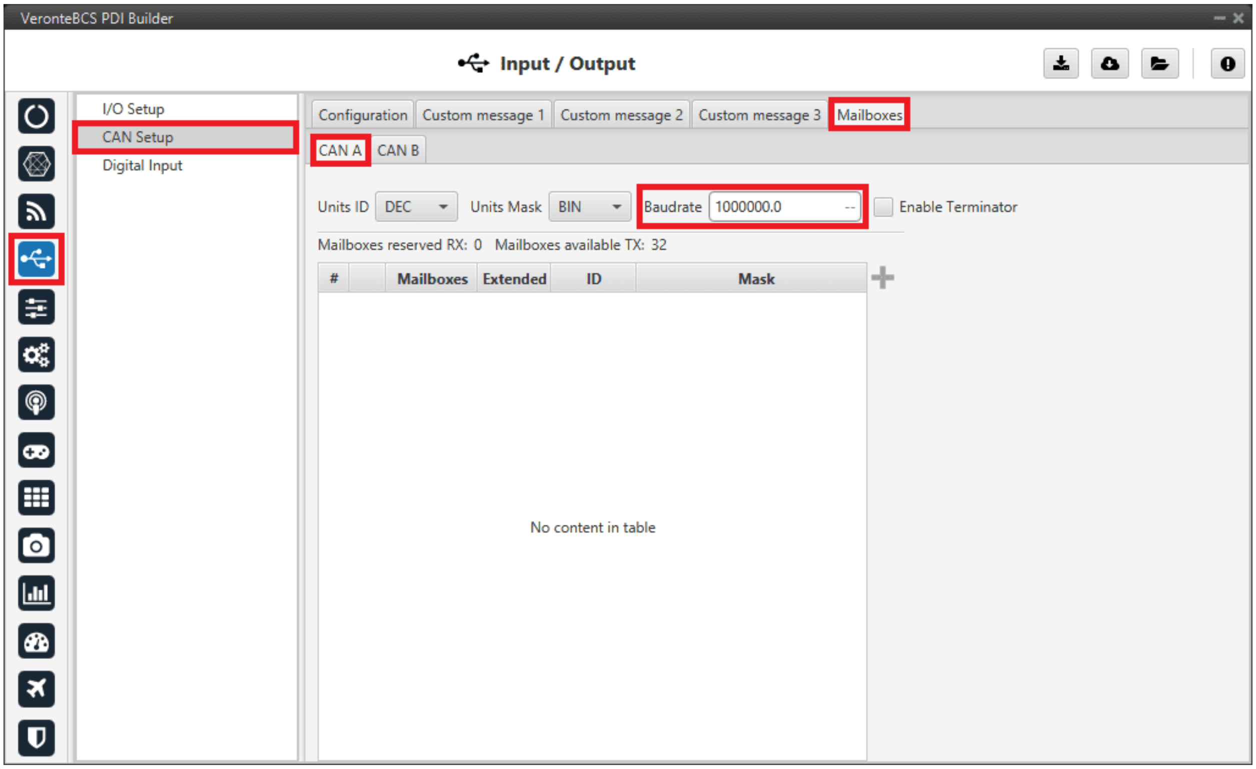
- Micro Radar Altimeter Web
- Micro Radar Altimeter Data Sheet

These settings will allow **BCS** to read out via **CAN A** the radar altimeter readings, in particular **AGL and vertical speed**.

Note: In the datasheet the user can access the complete protocol of the device.

1. Go to Input/Output menu → CAN Setup section → **Mailboxes tab**.

   Configure the **CAN A** baudrate:



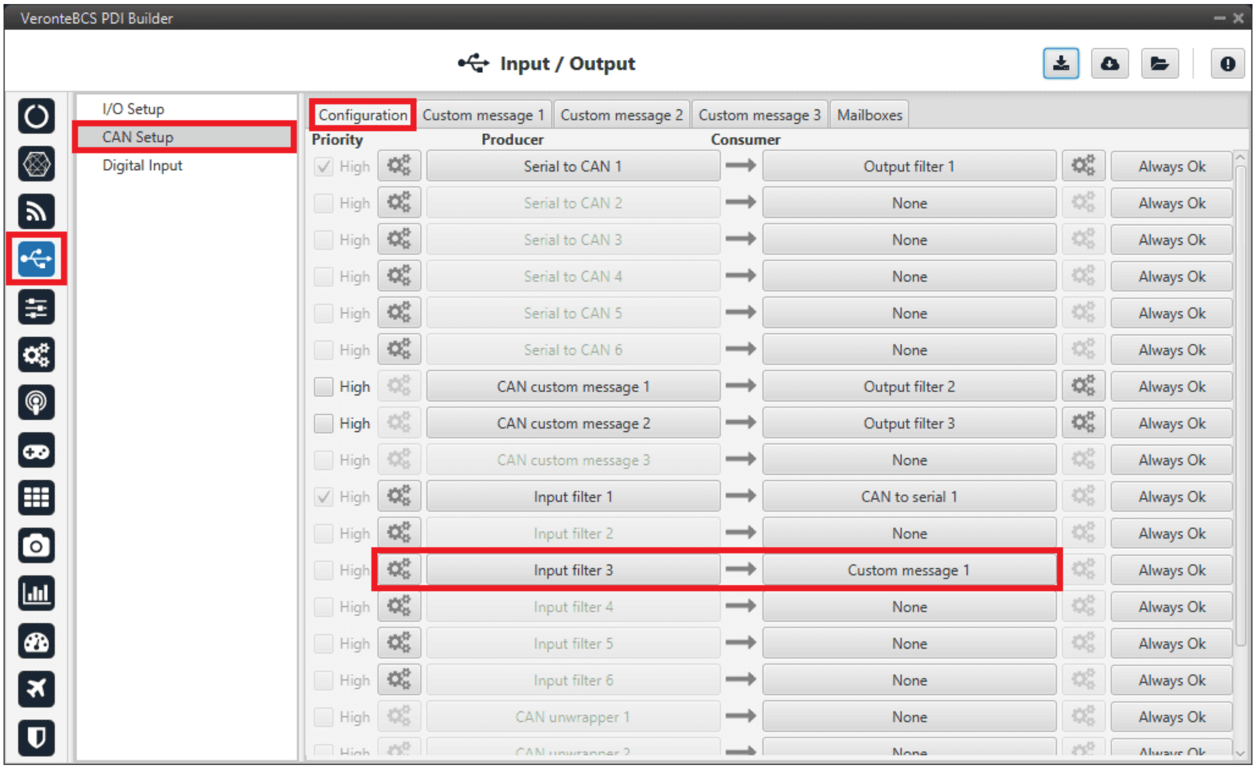
Units ID DEC Units Mask BIN Baudrate 1000000.0 -- □ Enable Terminator

Mailboxes reserved RX: 0 Mailboxes available TX: 32

Fig. 23: **Baudrate configuration**

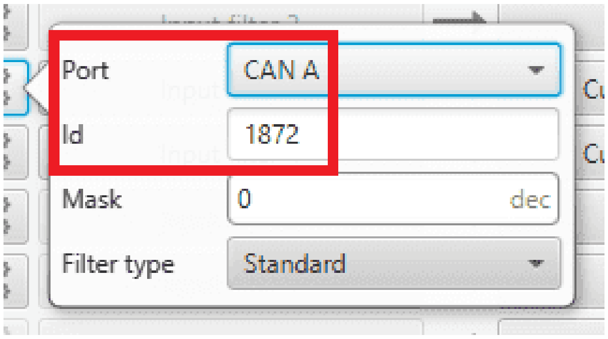
2. Once the baudrate is set, go to **Configuration tab**.

   Set an Input Filter to read from **CAN A** in **Custom message 1**, in this example *Input Filter 3*.



Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
□ High	Serial to CAN 2	→	None	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message 1	→	Output filter 2	Always Ok
□ High	CAN custom message 2	→	Output filter 3	Always Ok
□ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
□ High	Input filter 2	→	None	Always Ok
□ High	Input filter 3	→	Custom message 1	Always Ok
□ High	Input filter 4	→	None	Always Ok
□ High	Input filter 5	→	None	Always Ok
□ High	Input filter 6	→	None	Always Ok
□ High	CAN unwrapper 1	→	None	Always Ok
□ High	CAN unwrapper 2	→	None	Always Ok

Fig. 24: **CAN Setup - Input Filter**



Port: CAN A

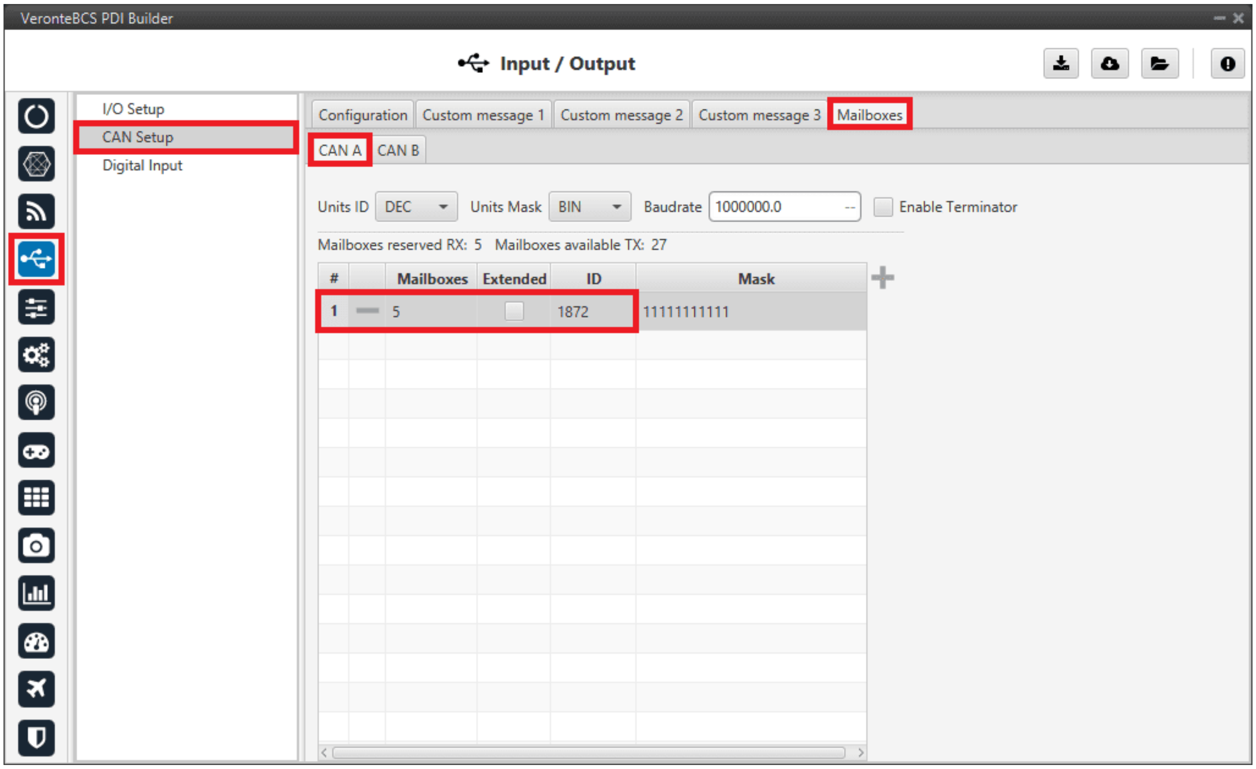
Id: 1872

Mask: 0 dec

Filter type: Standard

Fig. 25: **CAN Setup - Input Filter configuration**

3. After specifying that **Custom message 1** will receive the data from **CAN A**, go to **Mailboxes tab**. Configure **CAN A mailboxes** for **CAN ID message: 1872**.



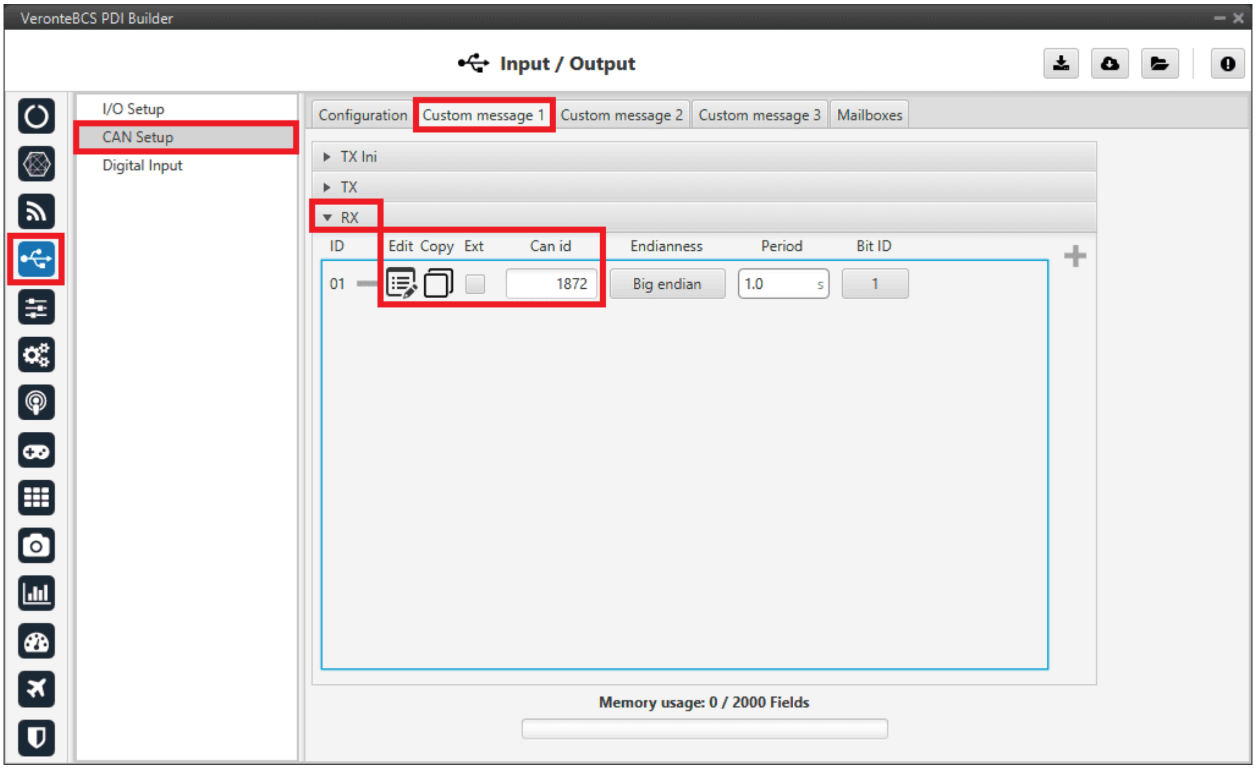
Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

Mailboxes reserved RX: 5 Mailboxes available TX: 27

#	Mailboxes	Extended	ID	Mask
1	5	□	1872	11111111111

Fig. 26: **Mailboxes configuration**

4. Go to **Custom message 1 tab**.
   - Add a new message in **RX** with **ID 1872**.

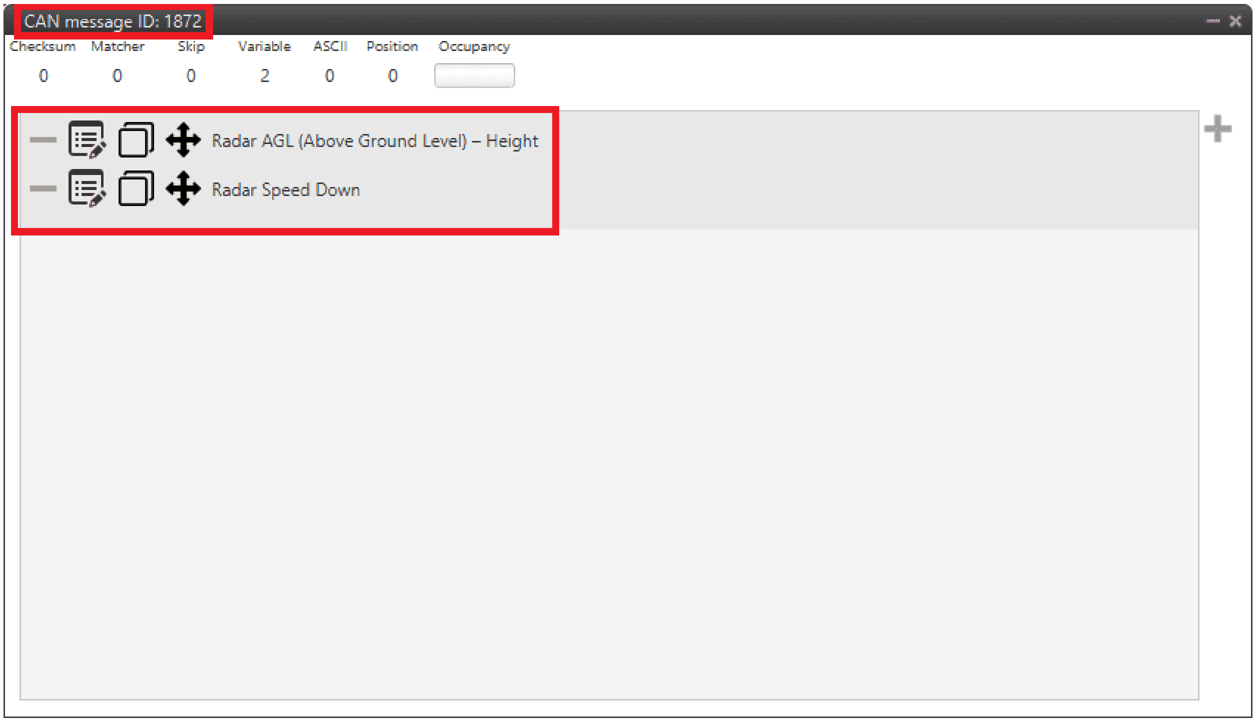


ID	Edit	Copy	Ext	Can id	Endianness	Period	Bit ID
01			□	1872	Big endian	1.0 s	1

Memory usage: 0 / 2000 Fields

Fig. 27: **Custom message 1**

- Define the content of the incoming message:


CAN message ID: 1872

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	0	0	2	0	0	

Radar AGL (Above Ground Level) – Height

Radar Speed Down

Fig. 28: **Custom message 1 configuration**

For more details on CAN configuration see *CAN Setup section*.

Note: CAN ID messages and messages content will change for different Radar altimeters. Check the documentation of your device for further details.

3.5.2 Radios

3.5.2.1 Digi internal radio

3.5.2.1.1 Configuration

This section describes the necessary configuration for **BCS PDI Builder** and the **Digi radio software** (XCTU) to allow a **correct communication between BCS and its internal Digi radio**.

To configure the communication between **BCS** and its internal Digi radio, apply the following steps:

1. Connect the **BCS** to a computer with **Veronte Link**, read the user manual to use it.

Configuration in BCS PDI Builder

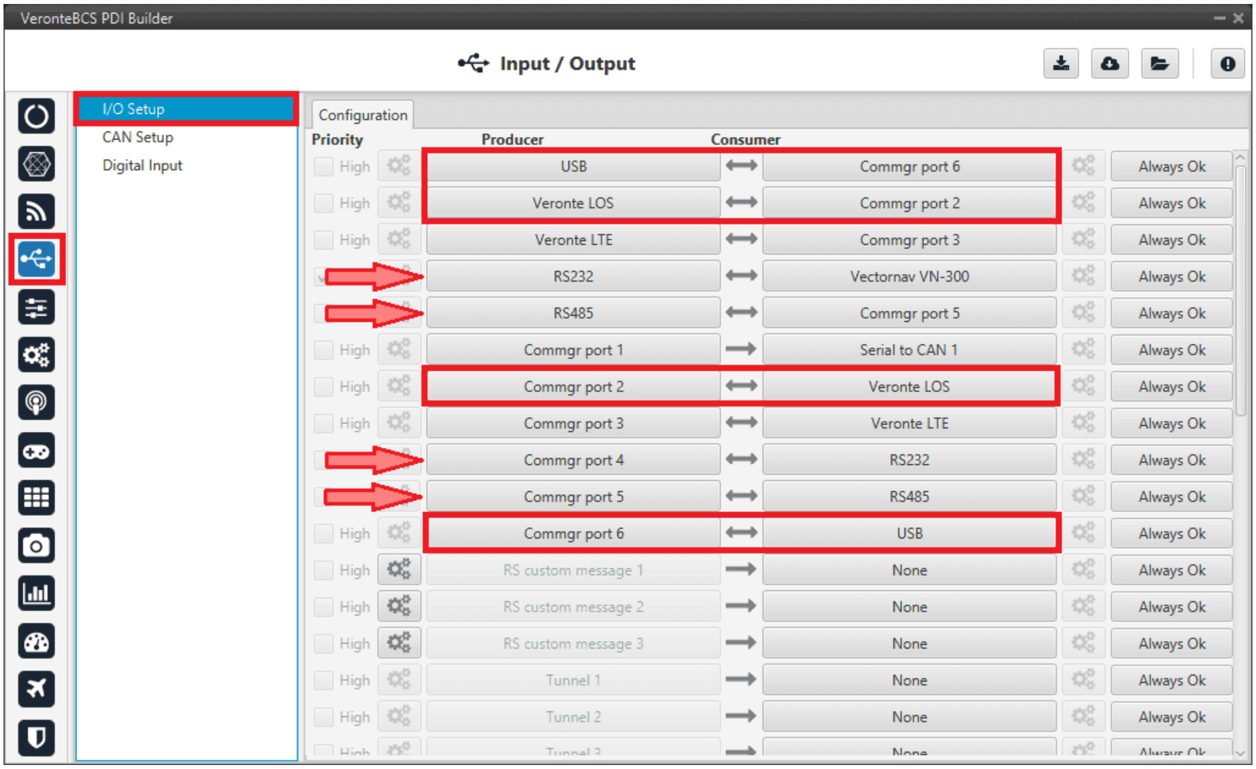
2. Go to Input/Output menu → **I/O Setup section**.

   Since the configuration of this menu is going to be modified **temporarily**, i.e. the current configuration will have to be re-established, just to be able to set up a tunnel between the control station and the radio.

   It is necessary that the user **first annotates the configuration** of **USB**, **Veronte LOS** and the ports to which they are connected. The following image shows an example.

   Note: It is recommended to take a screenshot for this step.

   Warning: Although the connection with **Veronte BCS** will be lost via USB, users can still "see" the **BCS** via serial (RS232 or RS485). For this purpose, the **bidirectional RS232 or RS485 connection must not be modified**.

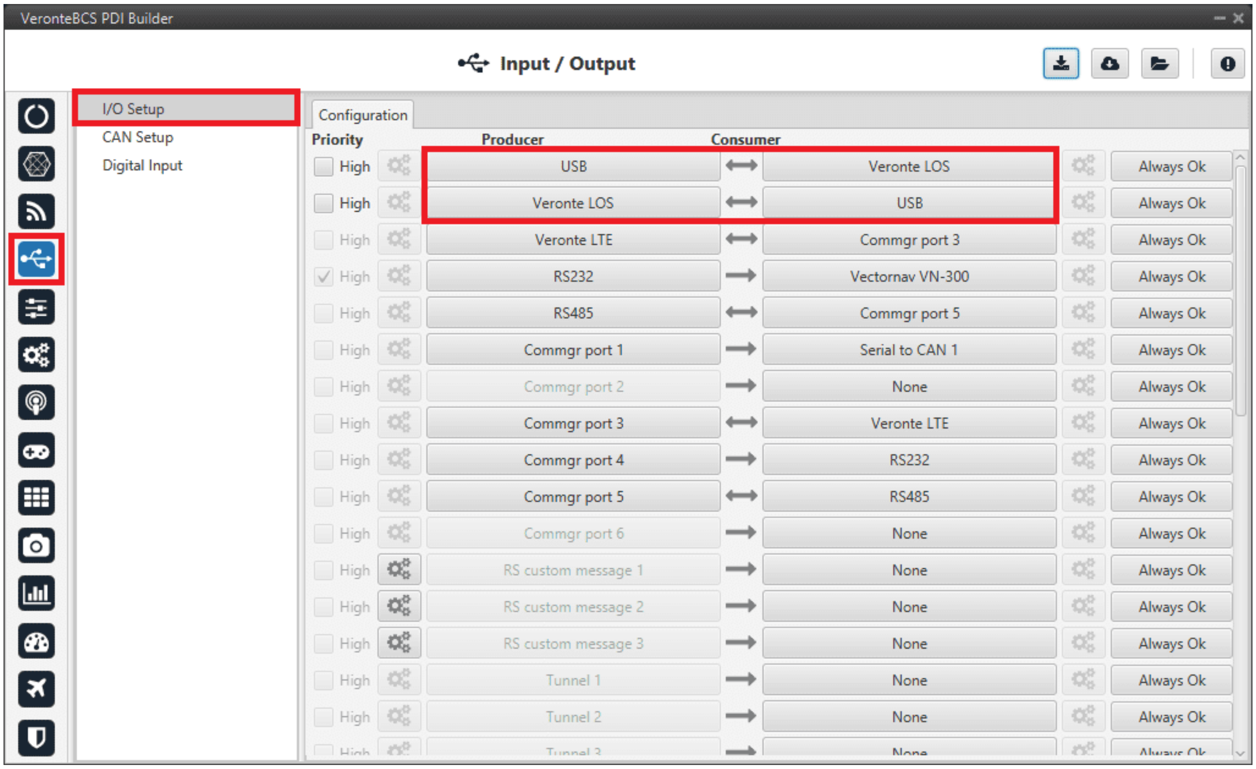


Priority	Producer		Consumer	
High	USB	⟷	Commgr port 6	Always Ok
High	Veronte LOS	⟷	Commgr port 2	Always Ok
High	Veronte LTE	⟷	Commgr port 3	Always Ok
	RS232	⟷	Vectornav VN-300	Always Ok
	RS485	⟷	Commgr port 5	Always Ok
High	Commgr port 1	→	Serial to CAN 1	Always Ok
High	Commgr port 2	⟷	Veronte LOS	Always Ok
High	Commgr port 3	⟷	Veronte LTE	Always Ok
	Commgr port 4	⟷	RS232	Always Ok
	Commgr port 5	⟷	RS485	Always Ok
High	Commgr port 6	⟷	USB	Always Ok
High	RS custom message 1	→	None	Always Ok
High	RS custom message 2	→	None	Always Ok
High	RS custom message 3	→	None	Always Ok
High	Tunnel 1	→	None	Always Ok
High	Tunnel 2	→	None	Always Ok
High	Tunnel 3	→	None	Always Ok

Fig. 29: **Example of configuration of USB and Veronte LOS ports**

3. Change the port which **USB producer** is connected to and select **Veronte LOS** as **consumer**.

   USB and **Veronte LOS** must have **bidirectional communication** ⟷.



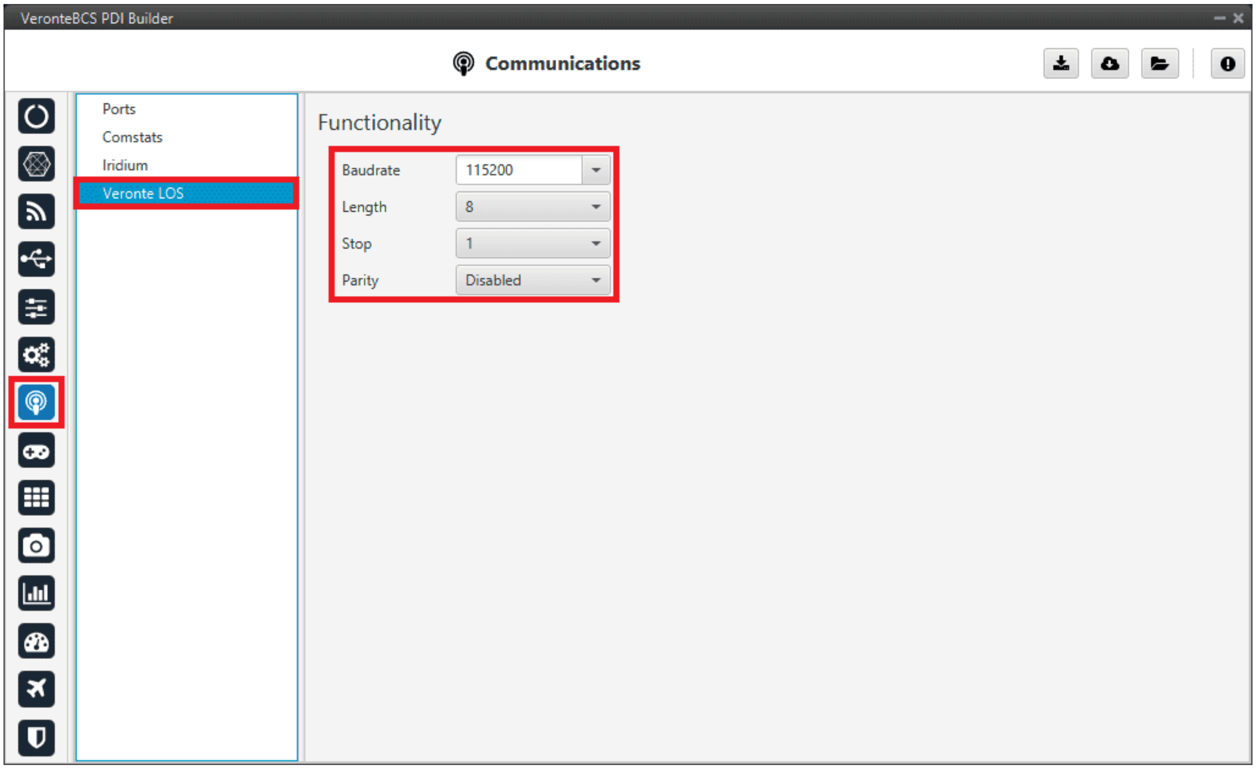
Priority	Producer		Consumer	
□ High	USB	⟷	Veronte LOS	Always Ok
□ High	Veronte LOS	⟷	USB	Always Ok
□ High	Veronte LTE	⟷	Commgr port 3	Always Ok
☑ High	RS232	→	Vectornav VN-300	Always Ok
□ High	RS485	⟷	Commgr port 5	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	→	None	Always Ok
□ High	Commgr port 3	⟷	Veronte LTE	Always Ok
□ High	Commgr port 4	→	RS232	Always Ok
□ High	Commgr port 5	⟷	RS485	Always Ok
□ High	Commgr port 6	→	None	Always Ok
□ High	RS custom message 1	→	None	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	→	None	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 30: **USB** ⟷ **Veronte LOS**

4. Go to Communications menu → **Veronte LOS section**.

   It is important to know which **baudrate** is configured for the **Veronte LOS serial port** in order to **match** it with the one configured in the **Digi radio**.

   By default, the baudrate configured in **BCS PDI Builder** is set to 115200.



Functionality

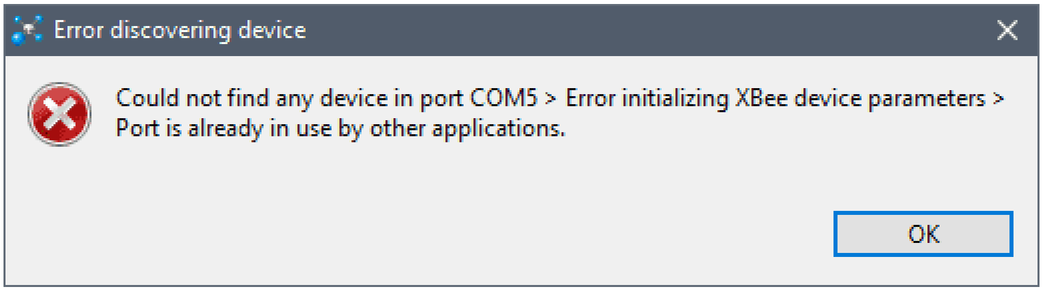
Baudrate 115200
Length 8
Stop 1
Parity Disabled

Fig. 31: **Veronte LOS baudrate**

5. Click on 💾 to apply changes to the control station.

Warning: The communication between computer and **BCS** will be disconnected, since the control station is working as a tunnel between computer and radio. The computer will be communicating only with the Digi radio.

6. **Wait for the device to disconnect and close Veronte Link**. If the user does not close it, XCTU software will not be able to detect the radio as the COM is being managed by Veronte Link, and the following error message will appear:


Error discovering device

Could not find any device in port COM5 > Error initializing XBee device parameters > Port is already in use by other applications.

Fig. 32: **XCTU error message**

Important: Remember that to completely close the application the user must close it from the windows system tray.

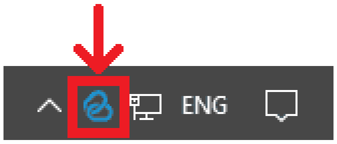


Fig. 33: **Close Veronte Link**

Configuration in Digi radio software

7. Download and install XCTU (Digi radio software).
8. Build a configuration for 'air' or 'bcs' in **XCTU**:

   The integrated radio is the model **DIGI-XBEE3 XB3-24Z8UM**. For more information about how to configure it, read the XCTU User Guide.

   The following table shows which parameters can be configured. The rest of parameters should remain as default.

DIGI Parameter	Description
PL	Transmit power (100 mW)
ID	Network addres PAN ID
DD	Device type identifier
BD	UART baud rate (115200)
RR	Retries (minimum 5)
CH	2.4 GHz channel to send
MM	Mac mode, 802.15.4 with Digi header for discovery and packages duplicate
CA	Clear channel threshold as dBm
EA	Ack failures
EC	Failure to sent due to excess of energy in channel

Warning: Check that the baudrate of the radio matches the baudrate configured in *BCS PDI Builder**. If it is not the same, change one of them to match.

9. After configuring the radio, the communication between computer and **BCS** should be restored. To do it, force the maintenance mode.

Configuration in BCS PDI Builder

10. Go to Input/Output menu → **I/O Setup section**.

    Finally, after configuring the Digi radio in its software, restore the annotated **USB** and **Veronte LOS** configuration (step 2).

If after the whole process described above for setting up the radio or later during operation the **communication between the Digi radio and the BCS is lost**, please check the *Communication lost with intenal Digi radio - > Troubleshooting section* of this manual.

3.5.2.1.2 Operational range

The following table is a **reference** of the functional range for each telemetry load (**it may be affected by enviromental conditions**):

	Frequency		
Load	**5 Hz**	**10 Hz**	**20 Hz**
Low (Half telemetry vector)	> 700 m	500 m	300 m
Medium (one telemetry vector)	> 700 m	100 m	80 m
High (two or more telemetry vectors)	300 m	80 m	X

Note: Telemetry vectors are structured messages with up to 255 bytes of data. To know more about them, read VCP manual -> Message structure.

3.5.2.2 External radios

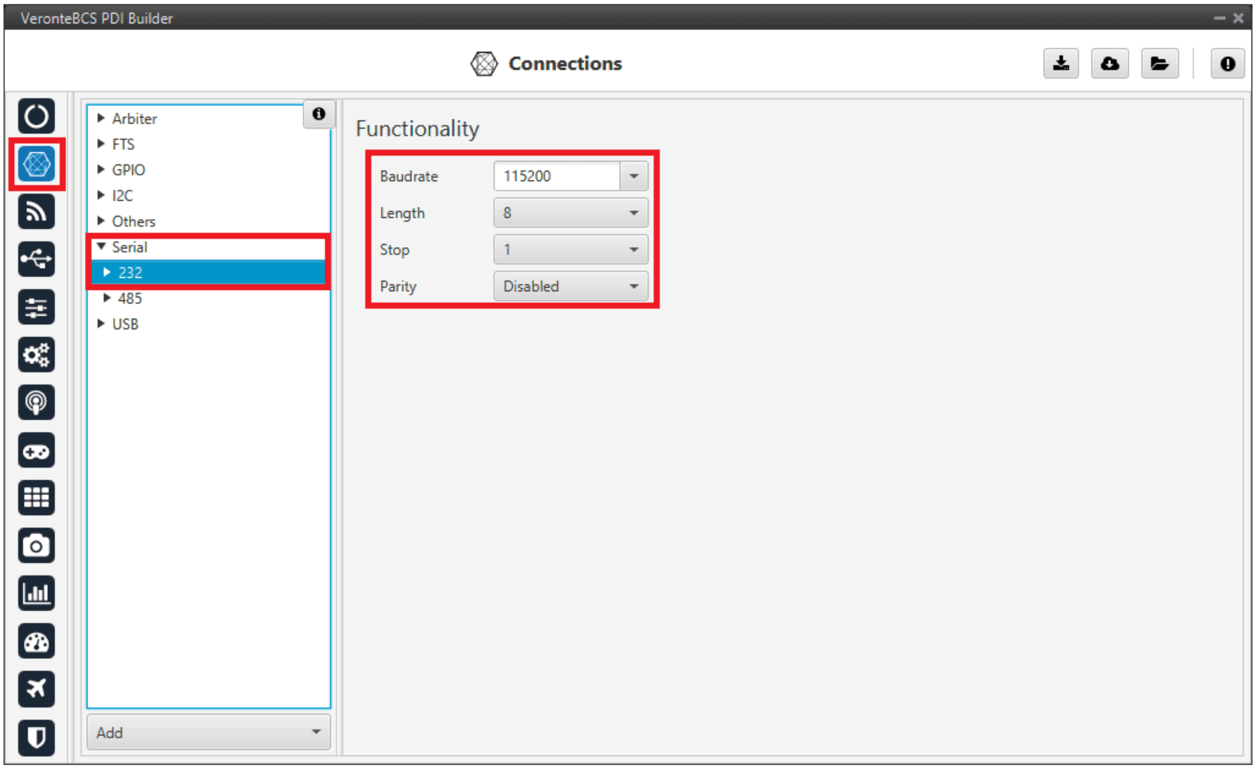
This section describes the required configuration to be performed in **BCS PDI Builder** to allow a **correct connection between BCS and any external radio**.

External radios compatible with Veronte, such as *Microhard*, *DTC*, *Digi*, *Silvus* and *Veronte Data Link* (Embention external radio, contact sales@embention.com for more information).

After configuring the external radio in the corresponding software, follow the steps below:

1. Go to **Connections menu** → Serial section → **232 tab**.

   Check that these parameters are the same as the parameter values previously set in the external radio.



Functionality

Baudrate 115200

Length 8

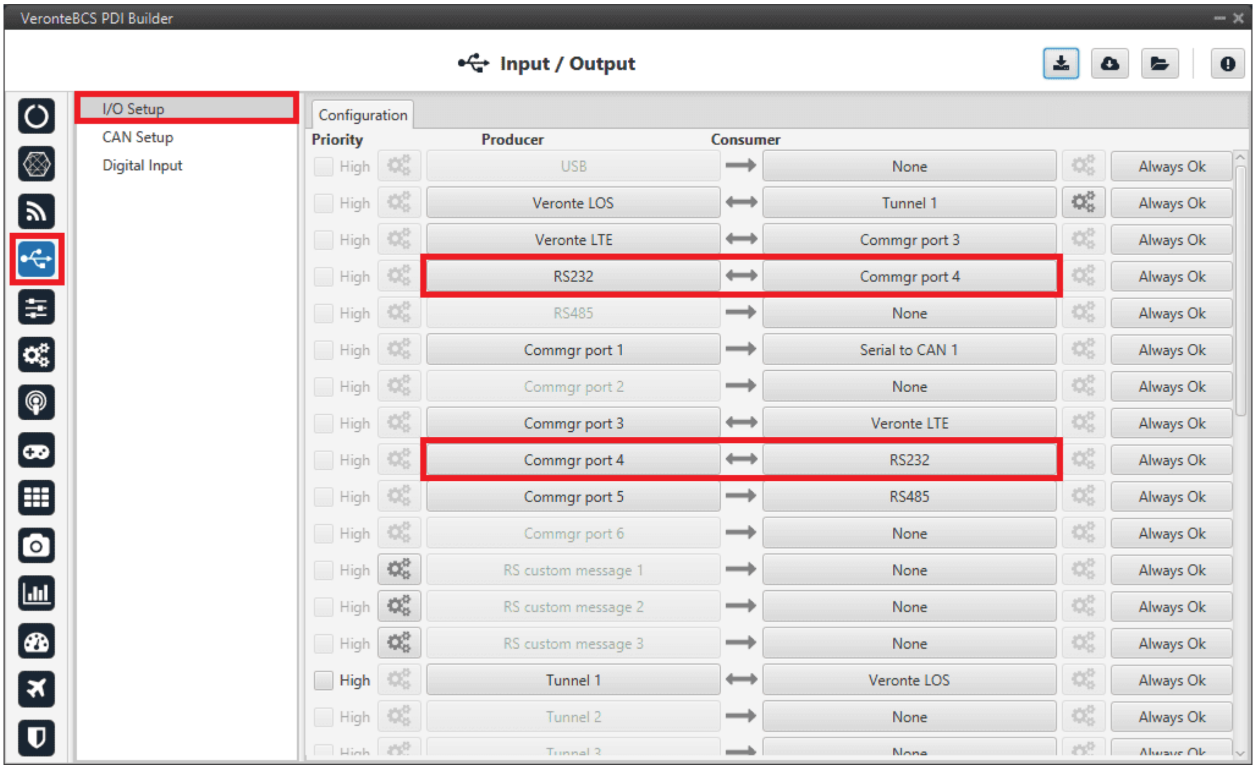
Stop 1

Parity Disabled

Fig. 34: **RS-232 connection configuration**

2. Go to Input/Output menu → **I/O Setup section**.

   RS-232 has to be configured as a **bidirectional** commgr port.



Priority	Producer		Consumer	
□ High	USB	→	None	Always Ok
□ High	Veronte LOS	↔	Tunnel 1	Always Ok
□ High	Veronte LTE	↔	Commgr port 3	Always Ok
□ High	RS232	↔	Commgr port 4	Always Ok
□ High	RS485	→	None	Always Ok
□ High	Commgr port 1	→	Serial to CAN 1	Always Ok
□ High	Commgr port 2	→	None	Always Ok
□ High	Commgr port 3	↔	Veronte LTE	Always Ok
□ High	Commgr port 4	↔	RS232	Always Ok
□ High	Commgr port 5	→	RS485	Always Ok
□ High	Commgr port 6	→	None	Always Ok
□ High	RS custom message 1	→	None	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	↔	Veronte LOS	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 35: **RS-232 I/O configuration**

Note: These settings have to be made in both Veronte units (**BCS** and **1x air**).

3.5.3 Servos

The user can configure any actuator compatible with the communication interfaces.

3.5.3.1 Serial

Serial servos are configured differently than PWM servos as the protocol of a serial device must be defined with *custom messages*.

In this case a **PWM** variable must be sent through a serial interface.

3.5.3.1.1 Volz DA26 - RS485

Firstly, the following wiring connection is recommended for a **RS485 connection between Volz DA26 servos and BCS**:

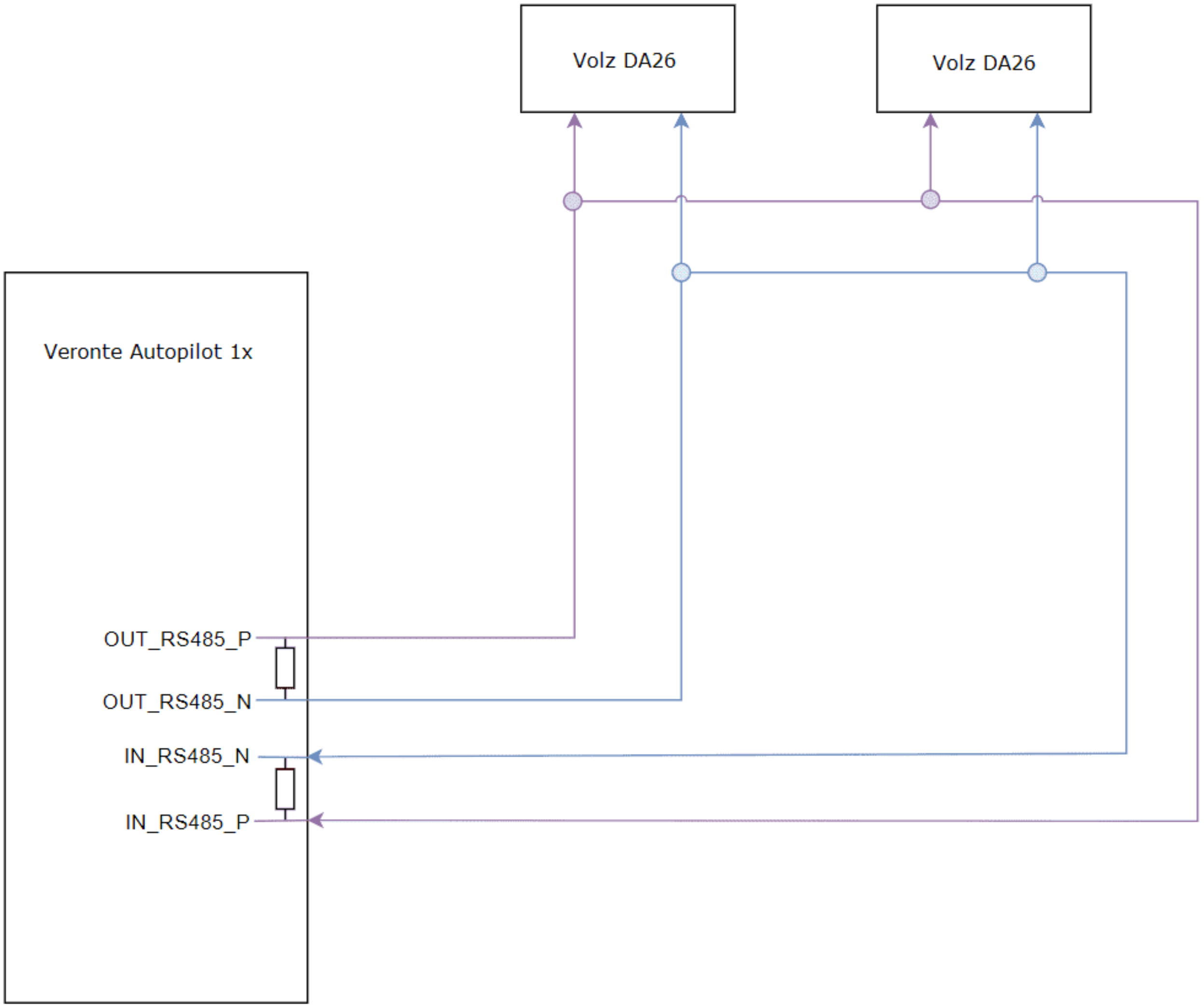


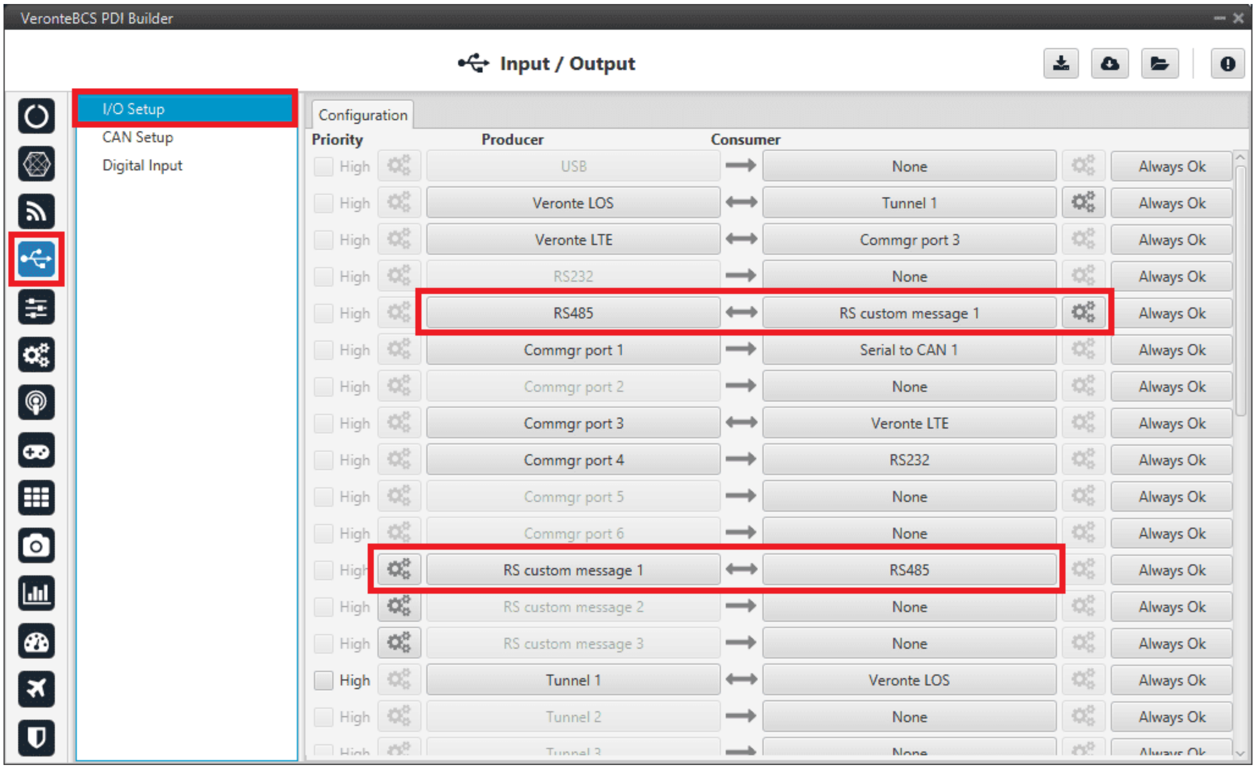
Fig. 36: **Volz DA26 - BCS wiring connection**

The above diagram is made for the case where 2 Volz DA26 servos are connected, however, the connection is the same in case the user wants to connect only one or as many servos as the bus allows.

Follow the steps below to configure a **Volz DA26 servo via RS-485**.

1. Go to Input/Output menu → **I/O Setup section**.

   Bidirectionally connect the **RS485** port to a **RS custom message**, in this example *RS custom message 1* is used:

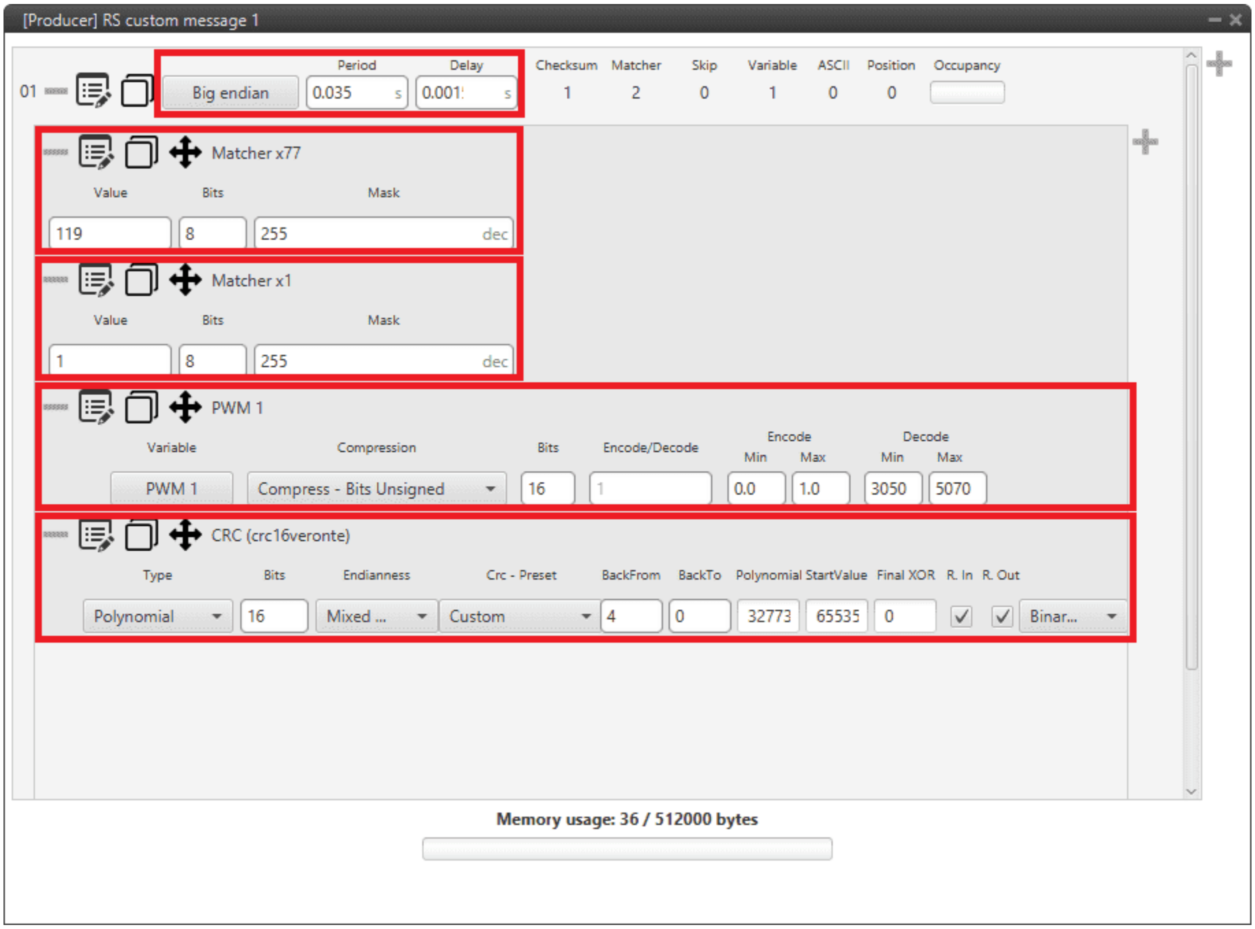


Priority	Producer		Consumer	
☐ High	USB	→	None	Always Ok
☐ High	Veronte LOS	↔	Tunnel 1	Always Ok
☐ High	Veronte LTE	↔	Commgr port 3	Always Ok
☐ High	RS232	→	None	Always Ok
☐ High	RS485	↔	RS custom message 1	Always Ok
☐ High	Commgr port 1	→	Serial to CAN 1	Always Ok
☐ High	Commgr port 2	→	None	Always Ok
☐ High	Commgr port 3	↔	Veronte LTE	Always Ok
☐ High	Commgr port 4	→	RS232	Always Ok
☐ High	Commgr port 5	→	None	Always Ok
☐ High	Commgr port 6	→	None	Always Ok
☐ High	RS custom message 1	↔	RS485	Always Ok
☐ High	RS custom message 2	→	None	Always Ok
☐ High	RS custom message 3	→	None	Always Ok
☐ High	Tunnel 1	↔	Veronte LOS	Always Ok
☐ High	Tunnel 2	→	None	Always Ok
☐ High	Tunnel 3	→	None	Always Ok

Fig. 37: **RS485 ↔ RS custom message 1**

2. Configure the *RS custom message 1* producer by defining the protocol specified by the manufacturer:

Note: As the RS-485 is a **Half Full duplex** serial port, **BCS** needs this serial to be free for a certain time in order to receive the servo response. This is done by setting the **Delay** parameter.



[Producer] RS custom message 1

	Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01 Big endian	0.035 s	0.001! s	1	2	0	1	0	0	

Matcher x77

Value	Bits	Mask
119	8	255 dec

Matcher x1

Value	Bits	Mask
1	8	255 dec

PWM 1

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
PWM 1	Compress - Bits Unsigned	16	1	0.0	1.0	3050	5070

CRC (crc16veronte)

Type	Bits	Endianness	Crc - Preset	BackFrom	BackTo	Polynomial	StartValue	Final XOR	R. In	R. Out	
Polynomial	16	Mixed ...	Custom	4	0	32773	65535	0	☑	☑	Binar...

Memory usage: 36 / 512000 bytes

Fig. 38: **RS custom message 1 - Manufacturer's communication protocol**

- **Endianness**: Big endian
- **Period**: 0.035
- **Delay**: 0.0015
  - **Matcher x77**: Silent mode command (0x77).
    - **Value**: 119
    - **Bits**: 8
    - **Mask**: 255
  - **Matcher x1**: Servo interface Id = 1. The Id will be different for each servo and/or interface.
    - **Value**: 1
    - **Bits**: 8
    - **Mask**: 255
  - **PWM 1**: *PWM* is the variable that carries the information that has to be applied to the servo. Therefore, it must be included in the message.
    - **Variable**: PWM 1
    - **Compression**: Compress - Bits Unsigned

    - **Encode**: 0 / 1
    - **Decode**: 3050 / 5070
  - **CRC (Custom)**: A *Checksum* is needed to complete the communication protocol.
    - **Type**: Polynomial
    - **Bits**: 16
    - **Endianness**: Mixed endian
    - **CRC - Preset**: Custom
    - **BackFrom**: 4
    - **BackTo**: 0
    - **Polynomial**: 32773
    - **Start Value**: 65535
    - **Final XOR**: 0

> **Note:** For more information on checksum, see Checksum (CRC) explanation of the **1x PDI Builder** manual.

3.5.4 Stick

3.5.4.1 PPM Stick

3.5.4.1.1 General case: BCS unit sends commands directly to the 1x air unit

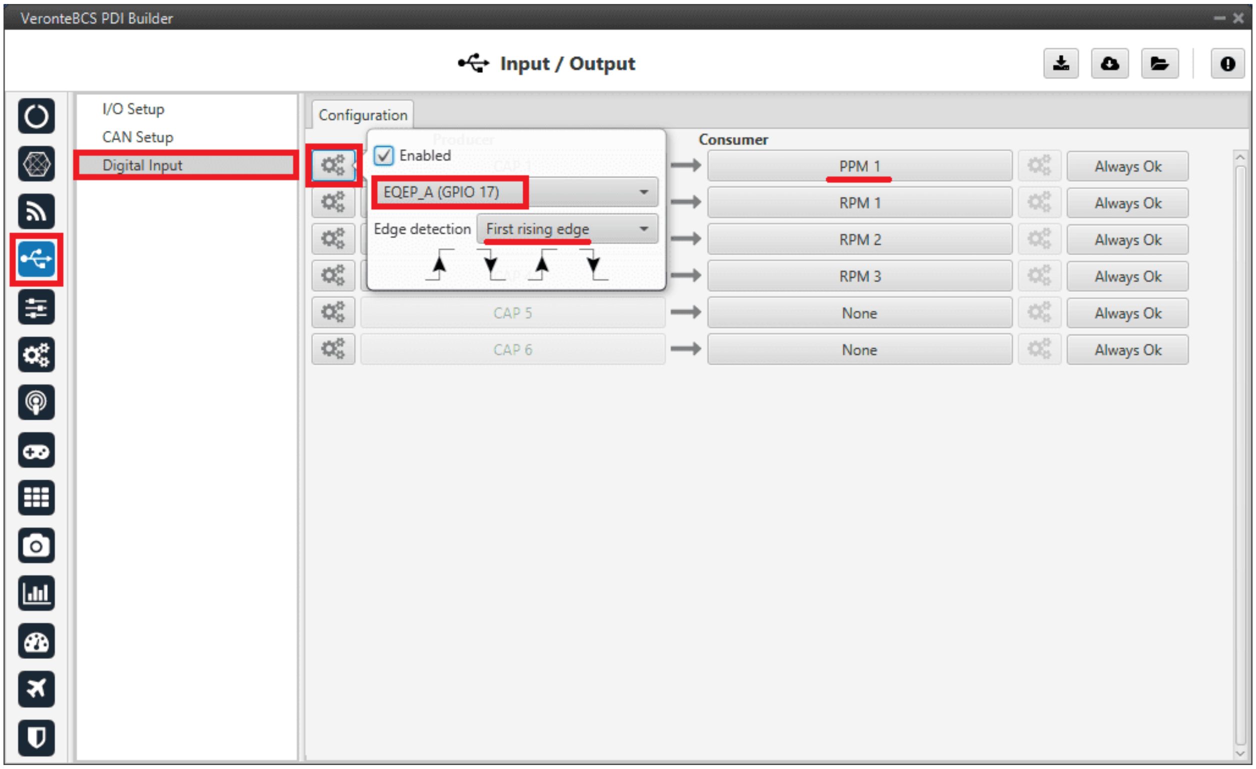
Follow the steps below to perform a correct stick configuration on the **BCS** and **1x** units.

BCS unit (BCS PDI Builder configuration)

1. Go to Input/Output menu → **Digital Input section**.

   Make sure that the following parameters have been configured:

   - Producer: **CAP 1**
     - Enabled
     - Select the pin to which the transmitter is connected (normally *EQEP A (i.e., GPIO 17)*)
     - Edge detection: First rising edge
   - Consumer: **PPM 1**



☑ Enabled

EQEP_A (GPIO 17)

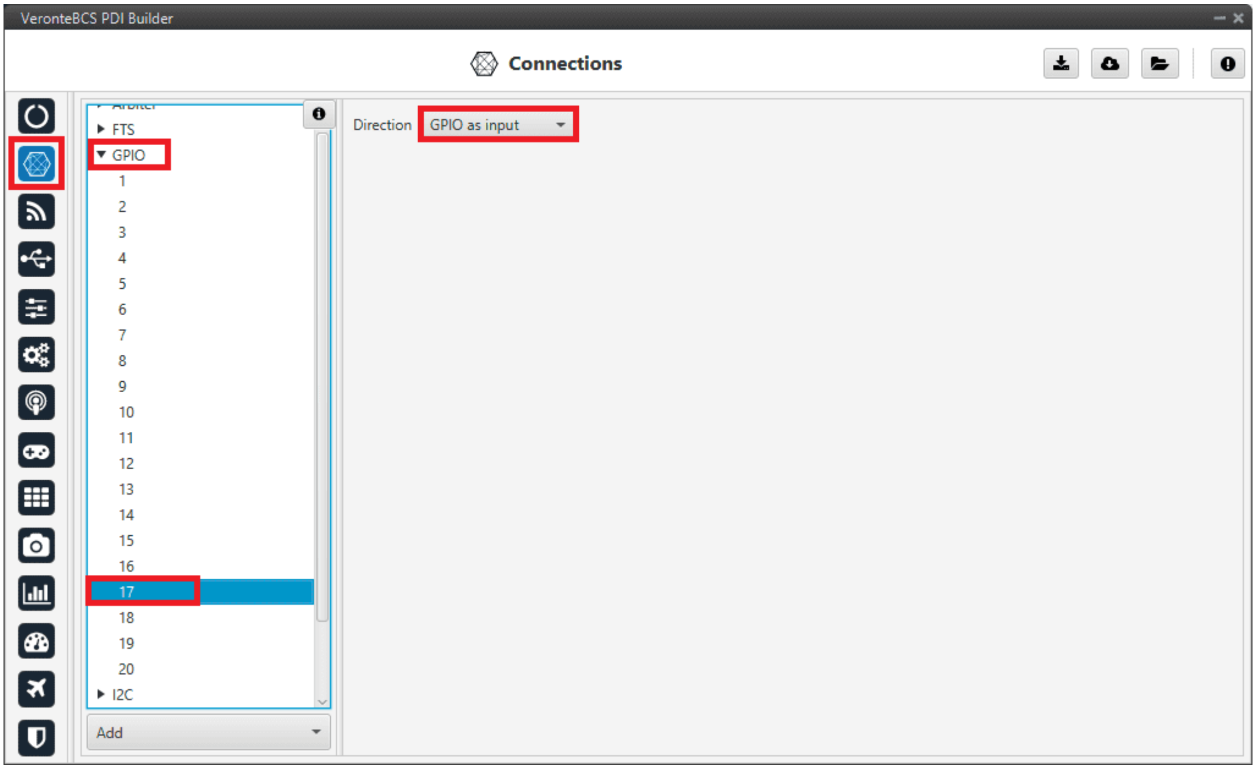
Edge detection: First rising edge

Producer	Consumer	
	PPM 1	Always Ok
	RPM 1	Always Ok
	RPM 2	Always Ok
	RPM 3	Always Ok
CAP 5	None	Always Ok
CAP 6	None	Always Ok

Fig. 39: **Stick - Digital Input configuration**

2. Go to Connections menu → **GPIO section**.

   Verify that the pin to which the transmitter is connected, in this case *GPIO 17 (i.e., EQEP A)*, is set as input.

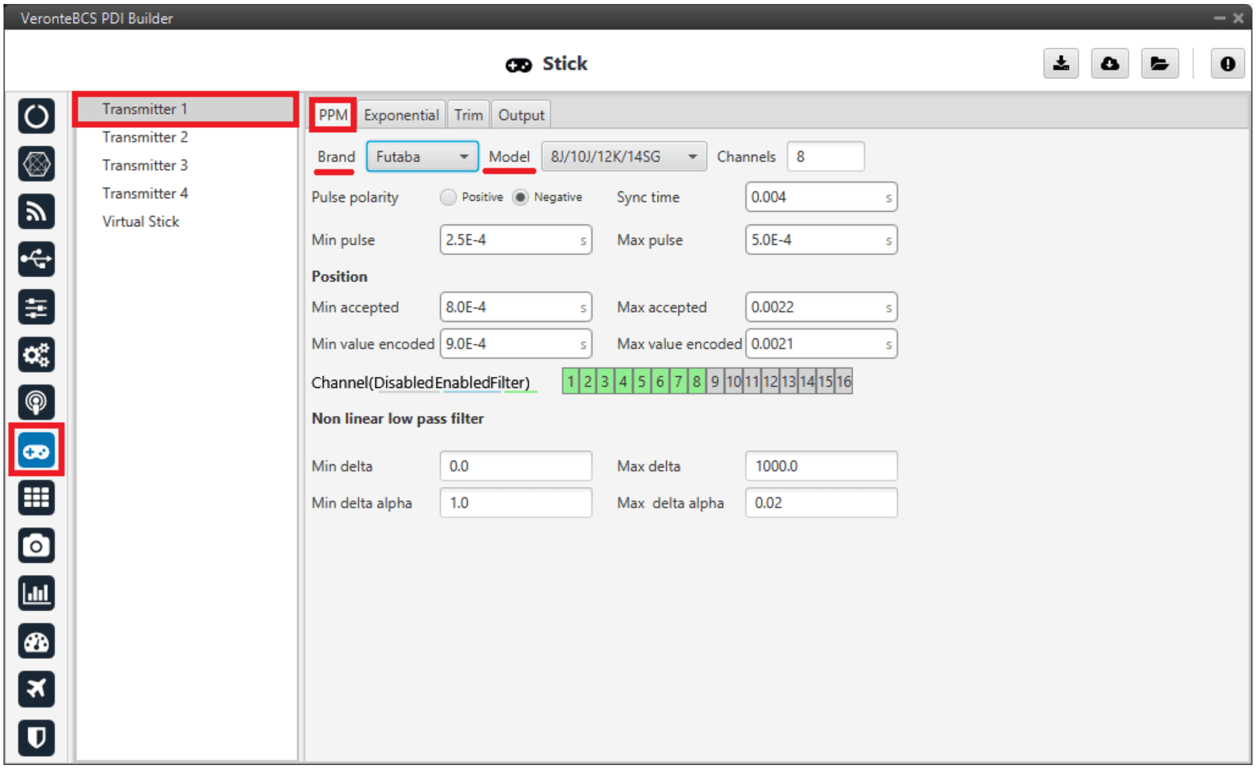


Direction: GPIO as input

Fig. 40: **Stick - GPIO/EQEP configuration**

3. Go to Stick menu → Transmitter 1 section → **PPM tab**.

   Select the brand of transmitter that applies.



Brand Futaba | Model 8J/10J/12K/14SG | Channels 8

Pulse polarity ○ Positive ● Negative | Sync time 0.004 s

Min pulse 2.5E-4 s | Max pulse 5.0E-4 s

Position

Min accepted 8.0E-4 s | Max accepted 0.0022 s

Min value encoded 9.0E-4 s | Max value encoded 0.0021 s

Channel(DisabledEnabledFilter) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Non linear low pass filter

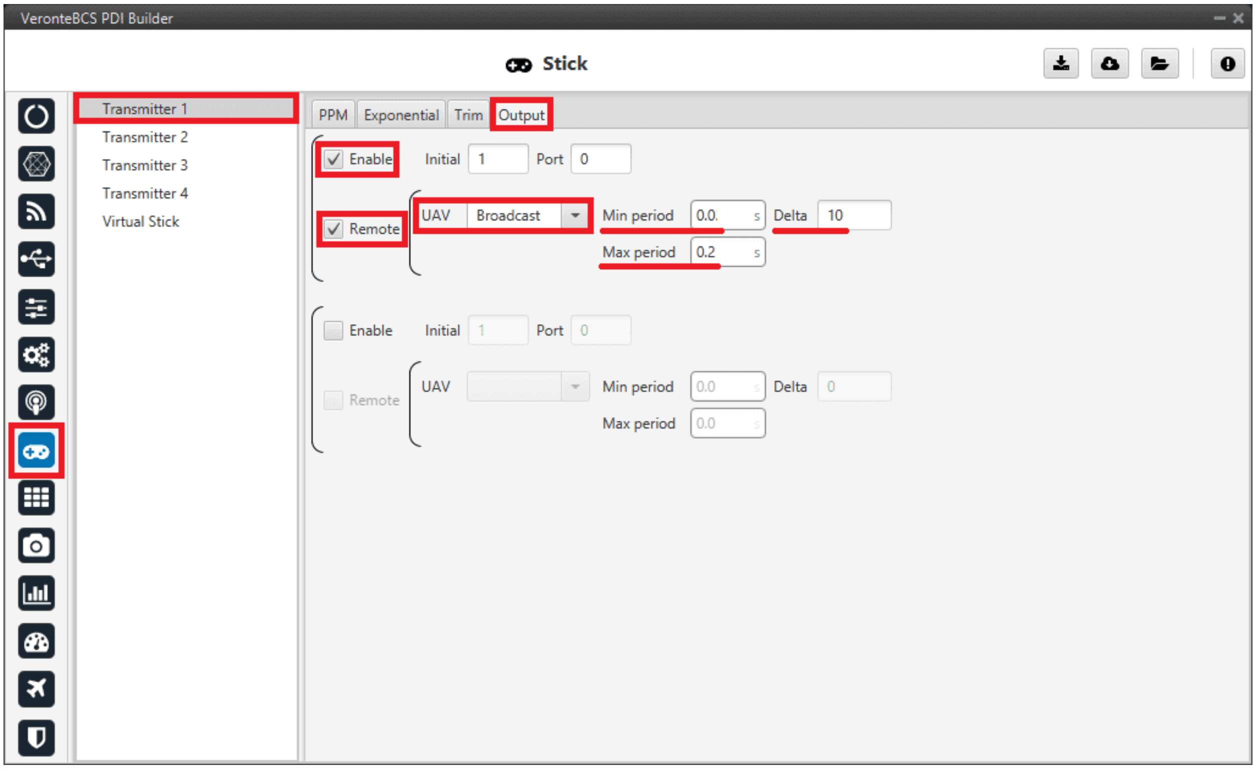
Min delta 0.0 | Max delta 1000.0

Min delta alpha 1.0 | Max delta alpha 0.02

Fig. 41: **Stick - PPM configuration**

4. Go to Stick menu → Transmitter 1 section → **Output tab**.

   Click on **Enable** and on **Remote** to send the stick information to the air unit. Please check the recommended values for the configurable parameters described in the *Ouput tab* of the Stick section.



☑ Enable Initial 1 Port 0

☑ Remote UAV Broadcast Min period 0.0 s Delta 10

Max period 0.2 s

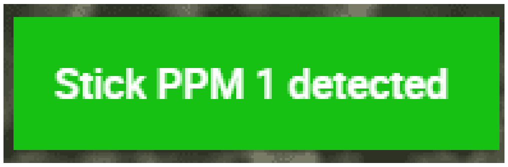
□ Enable Initial 1 Port 0

□ Remote UAV Min period 0.0 s Delta 0

Max period 0.0 s

Fig. 42: **Stick - Output configuration**

If all these settings are correct, users can check that **'Stick PPM 1 not detected'** variable of the **BCS unit** will be true.



Stick PPM 1 detected

Fig. 43: **Stick PPM 1 not detected variable - True**

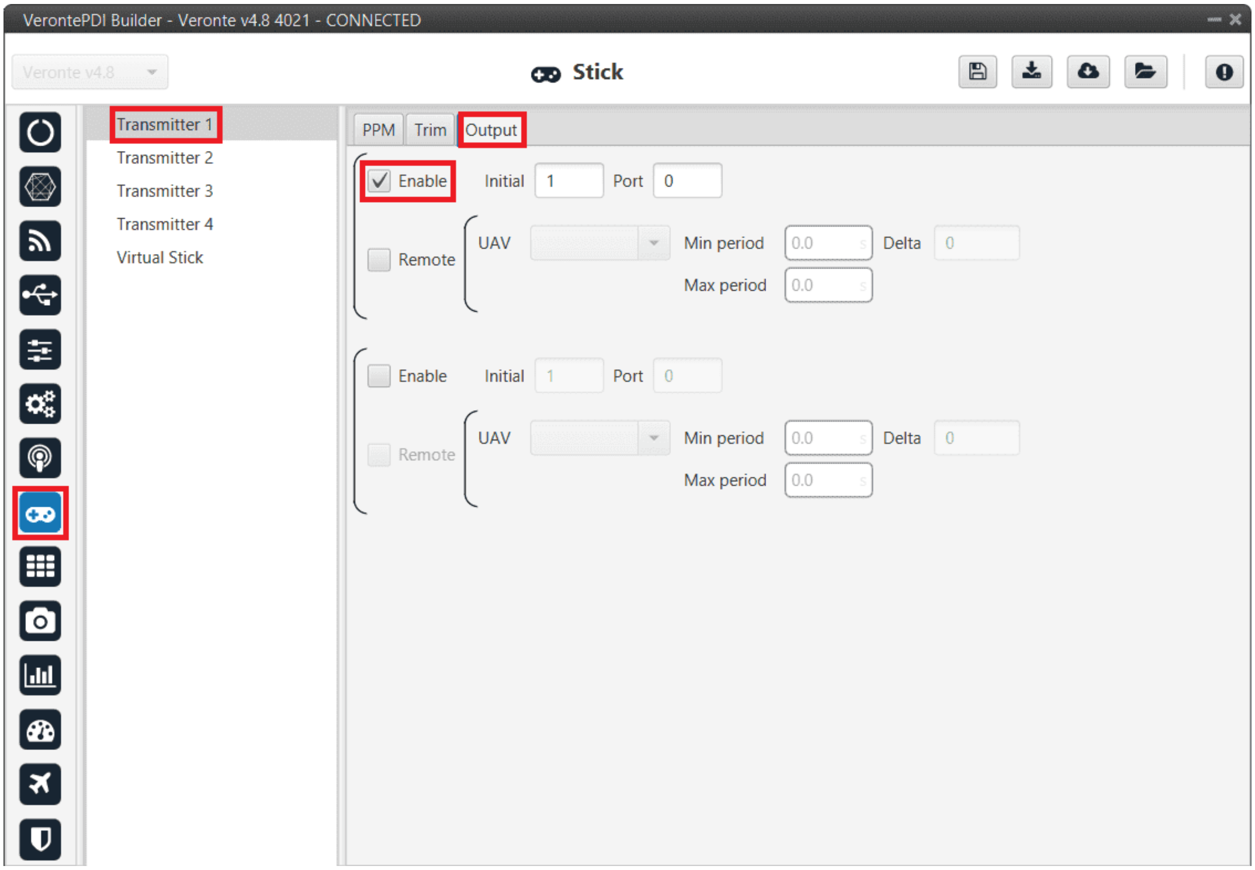
1x air unit (1x PDI Builder configuration)

1. Go to Stick menu → Transmitter 1 section → **PPM tab**.

   Select the brand of transmitter that applies (make the same configuration as the BCS unit).

2. Go to Stick menu → Transmitter 1 section → **Output tab**.

   Just click on **Enable**.



☑ Enable Initial 1 Port 0

□ Remote UAV Min period 0.0 s Delta 0 Max period 0.0 s

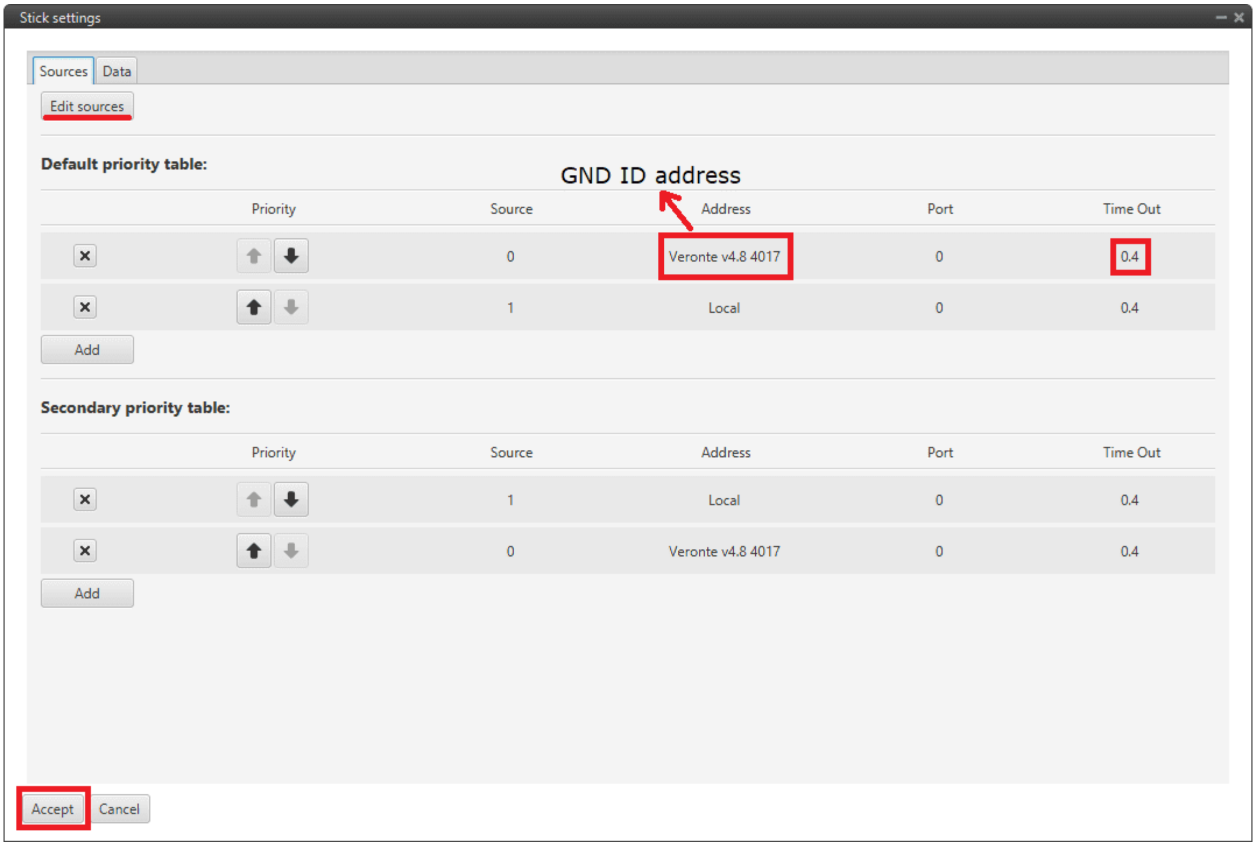
□ Enable Initial 1 Port 0

□ Remote UAV Min period 0.0 s Delta 0 Max period 0.0 s

Fig. 44: **Stick - Output configuration**

3. Go to Block Programs menu → **Stick program** → Double click on the **Stick block** → **Edit sources**.

   Input the **ground unit address** to receive the stick information from that source and put it as the **highest priority** in the priority table. We recommend a Time Out of **0.4 s**.



Default priority table:

	Priority	Source	Address	Port	Time Out
×	↑ ↓	0	Veronte v4.8 4017	0	0.4
×	↑ ↓	1	Local	0	0.4

Secondary priority table:

	Priority	Source	Address	Port	Time Out
×	↑ ↓	1	Local	0	0.4
×	↑ ↓	0	Veronte v4.8 4017	0	0.4

Fig. 45: **Stick block configuration**

Then, if all is correct, users can check that **‘Stick not detected’** variable of the **1x air unit** will be true.

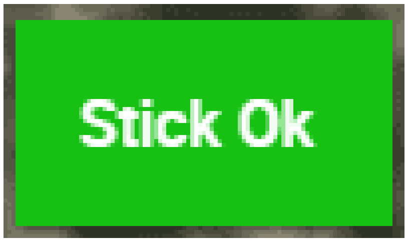


Fig. 46: **Stick not detected variable - True**

And that means that the communication between the BCS and the 1x air unit is correctly configured.

3.5.4.1.2 Simulation case (HIL)

In this case, the user is only using one **Veronte BCS**.

So users will have to follow **steps 1**, **2** and **3** explained above for the **BCS unit**, but also **steps 2** and **3** of the **1x air unit** configuration. However, instead of entering the **BCS unit address**, select the **Local** option.

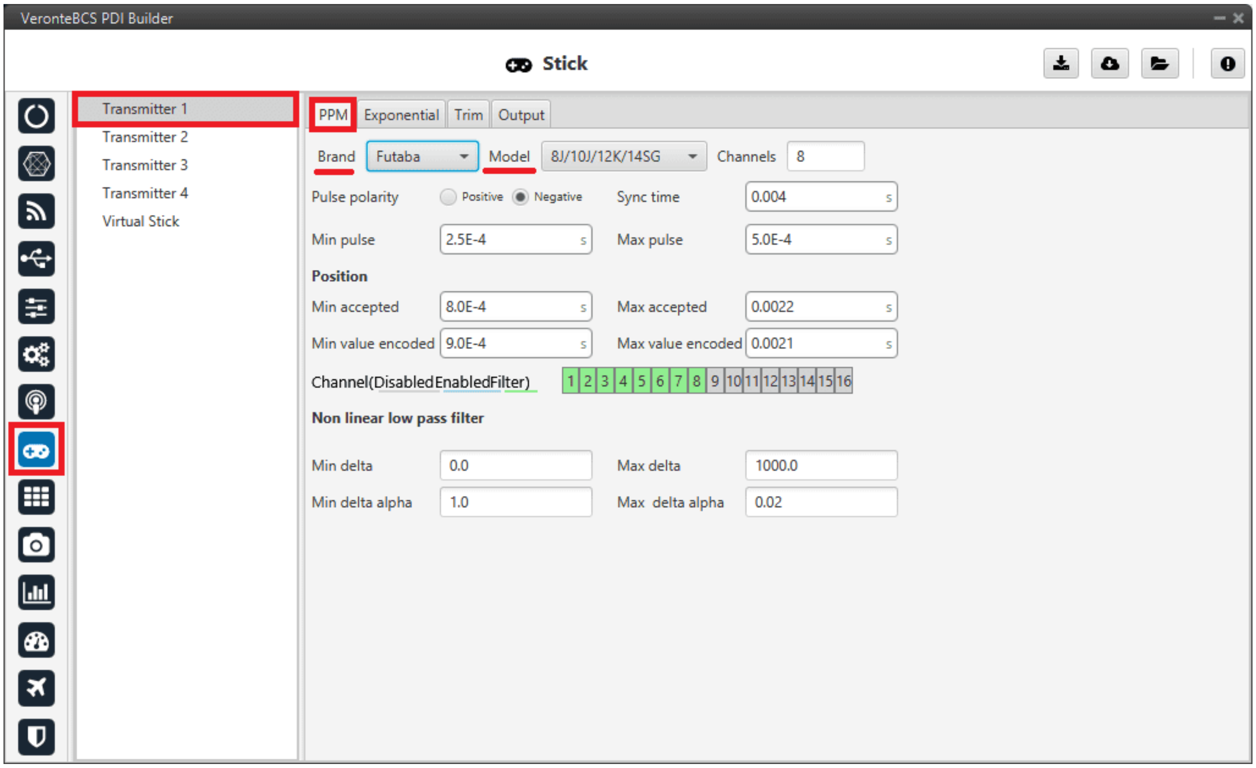
3.5.4.1.3 On-board PPM receiver case

In that case, follow the next steps:

BCS unit (BCS PDI Builder configuration)

1. Go to Stick menu → Transmitter 1 section → **PPM tab**.

   Select the brand of transmitter that applies.



Brand: Futaba; Model: 8J/10J/12K/14SG; Channels: 8

Pulse polarity: ○ Positive ● Negative; Sync time: 0.004 s

Min pulse: 2.5E-4 s; Max pulse: 5.0E-4 s

Position

Min accepted: 8.0E-4 s; Max accepted: 0.0022 s

Min value encoded: 9.0E-4 s; Max value encoded: 0.0021 s

Channel(DisabledEnabledFilter): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Non linear low pass filter

Min delta: 0.0; Max delta: 1000.0

Min delta alpha: 1.0; Max delta alpha: 0.02

Fig. 47: **Stick - PPM configuration**

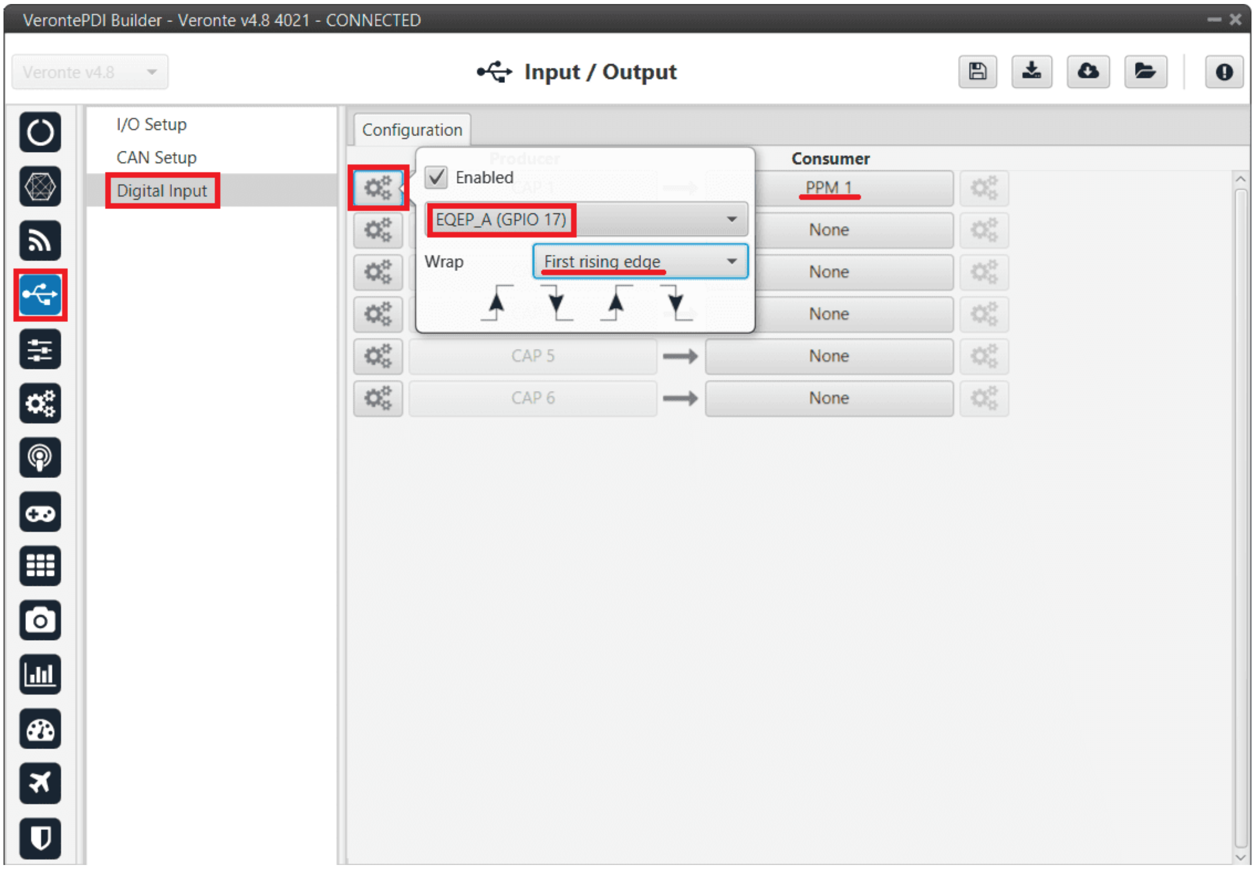
1x air unit (1x PDI Builder)

1. Go to Input/Output menu → **Digital Input section**.

   Make sure that the following parameters have been configured:

   - Producer: **CAP 1**
     - Enabled
     - Select the pin to which the transmitter is connected (normally *EQEP A (i.e., GPIO 17)*)
     - Edge detection: First rising edge

- Consumer: **PPM 1**



☑ Enabled

EQEP_A (GPIO 17)

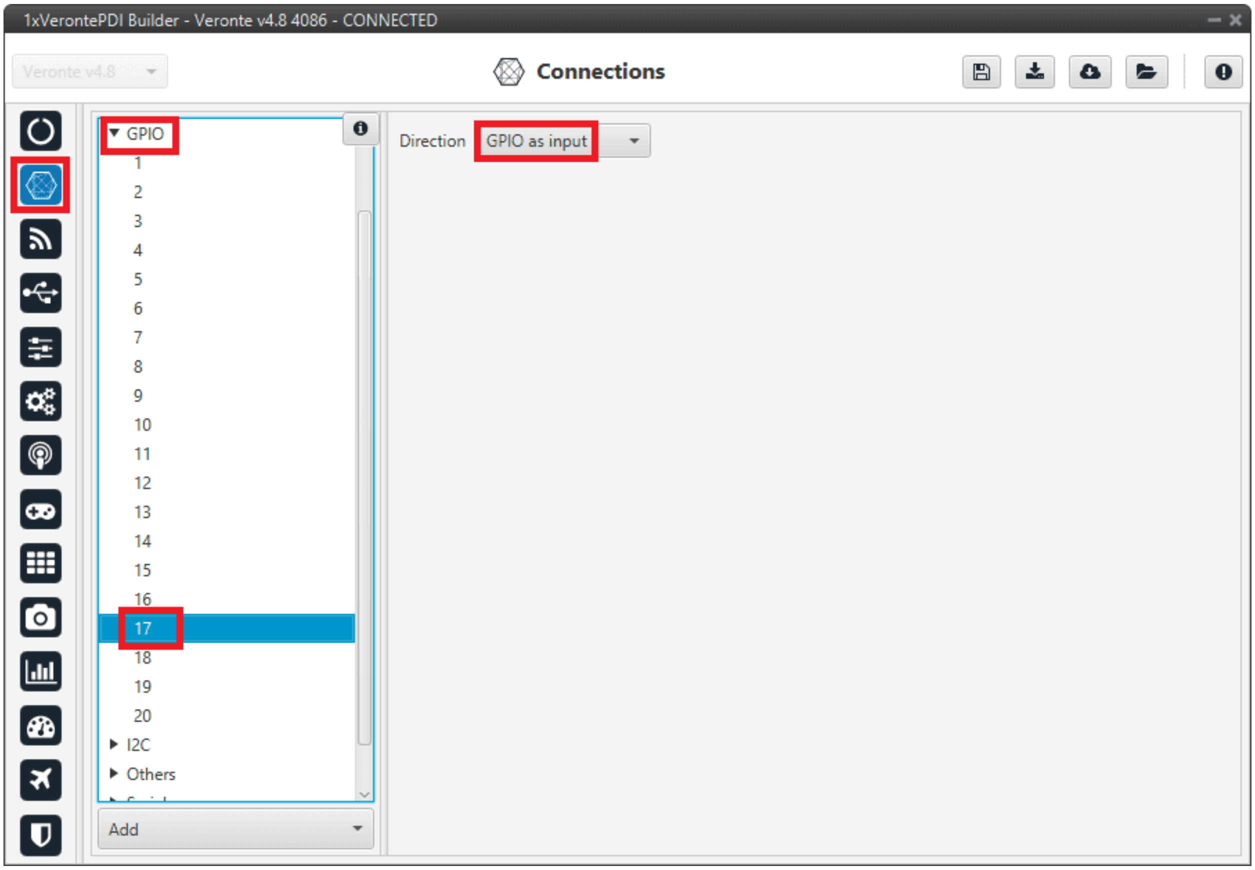
Wrap: First rising edge

Producer		Consumer
CAP 1	→	PPM 1
		None
		None
		None
CAP 5	→	None
CAP 6	→	None

Fig. 48: **Stick - Digital Input configuration**

2. Go to Connections menu → **GPIO section**.

   Verify that the pin to which the transmitter is connected, in this case *GPIO 17 (i.e., EQEP A)*, is set as input.

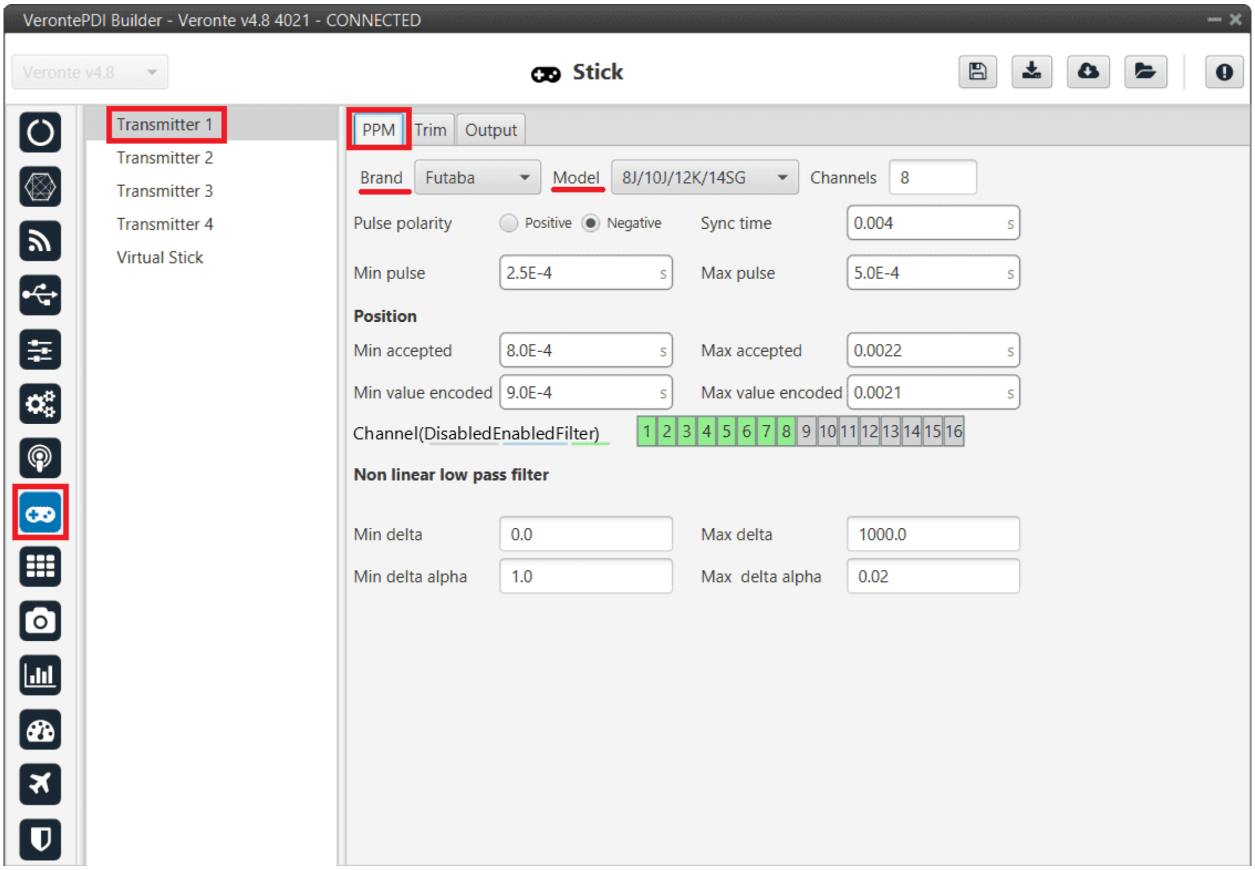


Direction GPIO as input

Fig. 49: **Stick - GPIO/EQEP configuration**

3. Go to Stick menu → Transmitter 1 section → **PPM tab**.

   Select the brand of transmitter that applies.



Brand: Futaba
Model: 8J/10J/12K/14SG
Channels: 8

Pulse polarity: ○ Positive ● Negative
Sync time: 0.004 s

Min pulse: 2.5E-4 s
Max pulse: 5.0E-4 s

Position

Min accepted: 8.0E-4 s
Max accepted: 0.0022 s

Min value encoded: 9.0E-4 s
Max value encoded: 0.0021 s

Channel(DisabledEnabledFilter): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Non linear low pass filter

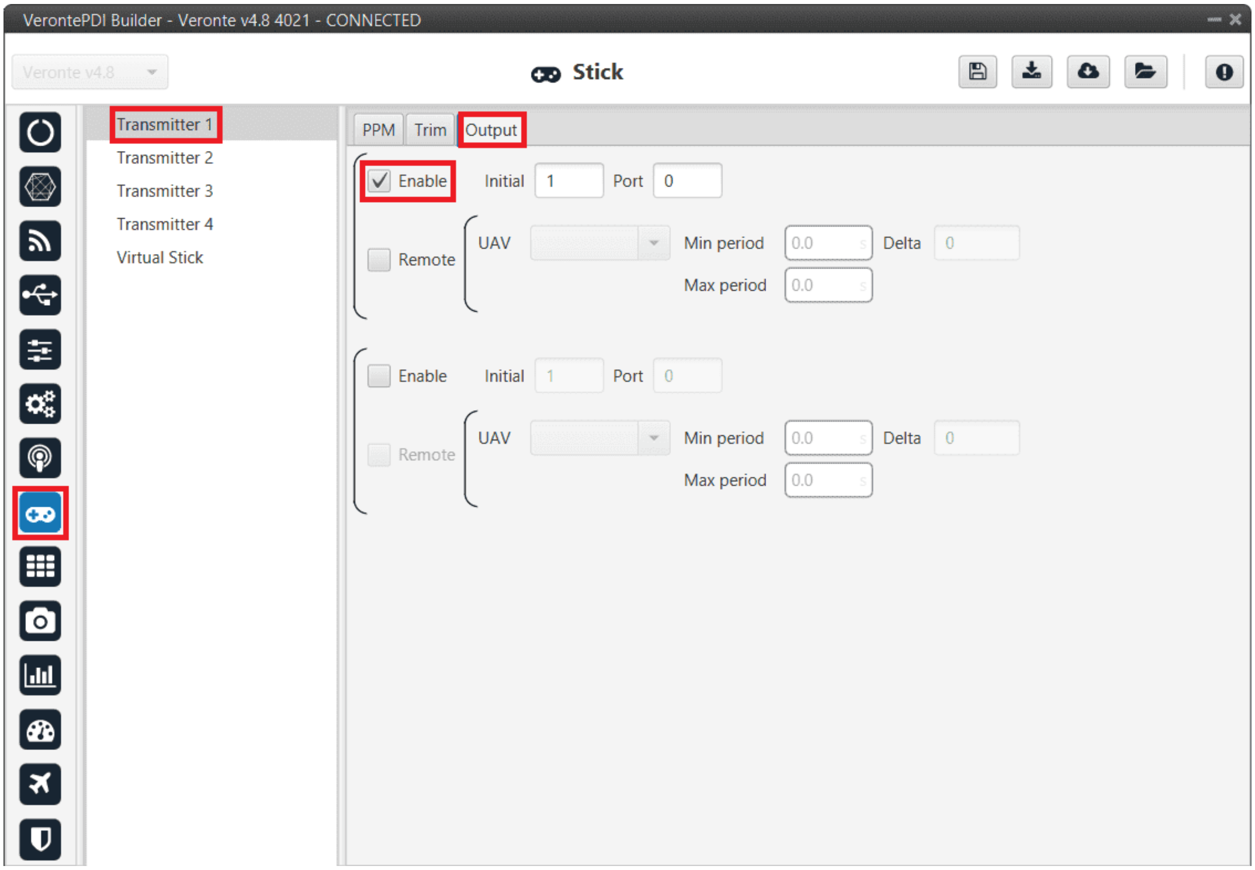
Min delta: 0.0
Max delta: 1000.0

Min delta alpha: 1.0
Max delta alpha: 0.02

Fig. 50: **Stick - PPM configuration**

4. Go to Stick menu → Transmitter 1 section → **Output tab**.

   Just click on **Enable**.



☑ Enable Initial 1 Port 0

□ Remote UAV Min period 0.0 s Delta 0

Max period 0.0 s

□ Enable Initial 1 Port 0

□ Remote UAV Min period 0.0 s Delta 0

Max period 0.0 s

Fig. 51: **Stick - Output configuration**

If all these settings are correct, users can check that **'Stick PPM 1 not detected'** variable of the **1x air unit** will be true.

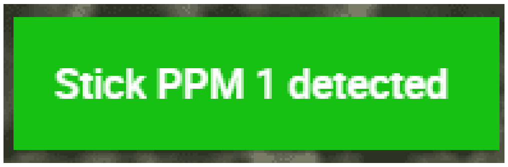
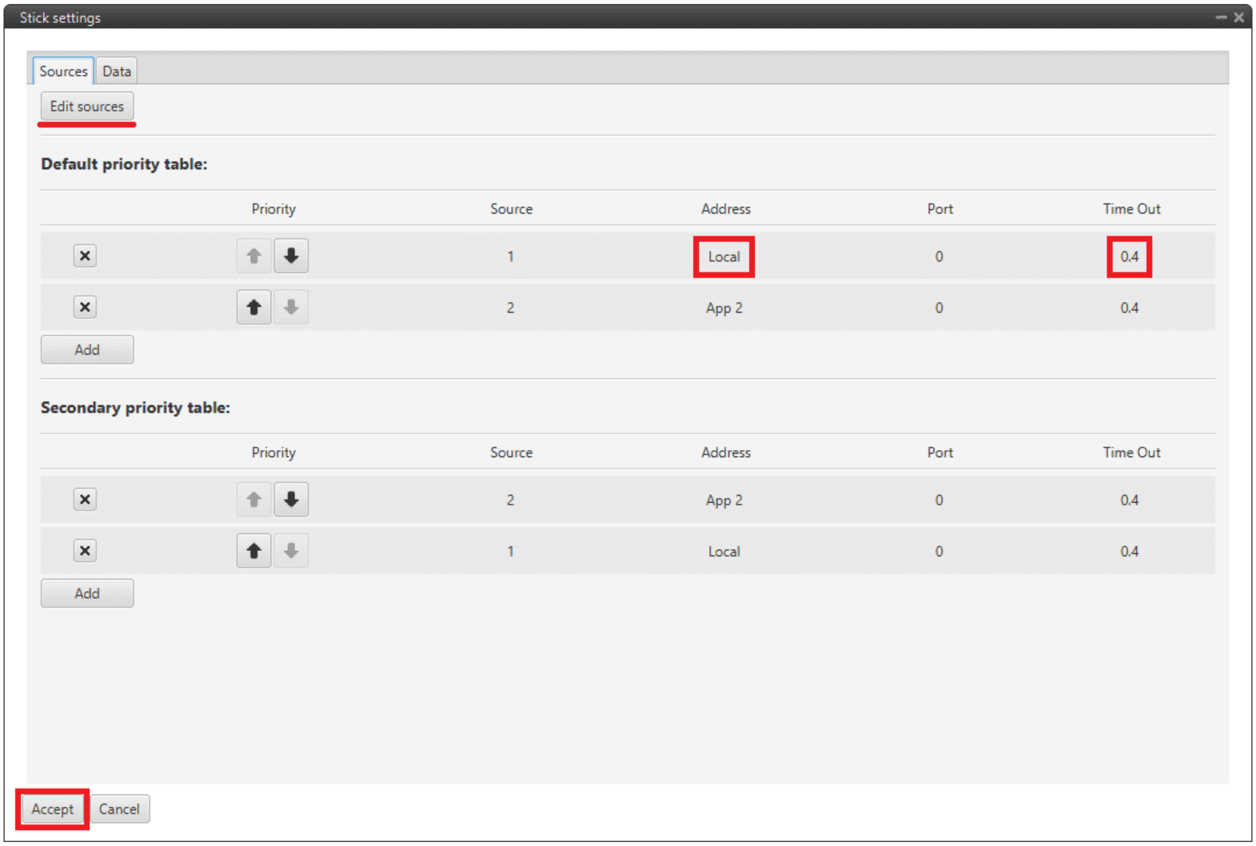


Fig. 52: **Stick PPM 1 not detected variable - True**

5. Go to Block Programs menu → **Stick program** → Double click on the **Stick block** → **Edit sources**.

   Input the address as **Local** to receive the stick information from that source and put it as the **highest priority** in the priority table. We recommend a Time Out of **0.4 s**.



Default priority table:

	Priority	Source	Address	Port	Time Out
✕	↑ ↓	1	Local	0	0.4
✕	↑ ↓	2	App 2	0	0.4

Secondary priority table:

	Priority	Source	Address	Port	Time Out
✕	↑ ↓	2	App 2	0	0.4
✕	↑ ↓	1	Local	0	0.4

Fig. 53: **Stick block configuration**

Then, if all is correct, users can check that **'Stick not detected'** variable of the **1x air unit** will be true.

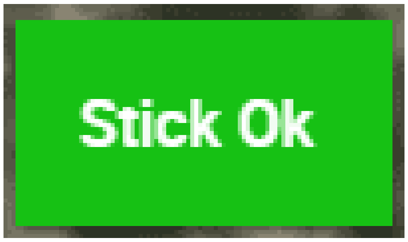

Stick Ok

Fig. 54: **Stick not detected variable - True**

And that means that the communication between the BCS and the 1x air unit is correctly configured.

3.5.4.2 USB joystick

Veronte software is able to detect USB devices such as joysticks. The axis of these devices can be read and configured to send stick information to **Veronte Autopilot 1x**.

To configure them:

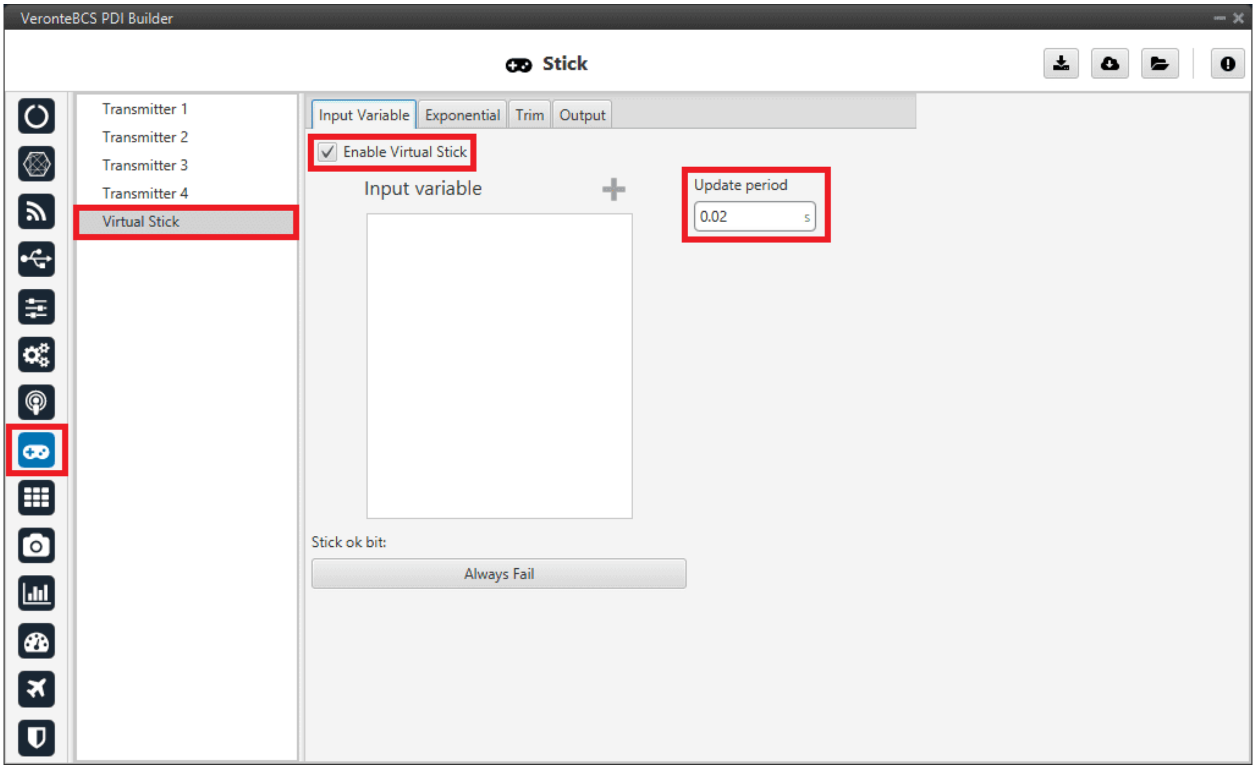
1. Connect the USB joystick to the computer.
2. Configure a Virtual Stick as explained in *Virtual Stick Integration*.

3.5.4.3 Virtual Stick

To configure a virtual stick, follow the next steps:

1. Go to Stick menu → Virtual Stick section → **Input variable tab**.

   Enable the virtual stick and enter an **update period** (we recommend **0.02 s**).



☑ Enable Virtual Stick

Update period 0.02 s

Fig. 55: **Virtual Stick configuration**

2. Go to Stick menu → Virtual Stick section → **Output tab**.

   Just click on **Enable**.

3. Go to Block Programs menu → **Stick program** → Double click on the **Stick block** → **Edit sources**.

   Input **App 2** to receive the stick information from the virtual stick widget and put it as the **highest priority** in the priority table. We recommend a Time Out of **0.4 s**.

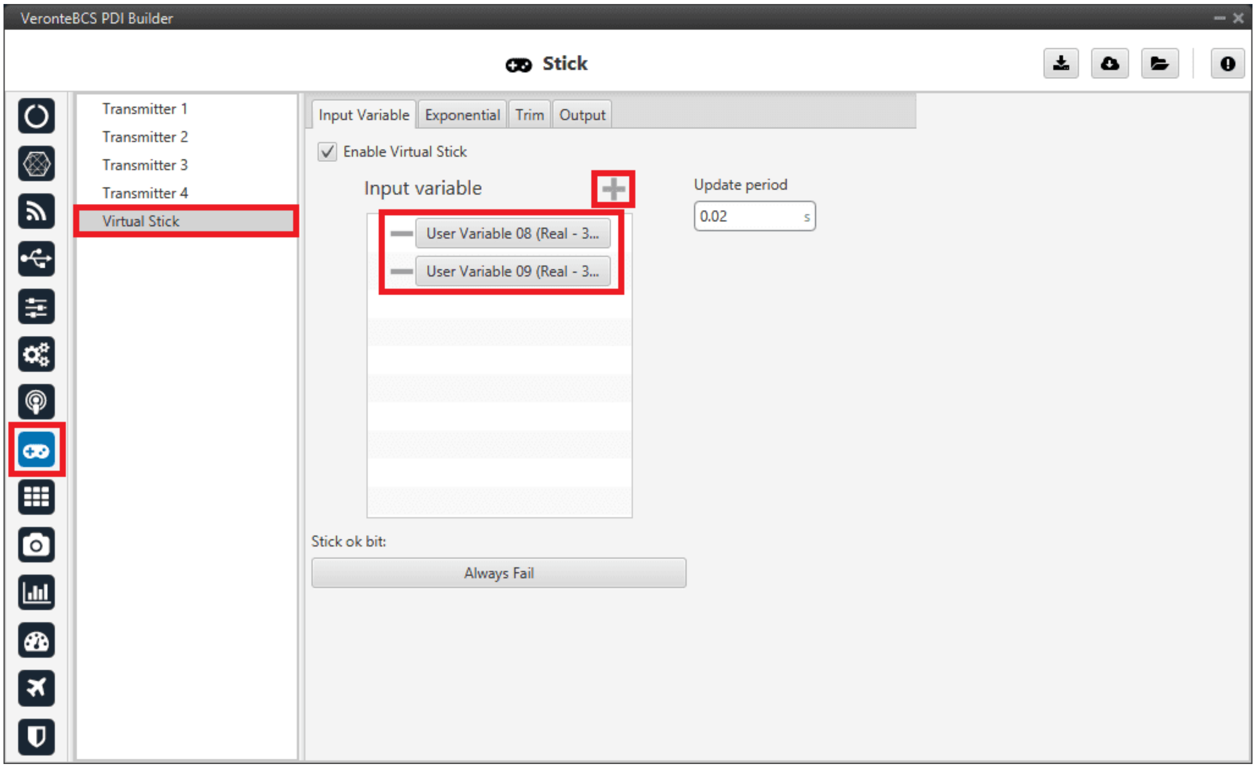
4. Configure a **Virtual Stick Widget**.

Please find an example of how to configure it in Virtual stick widget in the Integration examples section of the **Veronte Ops** manual.

If the user creates a virtual stick to process the information received through a different channel than PPM (e.g., by CAN or ADC), the user will also have to:

- Go to Stick menu → Virtual Stick section → **Input variable tab**.

  Add the variables containing the stick information in **Input Variable**.



☑ Enable Virtual Stick

Input variable: User Variable 08 (Real - 3... / User Variable 09 (Real - 3...

Update period: 0.02 s

Stick ok bit: Always Fail

Fig. 56: **Virtual Stick with Input variables configuration**

3.5.5 Veronte products

3.5.5.1 CEX/MEX

As it is sometimes not possible to connect a CEX/MEX directly to the PC in order to configure it (access CEX/MEX PDI Builder), the **BCS** is connected to the computer and a connection is made between **CEX/MEX and BCS via CAN**.

To be able to communicate with CEX/MEX via CAN, the following connection is necessary:

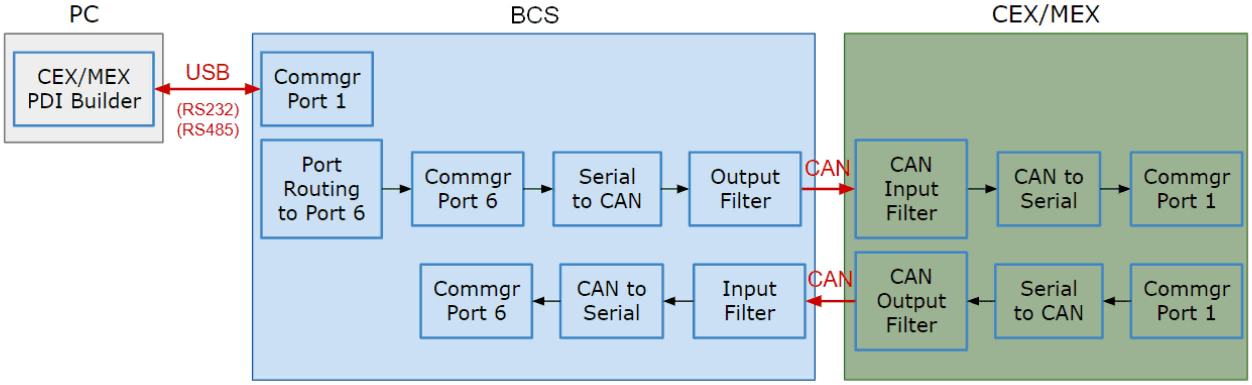


Fig. 57: **Communication diagram BCS - CEX/MEX**

Note:

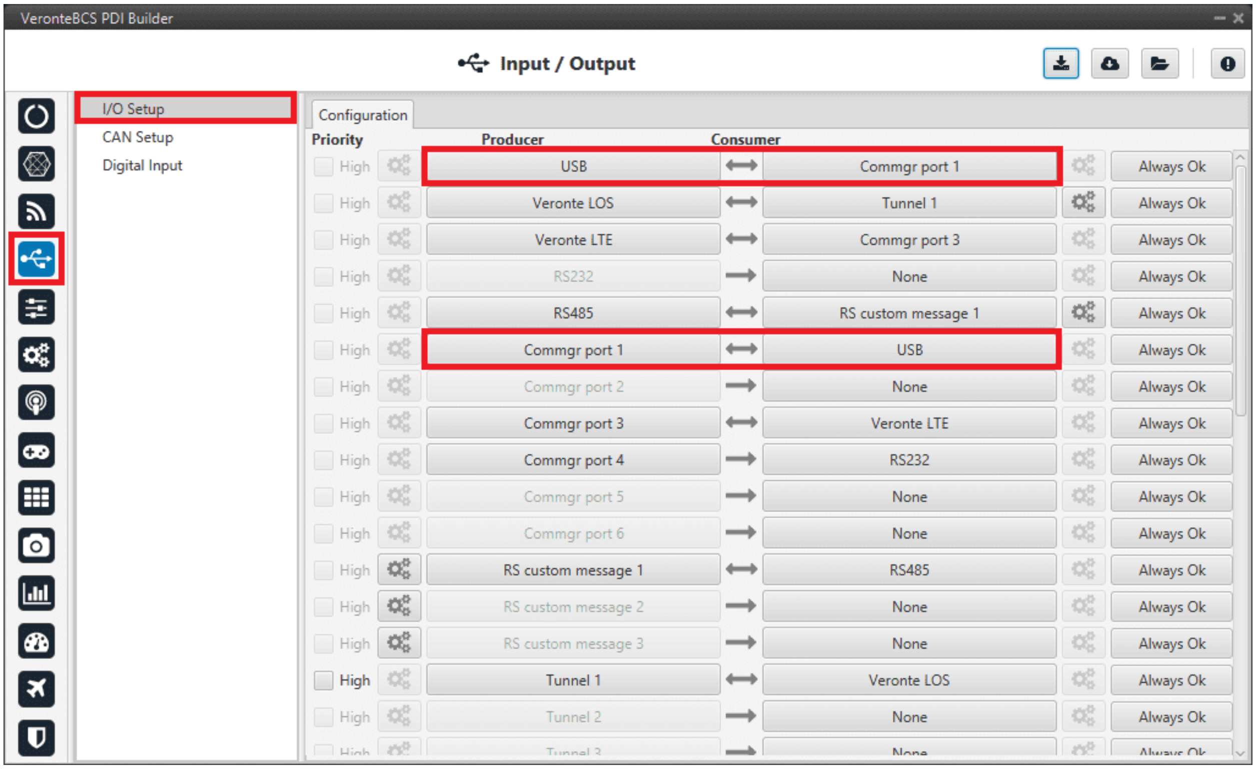
- **BCS** usually has this configuration by default, but check it out.
- As the steps to be performed in **CEX PDI Builder and MEX PDI Builder is exactly the same**, **only the steps for one of them will be detailed**. The interface may differ slightly, but the configuration is the same.

Follow the steps below to make this configuration:

BCS PDI Builder side

1. Go to Input/Output menu → **I/O Setup section**.

   Check the connection between the computer and the **BCS** (usually via USB, but RS232 and RS485 are also possible).



Priority	Producer		Consumer	
□ High	USB	⟷	Commgr port 1	Always Ok
□ High	Veronte LOS	⟷	Tunnel 1	Always Ok
□ High	Veronte LTE	⟷	Commgr port 3	Always Ok
□ High	RS232	→	None	Always Ok
□ High	RS485	⟷	RS custom message 1	Always Ok
□ High	Commgr port 1	⟷	USB	Always Ok
□ High	Commgr port 2	→	None	Always Ok
□ High	Commgr port 3	⟷	Veronte LTE	Always Ok
□ High	Commgr port 4	→	RS232	Always Ok
□ High	Commgr port 5	→	None	Always Ok
□ High	Commgr port 6	→	None	Always Ok
□ High	RS custom message 1	⟷	RS485	Always Ok
□ High	RS custom message 2	→	None	Always Ok
□ High	RS custom message 3	→	None	Always Ok
□ High	Tunnel 1	⟷	Veronte LOS	Always Ok
□ High	Tunnel 2	→	None	Always Ok
□ High	Tunnel 3	→	None	Always Ok

Fig. 58: **BCS - USB communication**

2. Go to Communications menu → **Ports section**.

   Remove Port 6 from the Forward group and **Add Port 6 to the Route group**, with target CEX's Address:

   ⇒ **Address = 44000 + Serial number**.

   The CEX address must be in the **range 45000 - 49999**.

Note:

- For **MEX**, the address should look like this:
  - **Address = 42000 + Serial number**.
  - The MEX address must be in the **range 43000 - 43999**.
- If the theorical address does not work, 999 (unknown) can be used as sometimes the address has not been set in CEX/MEX.

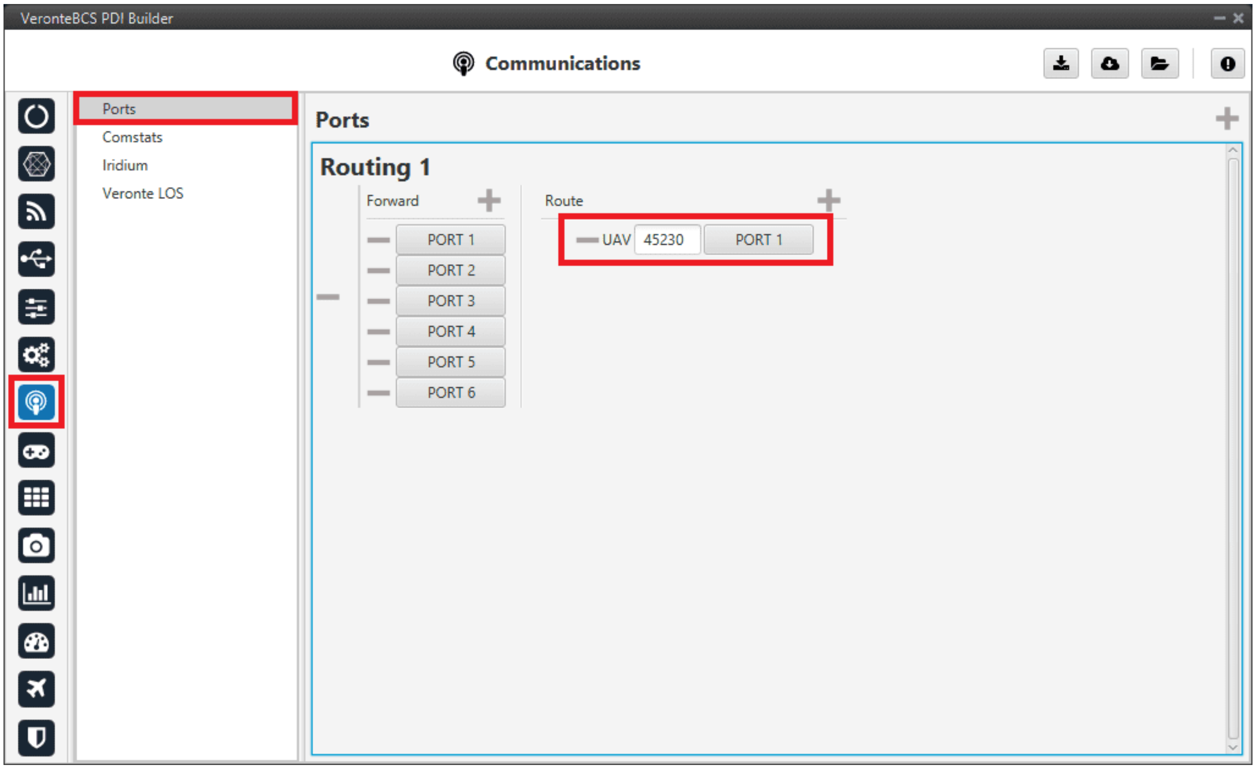
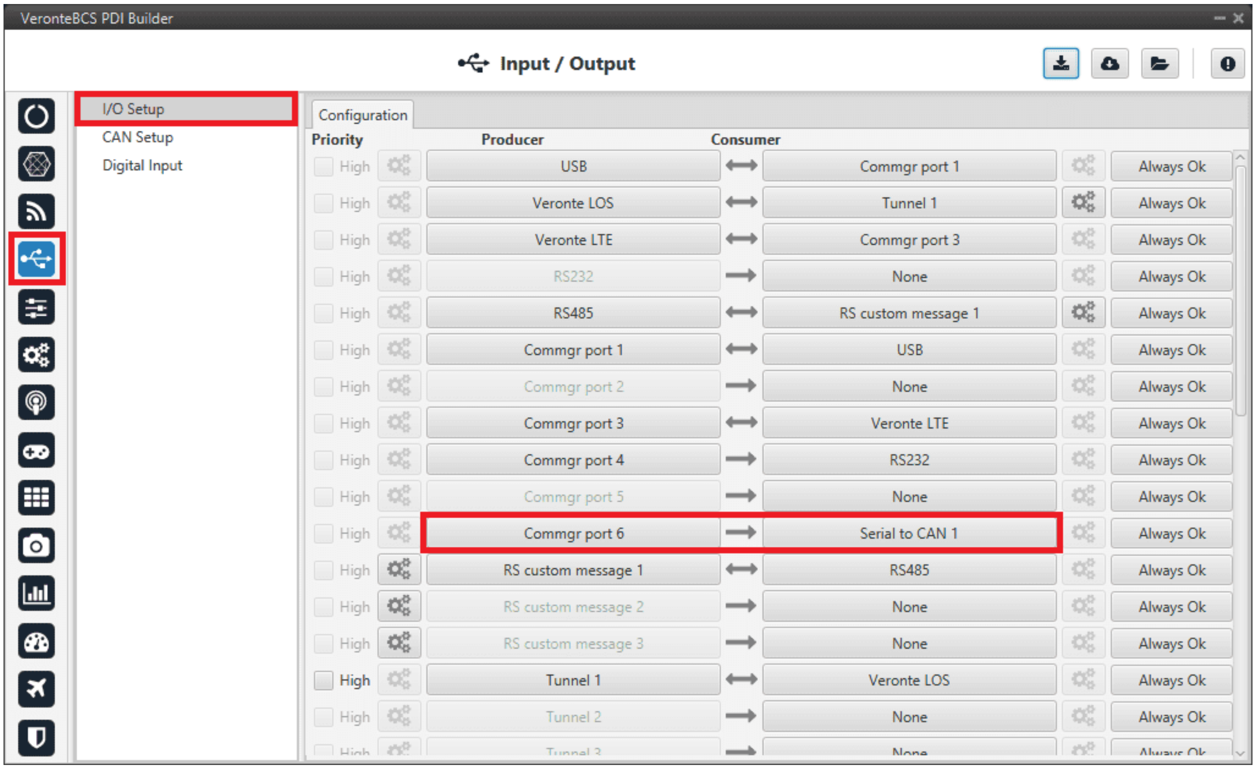


Fig. 59: **BCS - Routing**

3. Go to Input/Output menu → **I/O Setup section**.

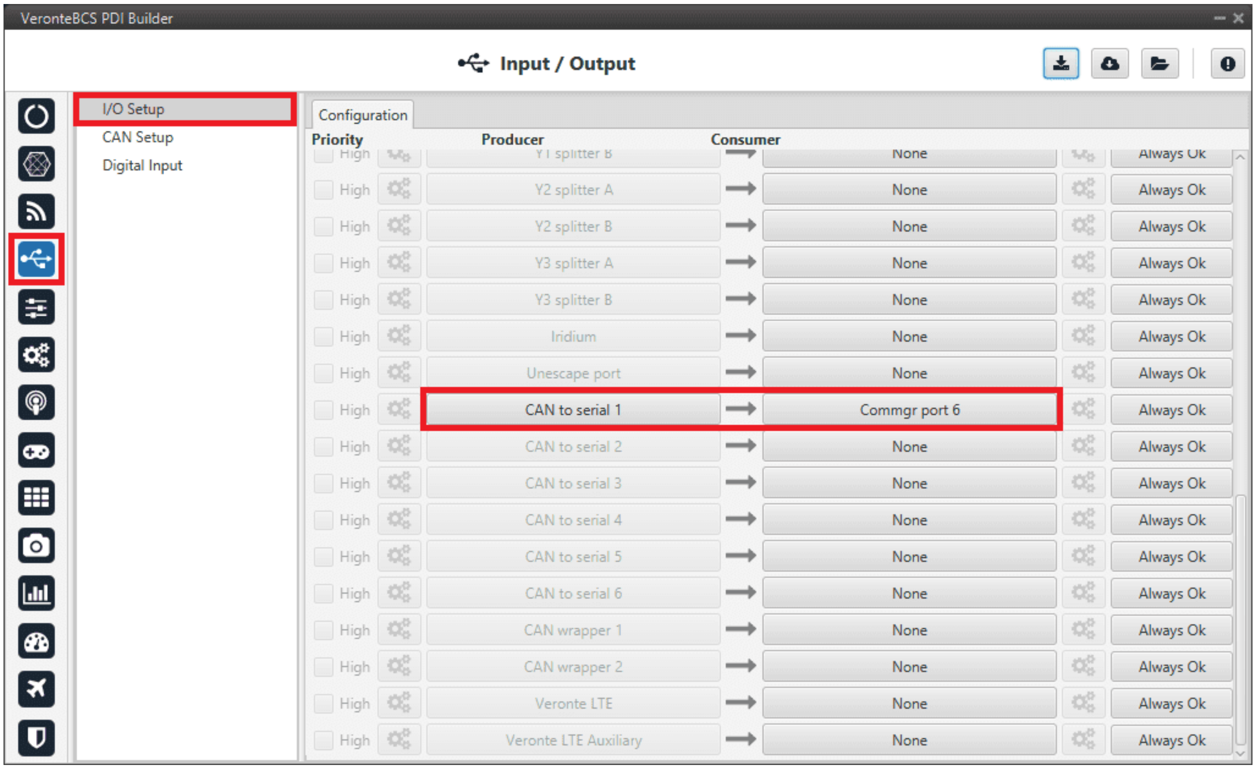
   Connect **Commgr Port 6** to **Serial to CAN 1** consumer:



Priority	Producer		Consumer	
☐ High	USB	⟷	Commgr port 1	Always Ok
☐ High	Veronte LOS	⟷	Tunnel 1	Always Ok
☐ High	Veronte LTE	⟷	Commgr port 3	Always Ok
☐ High	RS232	→	None	Always Ok
☐ High	RS485	⟷	RS custom message 1	Always Ok
☐ High	Commgr port 1	⟷	USB	Always Ok
☐ High	Commgr port 2	→	None	Always Ok
☐ High	Commgr port 3	⟷	Veronte LTE	Always Ok
☐ High	Commgr port 4	→	RS232	Always Ok
☐ High	Commgr port 5	→	None	Always Ok
☐ High	Commgr port 6	→	Serial to CAN 1	Always Ok
☐ High	RS custom message 1	⟷	RS485	Always Ok
☐ High	RS custom message 2	→	None	Always Ok
☐ High	RS custom message 3	→	None	Always Ok
☐ High	Tunnel 1	⟷	Veronte LOS	Always Ok
☐ High	Tunnel 2	→	None	Always Ok
☐ High	Tunnel 3	→	None	Always Ok

Fig. 60: **BCS - I/O Setup - Serial to CAN**

Then, connect **CAN to Serial 1** to **Commgr Port 6**:



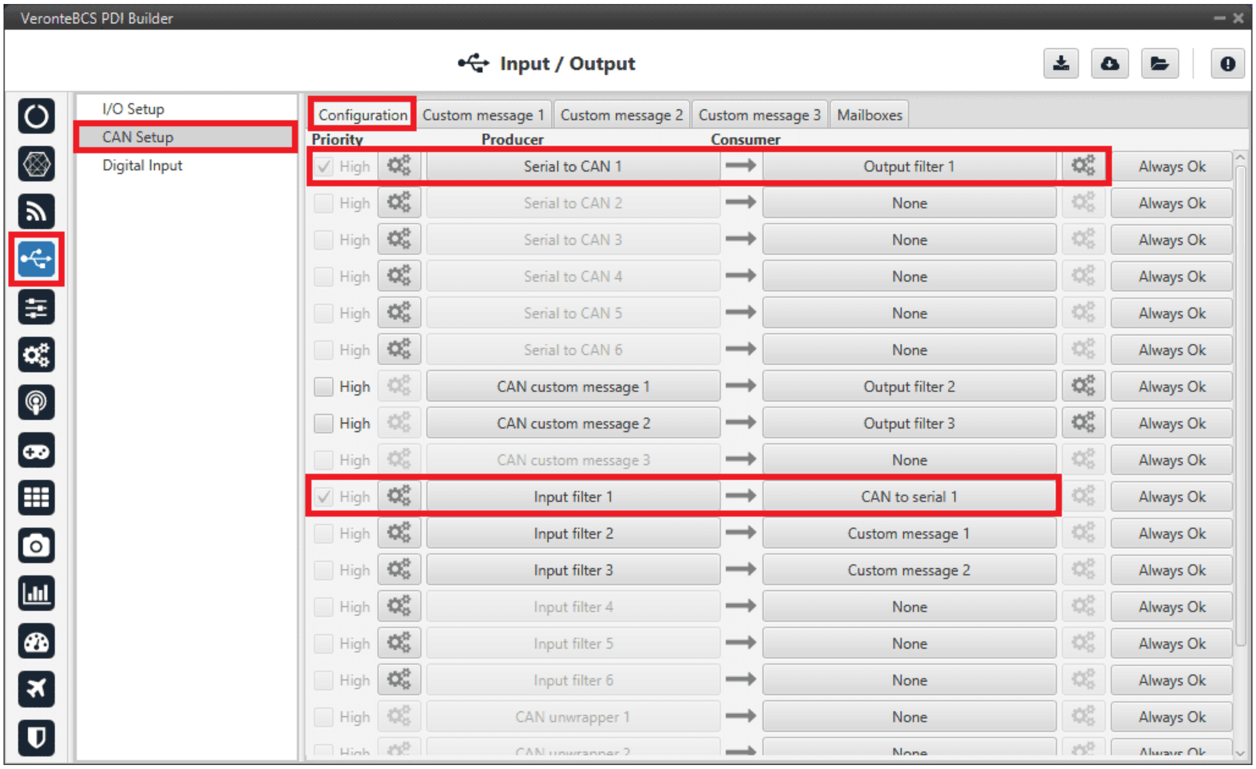
Priority	Producer		Consumer	
High	Y1 splitter B	→	None	Always Ok
High	Y2 splitter A	→	None	Always Ok
High	Y2 splitter B	→	None	Always Ok
High	Y3 splitter A	→	None	Always Ok
High	Y3 splitter B	→	None	Always Ok
High	Iridium	→	None	Always Ok
High	Unescape port	→	None	Always Ok
High	CAN to serial 1	→	Commgr port 6	Always Ok
High	CAN to serial 2	→	None	Always Ok
High	CAN to serial 3	→	None	Always Ok
High	CAN to serial 4	→	None	Always Ok
High	CAN to serial 5	→	None	Always Ok
High	CAN to serial 6	→	None	Always Ok
High	CAN wrapper 1	→	None	Always Ok
High	CAN wrapper 2	→	None	Always Ok
High	Veronte LTE	→	None	Always Ok
High	Veronte LTE Auxiliary	→	None	Always Ok

Fig. 61: **BCS - I/O Setup - CAN to Serial**

4. Go to Input/Output menu → CAN Setup section → **Configuration tab**.

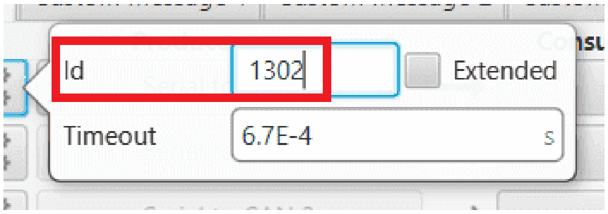
   Connect a **Serial to CAN** with the right Id (**CAN ID 1302**) to an **Output filter**.

   In addition, connect an **Input filter** with the right Id (**CAN ID 1301**) to a **CAN to Serial**:



Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
□ High	Serial to CAN 2	→	None	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message 1	→	Output filter 2	Always Ok
□ High	CAN custom message 2	→	Output filter 3	Always Ok
□ High	CAN custom message 3	→	None	Always Ok
☑ High	Input filter 1	→	CAN to serial 1	Always Ok
□ High	Input filter 2	→	Custom message 1	Always Ok
□ High	Input filter 3	→	Custom message 2	Always Ok
□ High	Input filter 4	→	None	Always Ok
□ High	Input filter 5	→	None	Always Ok
□ High	Input filter 6	→	None	Always Ok
□ High	CAN unwrapper 1	→	None	Always Ok
□ High	CAN unwrapper 2	→	None	Always Ok

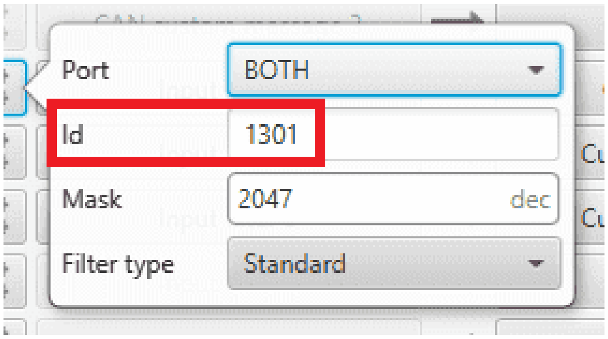
Fig. 62: **BCS - CAN Setup**



Id: 1302 □ Extended

Timeout: 6.7E-4 s

Fig. 63: **BCS - CAN Setup - Serial to CAN configuration**



Port: BOTH

Id: 1301

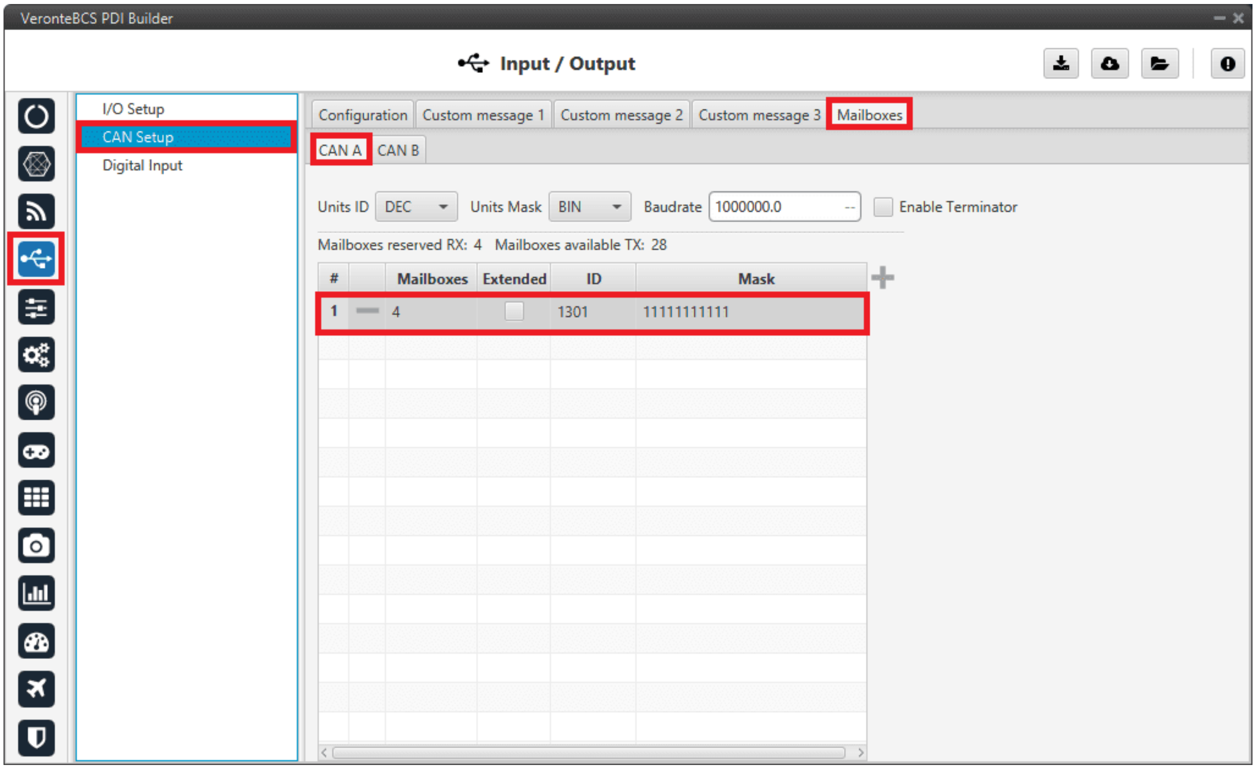
Mask: 2047 dec

Filter type: Standard

Fig. 64: **BCS - CAN Setup - Input filter configuration**

5. Go to Input/Output menu → CAN Setup section → **Mailboxes tab**.

   Finally, configure the reception **mailbox with ID 1301**, assign at least 4 mailboxes:



Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

Mailboxes reserved RX: 4 Mailboxes available TX: 28

#	Mailboxes	Extended	ID	Mask
1	4	□	1301	11111111111

Fig. 65: **BCS - Mailboxes configuration**

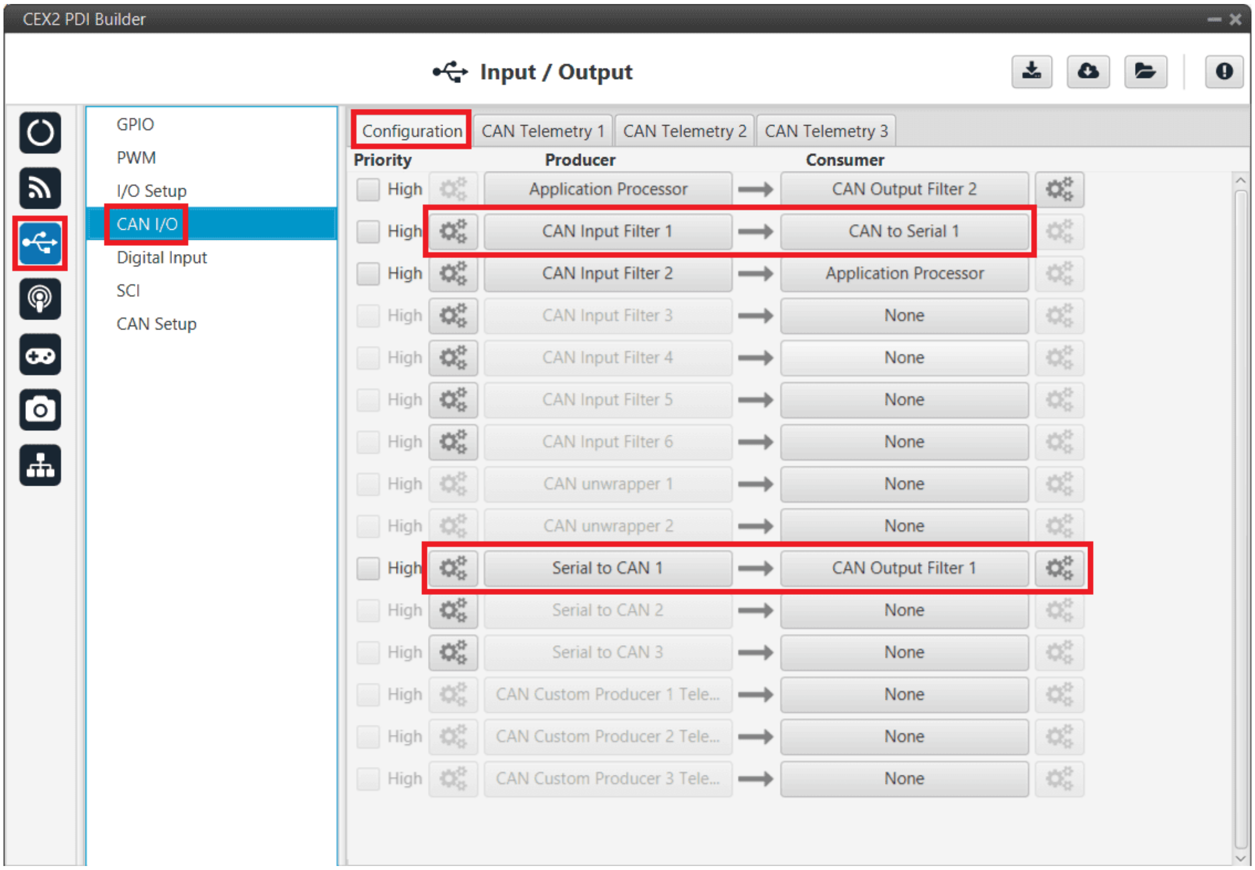
CEX PDI Builder side

Note: This part is already built for CEX default configuration, but the user can check it.

6. Go to Input/Output menu → CAN I/O section → **Configuration tab**.

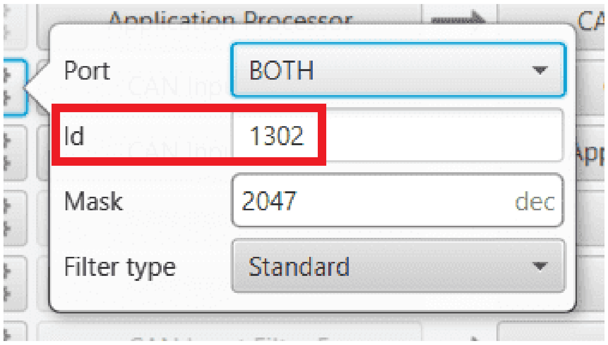
   Connect a **CAN Input filter** with the right CAN Address (**CAN ID 1302**) to **CAN to Serial 1**.

   In addition, connect **Serial to CAN 1** with the right CAN Address (**CAN ID 1301**) to a **CAN Output filter** port:



Priority	Producer		Consumer
□ High	Application Processor	→	CAN Output Filter 2
□ High	CAN Input Filter 1	→	CAN to Serial 1
□ High	CAN Input Filter 2	→	Application Processor
□ High	CAN Input Filter 3	→	None
□ High	CAN Input Filter 4	→	None
□ High	CAN Input Filter 5	→	None
□ High	CAN Input Filter 6	→	None
□ High	CAN unwrapper 1	→	None
□ High	CAN unwrapper 2	→	None
□ High	Serial to CAN 1	→	CAN Output Filter 1
□ High	Serial to CAN 2	→	None
□ High	Serial to CAN 3	→	None
□ High	CAN Custom Producer 1 Tele...	→	None
□ High	CAN Custom Producer 2 Tele...	→	None
□ High	CAN Custom Producer 3 Tele...	→	None

Fig. 66: **CEX - CAN I/O**



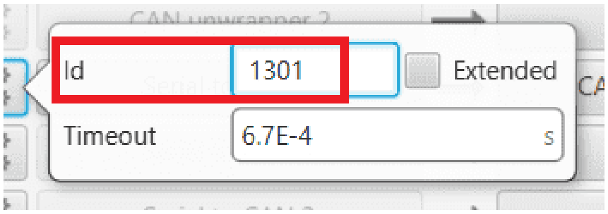
Port: BOTH

Id: 1302

Mask: 2047 dec

Filter type: Standard

Fig. 67: **CEX - CAN I/O - Input filter configuration**



Id: 1301 □ Extended

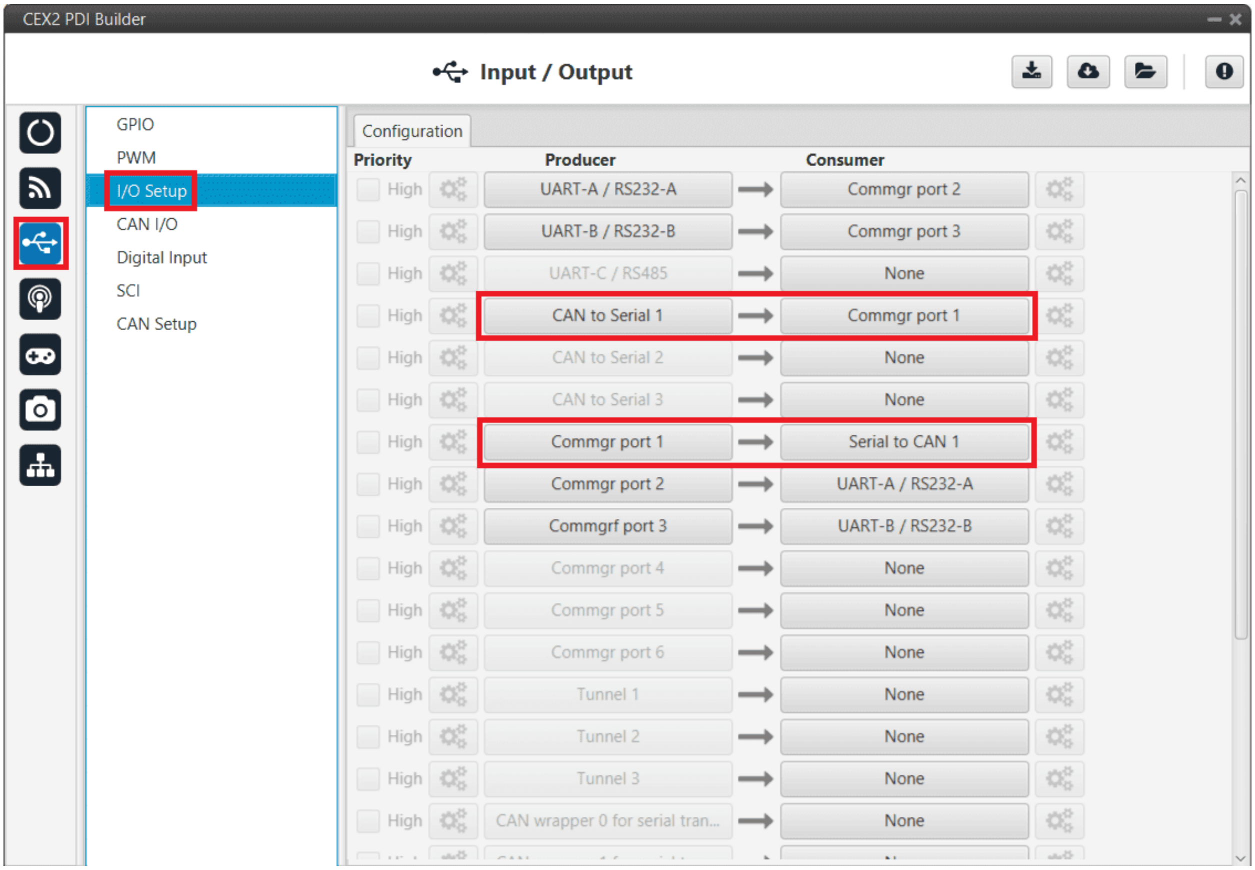
Timeout: 6.7E-4 s

Fig. 68: **CEX - CAN I/O - Serial to CAN configuration**

7. Go to Input/Output menu → **I/O Setup section**.

   Connect **CAN to Serial 1** to any **Commgr Port 1** in CEX.

   In addition, connect **Commgr Port 1** to **Serial to CAN 1** consumer:

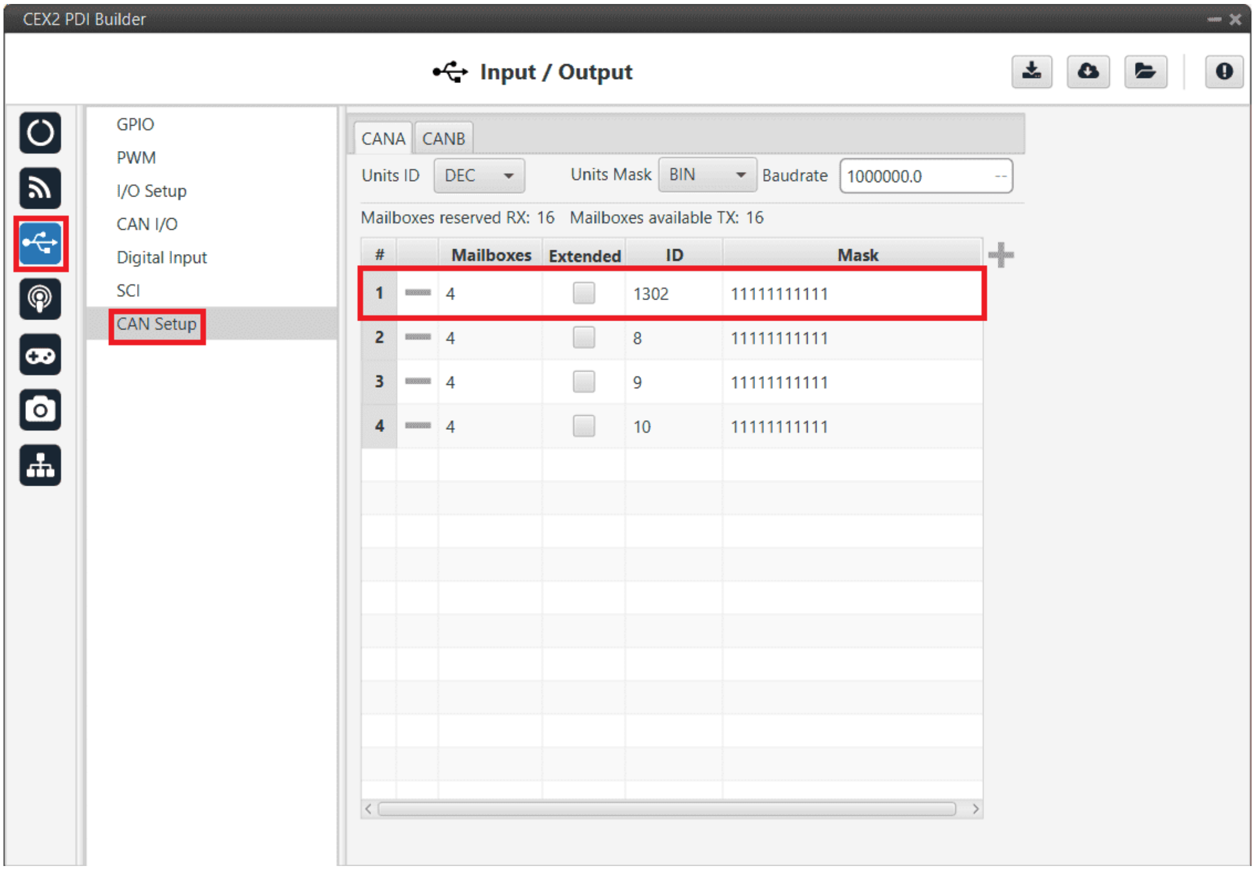


Priority	Producer		Consumer
☐ High	UART-A / RS232-A	→	Commgr port 2
☐ High	UART-B / RS232-B	→	Commgr port 3
☐ High	UART-C / RS485	→	None
☐ High	CAN to Serial 1	→	Commgr port 1
☐ High	CAN to Serial 2	→	None
☐ High	CAN to Serial 3	→	None
☐ High	Commgr port 1	→	Serial to CAN 1
☐ High	Commgr port 2	→	UART-A / RS232-A
☐ High	Commgrf port 3	→	UART-B / RS232-B
☐ High	Commgr port 4	→	None
☐ High	Commgr port 5	→	None
☐ High	Commgr port 6	→	None
☐ High	Tunnel 1	→	None
☐ High	Tunnel 2	→	None
☐ High	Tunnel 3	→	None
☐ High	CAN wrapper 0 for serial tran...	→	None

Fig. 69: **CEX - I/O Setup**

8. Go to Input/Output menu → **CAN Setup section**.

   Finally, configure the reception **mailbox with ID 1302**, assign at least 4 mailboxes:



Units ID DEC Units Mask BIN Baudrate 1000000.0

Mailboxes reserved RX: 16 Mailboxes available TX: 16

#		Mailboxes	Extended	ID	Mask
1	—	4	□	1302	11111111111
2	—	4	□	8	11111111111
3	—	4	□	9	11111111111
4	—	4	□	10	11111111111

Fig. 70: **CEX - CAN Seup (Mailboxes) configuration**

3.5.5.2 MC01

In order to communicate a **Veronte BCS** with a **MC01** via CAN, the following connection is required:

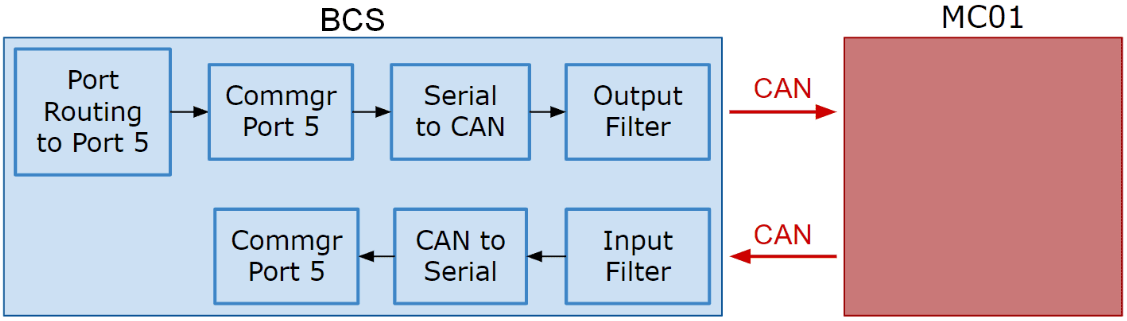


Fig. 71: **Communication diagram BCS - MC01**

The following steps explain how to configure the communication between a **BCS** and a **MC01**.

MC01 PDI Builder side

1. By default, MC01 is configurated with a connection Serial to CAN, with the following **Standard** CAN IDs:

   - Tx CAN Id: 1301

- Rx CAN Id: 1302

BCS PDI Builder side

2. Go to Communications menu → **Ports section**.

   Remove Port 5 from the Forward group and **Add Port 5 to the Route group**, with target MC01's Address. This address must be chosen in the destination path of the MC01 (40117 for the example).

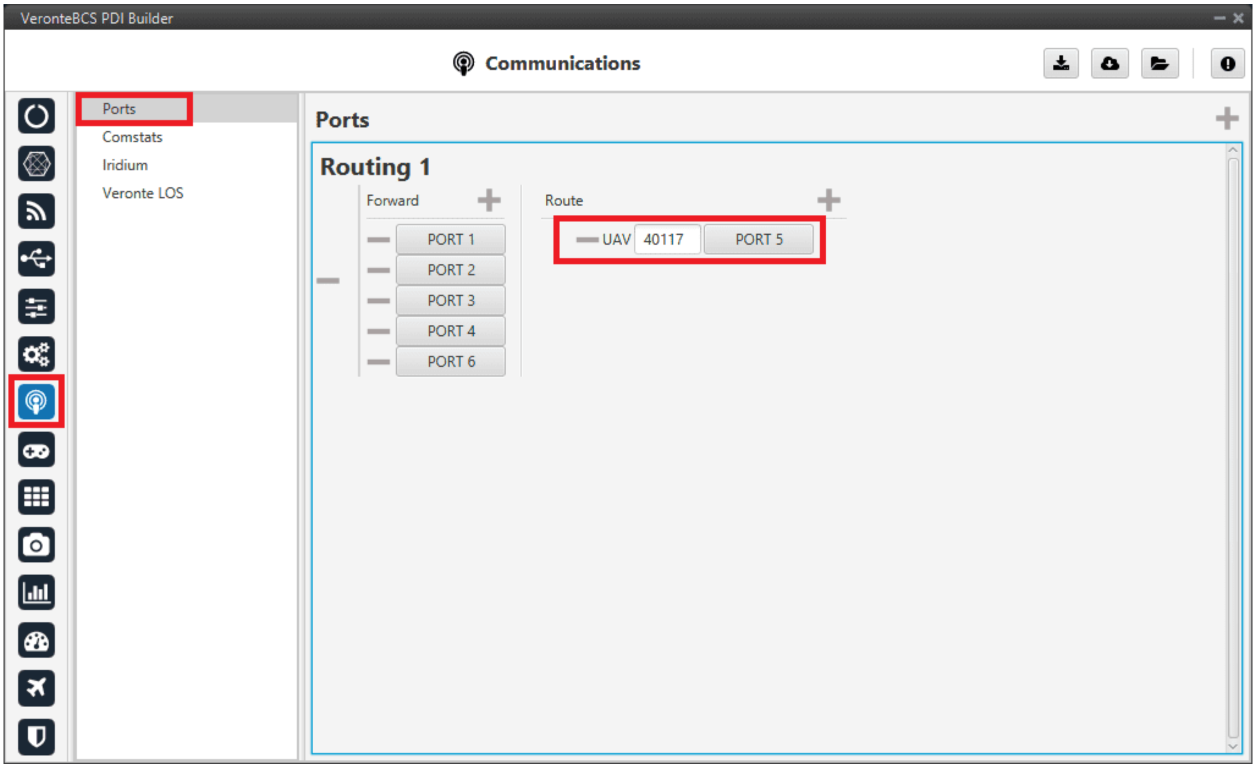
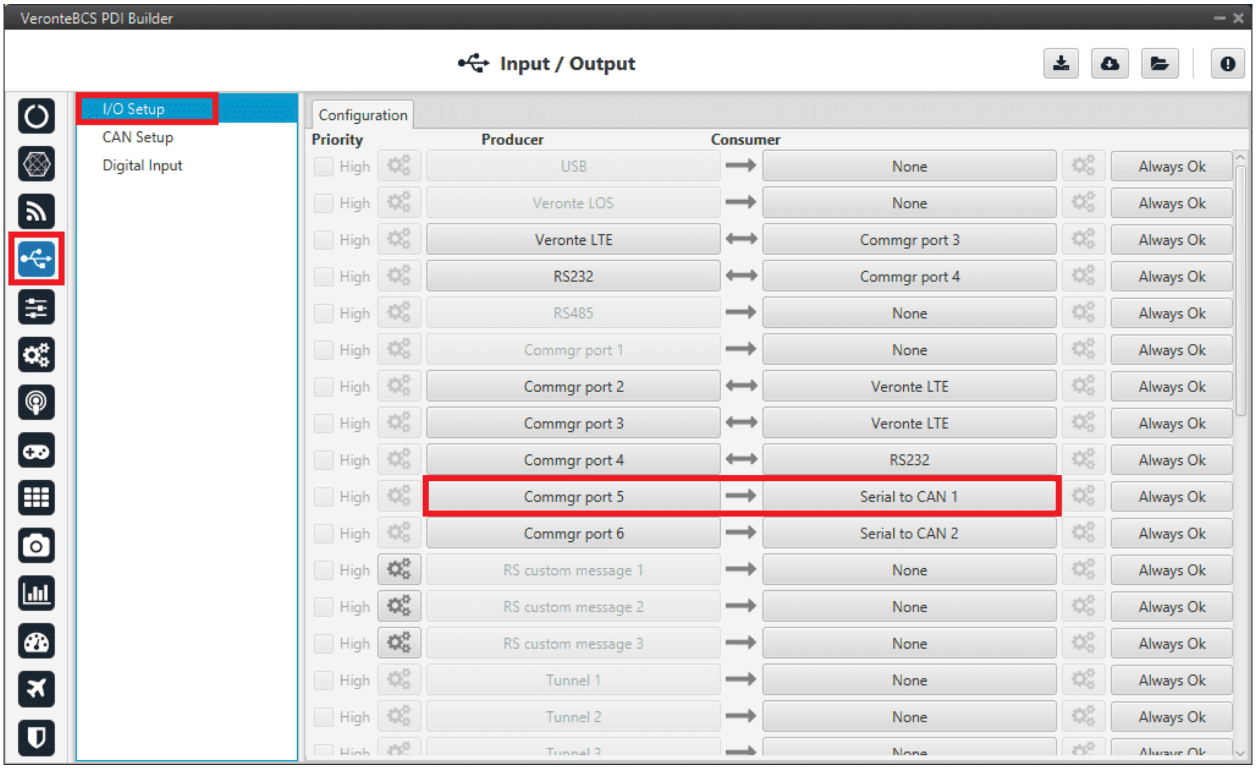


Fig. 72: **Routing configuration**

3. Go to Input/Output menu → **I/O Setup section**.

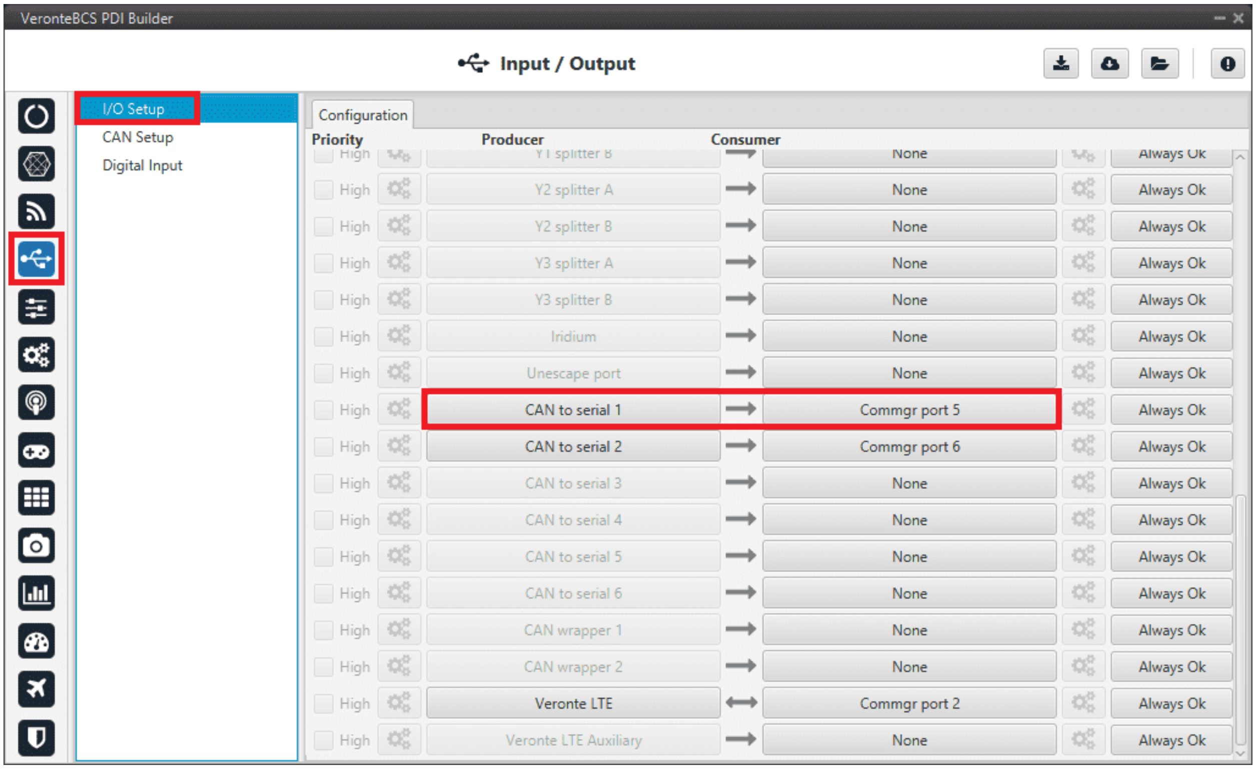
   Connect the **Commgr Port 5** to the **Serial to CAN 1**.



Priority		Producer		Consumer		
□ High		USB	→	None		Always Ok
□ High		Veronte LOS	→	None		Always Ok
□ High		Veronte LTE	↔	Commgr port 3		Always Ok
□ High		RS232	↔	Commgr port 4		Always Ok
□ High		RS485	→	None		Always Ok
□ High		Commgr port 1	→	None		Always Ok
□ High		Commgr port 2	↔	Veronte LTE		Always Ok
□ High		Commgr port 3	↔	Veronte LTE		Always Ok
□ High		Commgr port 4	↔	RS232		Always Ok
□ High		Commgr port 5	→	Serial to CAN 1		Always Ok
□ High		Commgr port 6	→	Serial to CAN 2		Always Ok
□ High		RS custom message 1	→	None		Always Ok
□ High		RS custom message 2	→	None		Always Ok
□ High		RS custom message 3	→	None		Always Ok
□ High		Tunnel 1	→	None		Always Ok
□ High		Tunnel 2	→	None		Always Ok
□ High		Tunnel 3	→	None		Always Ok

Fig. 73: **I/O Setup - Serial to CAN**

Then, connect **CAN to Serial 1** to **Commgr Port 5**:



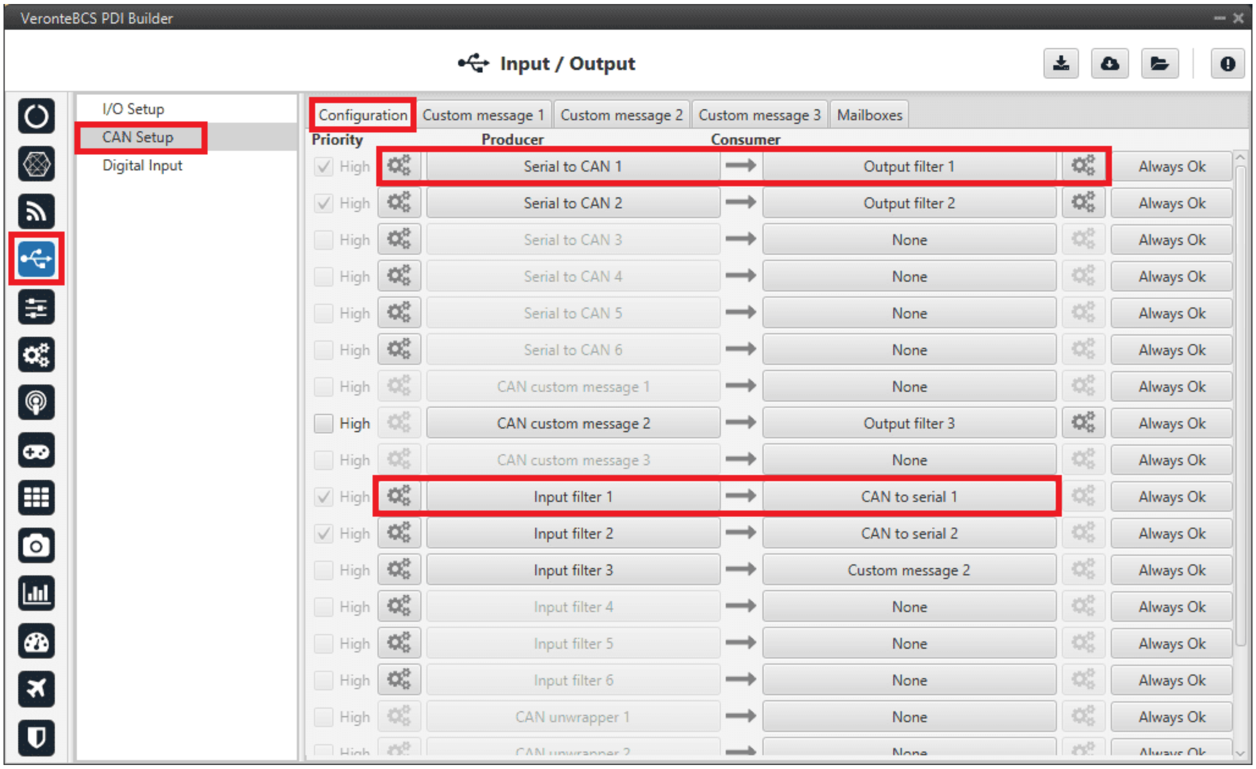
Priority		Producer		Consumer		
□ High		Y1 splitter B	→	None		Always Ok
□ High		Y2 splitter A	→	None		Always Ok
□ High		Y2 splitter B	→	None		Always Ok
□ High		Y3 splitter A	→	None		Always Ok
□ High		Y3 splitter B	→	None		Always Ok
□ High		Iridium	→	None		Always Ok
□ High		Unescape port	→	None		Always Ok
□ High		CAN to serial 1	→	Commgr port 5		Always Ok
□ High		CAN to serial 2	→	Commgr port 6		Always Ok
□ High		CAN to serial 3	→	None		Always Ok
□ High		CAN to serial 4	→	None		Always Ok
□ High		CAN to serial 5	→	None		Always Ok
□ High		CAN to serial 6	→	None		Always Ok
□ High		CAN wrapper 1	→	None		Always Ok
□ High		CAN wrapper 2	→	None		Always Ok
□ High		Veronte LTE	↔	Commgr port 2		Always Ok
□ High		Veronte LTE Auxiliary	→	None		Always Ok

Fig. 74: **I/O Setup - CAN to Serial**

4. Go to Input/Output menu → CAN Setup section → **Configuration tab**.

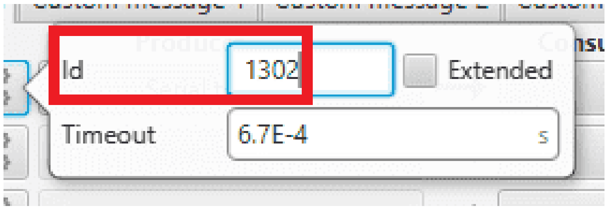
   Connect a **Serial to CAN** with the right Id (**CAN ID 1302**) to an **Output filter**.

   In addition, connect an **Input filter** with the right Id (**CAN ID 1301**) to a **CAN to Serial**:



Priority		Producer		Consumer	
☑	High	Serial to CAN 1	→	Output filter 1	Always Ok
☑	High	Serial to CAN 2	→	Output filter 2	Always Ok
□	High	Serial to CAN 3	→	None	Always Ok
□	High	Serial to CAN 4	→	None	Always Ok
□	High	Serial to CAN 5	→	None	Always Ok
□	High	Serial to CAN 6	→	None	Always Ok
□	High	CAN custom message 1	→	None	Always Ok
□	High	CAN custom message 2	→	Output filter 3	Always Ok
□	High	CAN custom message 3	→	None	Always Ok
☑	High	Input filter 1	→	CAN to serial 1	Always Ok
☑	High	Input filter 2	→	CAN to serial 2	Always Ok
□	High	Input filter 3	→	Custom message 2	Always Ok
□	High	Input filter 4	→	None	Always Ok
□	High	Input filter 5	→	None	Always Ok
□	High	Input filter 6	→	None	Always Ok
□	High	CAN unwrapper 1	→	None	Always Ok
□	High	CAN unwrapper 2	→	None	Always Ok

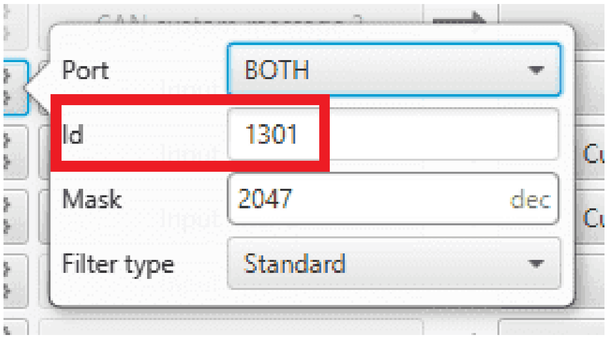
Fig. 75: **CAN Setup**



Id: 1302 □ Extended

Timeout: 6.7E-4 s

Fig. 76: **CAN Setup - Serial to CAN configuration**



Port: BOTH

Id: 1301

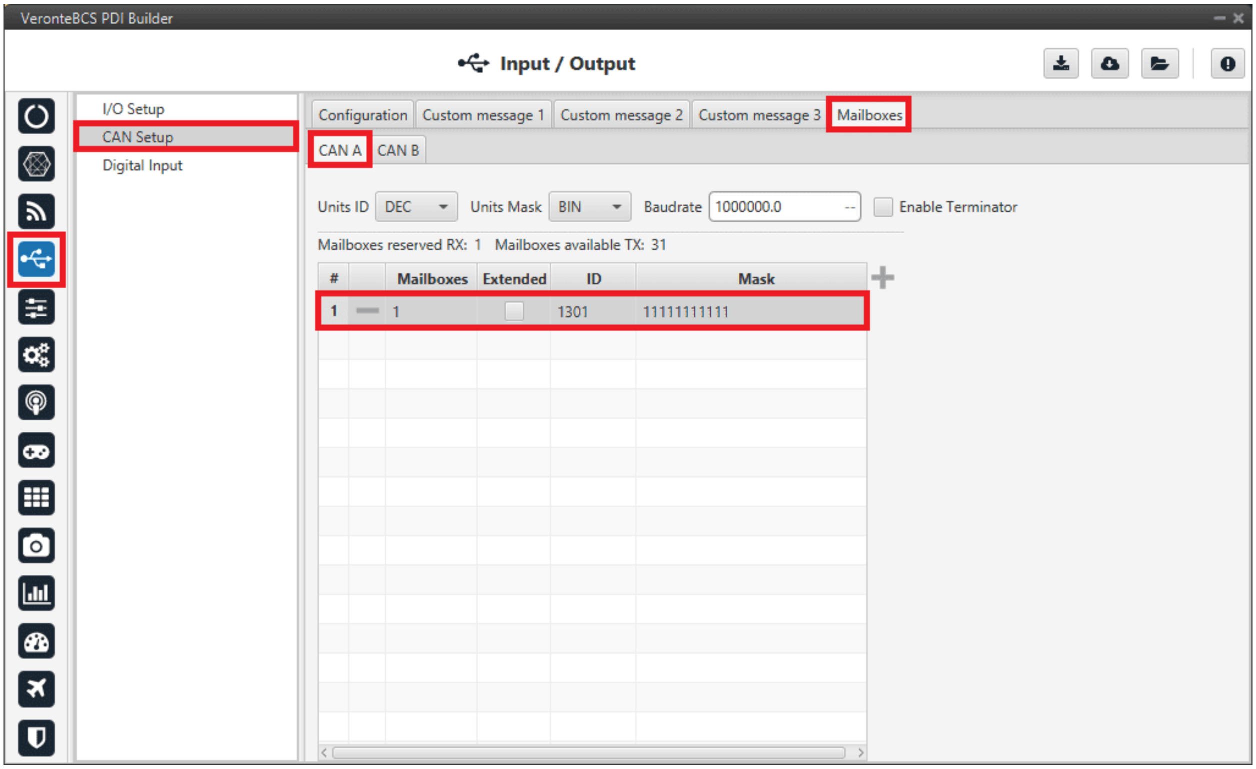
Mask: 2047 dec

Filter type: Standard

Fig. 77: **CAN Setup - Input filter configuration**

5. Go to Input/Output menu → CAN Setup section → **Mailboxes tab**.

   Finally, configure the reception **mailbox with ID 1301**, assign at least 1 mailbox:



Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

Mailboxes reserved RX: 1 Mailboxes available TX: 31

#		Mailboxes	Extended	ID	Mask
1	—	1	□	1301	11111111111

Fig. 78: **Mailboxes configuration**

3.5.5.3 VSE (Veronte Stick Expander)

To configure the VSE in **BCS PDI Builder** it is only needed to follow the steps explained in the **BCS unit configuration** in the *General case - PPM stick integration* section of this manual.

In the **step 1** of that explanation, there is already a transmitter configured with the required configuration of the VSE, users will find it as **Brand**: Embention and **Model**: Stick Expander.

CHAPTER

FOUR

TROUBLESHOOTING

4.1 Communication lost with intenal Digi radio

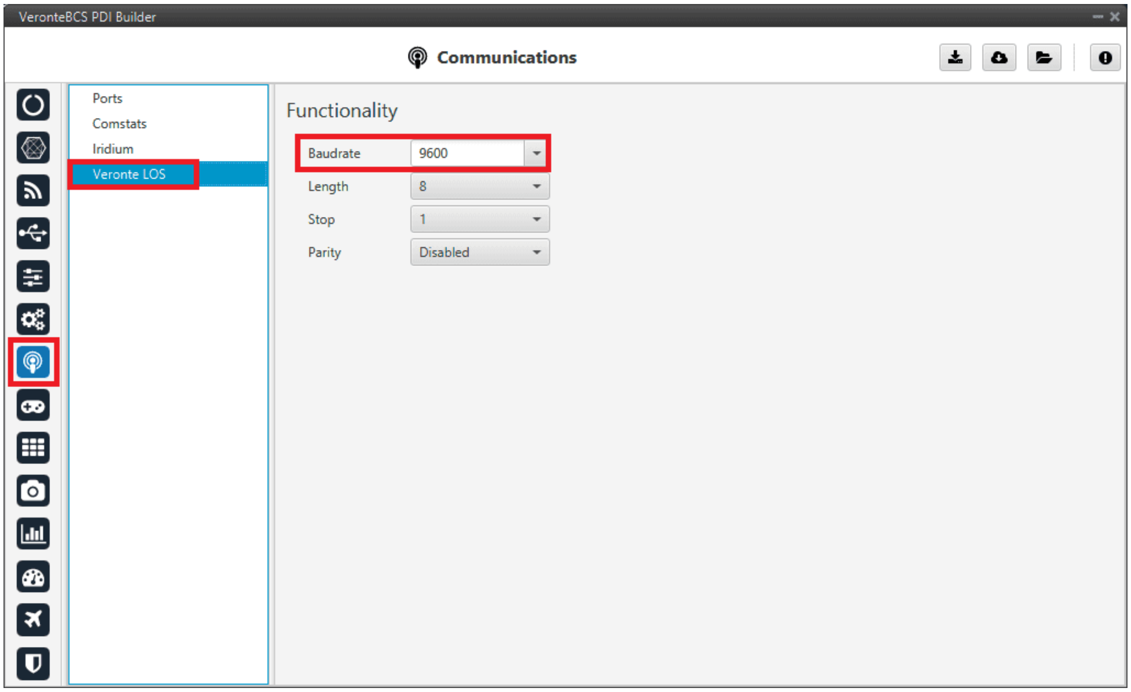
Most of the time, the communication between **BCS** and **Digi radio** is lost due to a change in its baudrate.

In **BCS PDI Builder** it is set to **115200** by default, however, in **Digi radios** the factory default baudrate at reset is **9600**.

To recover communication, try changing the baudrate on one of them to match.

1. Go to **Communications** → **Veronte LOS** section.

   Set the **Baudrate** to 9600.



Baudrate 9600
Length 8
Stop 1
Parity Disabled

Fig. 1: **Veronte LOS baudrate**

2. Check the steps described in the *Digi internal radio -> Integration examples* section to see if the module is now detected in XCTU software.

   Then, if desired, the user can change the radio baudrate to 115200 and after that also change it for **Veronte BCS**.

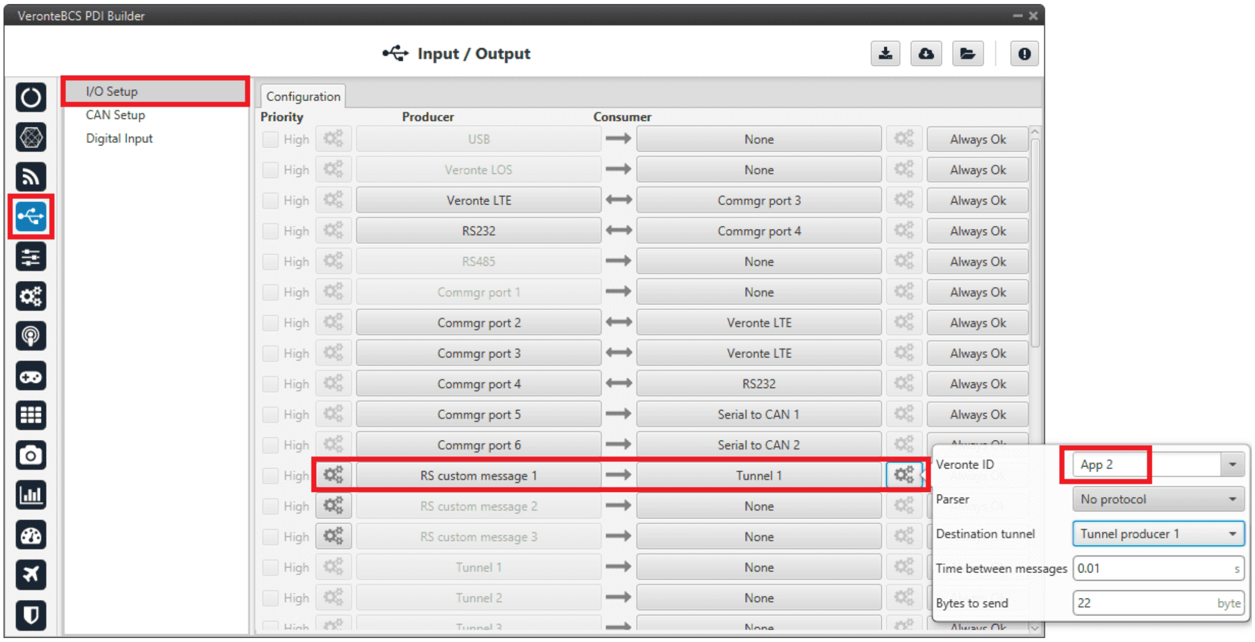
4.2 Debug serial messages transmission

To check that the transmission of serial messages is being carried out correctly, the user can view what is being sent in the **1x PDI Calibration** software hyperterminal. To do this:

In BCS PDI Builder

1. Go to **Input/Output** → **I/O Setup**.

   Connect the **RS custom message producer** (where the message is configured) to a **Tunnel** with **App2 address**. In this case the message is configured in the *RS custom message* 1 producer.



Priority	Producer		Consumer	
High	USB	→	None	Always Ok
High	Veronte LOS	→	None	Always Ok
High	Veronte LTE	↔	Commgr port 3	Always Ok
High	RS232	↔	Commgr port 4	Always Ok
High	RS485	→	None	Always Ok
High	Commgr port 1	→	None	Always Ok
High	Commgr port 2	↔	Veronte LTE	Always Ok
High	Commgr port 3	↔	Veronte LTE	Always Ok
High	Commgr port 4	↔	RS232	Always Ok
High	Commgr port 5	→	Serial to CAN 1	Always Ok
High	Commgr port 6	→	Serial to CAN 2	
High	RS custom message 1	→	Tunnel 1	
High	RS custom message 2	→	None	
High	RS custom message 3	→	None	
High	Tunnel 1	→	None	
High	Tunnel 2	→	None	
High	Tunnel 3	→	None	

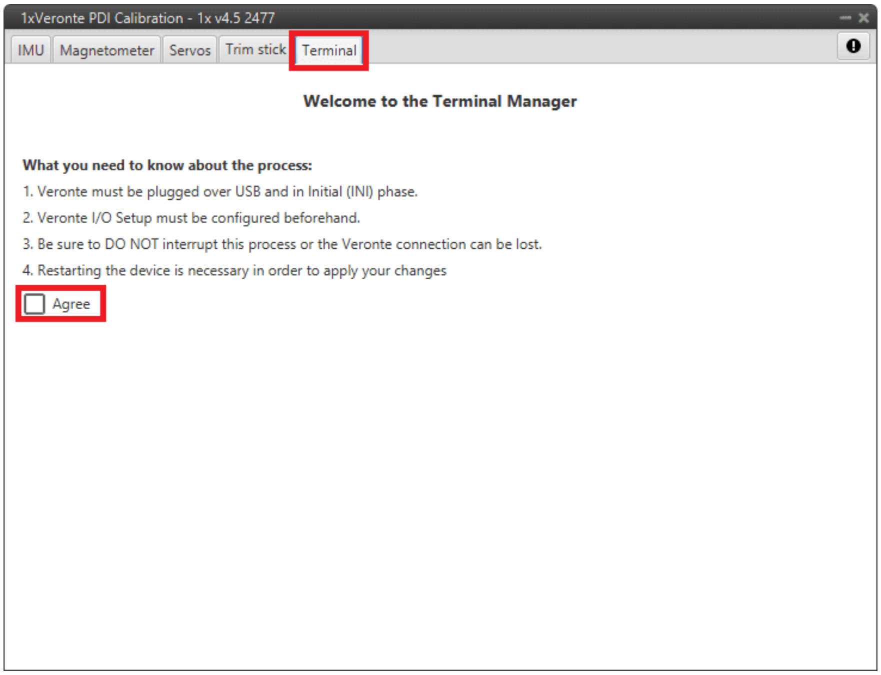
Veronte ID	App 2
Parser	No protocol
Destination tunnel	Tunnel producer 1
Time between messages	0.01 s
Bytes to send	22 byte

Fig. 2: **RS cutom message** → **Tunnel**

In 1x PDI Calibration

2. Go to **Terminal tab**.

   Click on **Agree**:



Welcome to the Terminal Manager

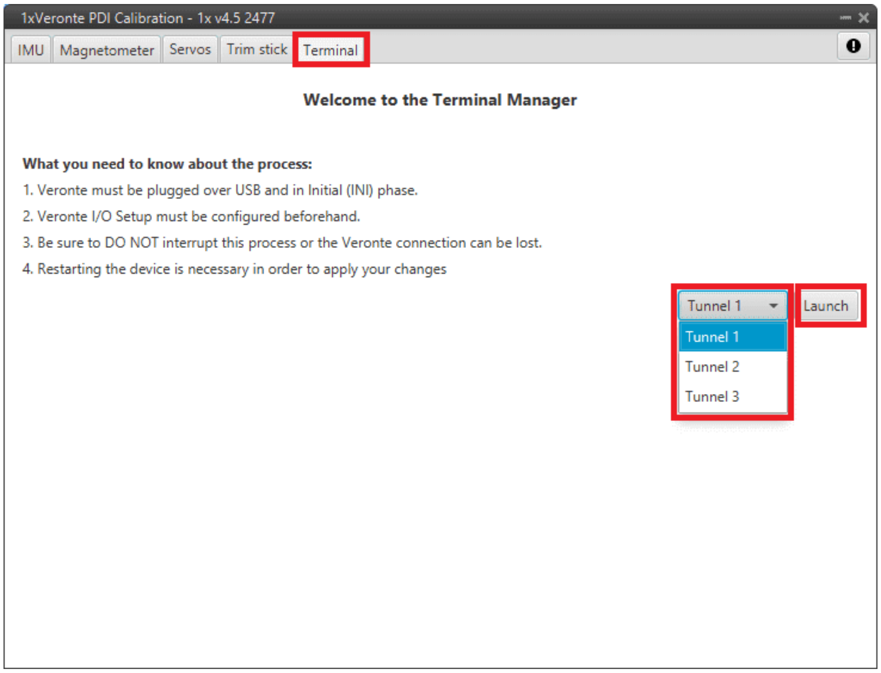
What you need to know about the process:

1. Veronte must be plugged over USB and in Initial (INI) phase.
2. Veronte I/O Setup must be configured beforehand.
3. Be sure to DO NOT interrupt this process or the Veronte connection can be lost.
4. Restarting the device is necessary in order to apply your changes

☐ Agree

Fig. 3: **Terminal tab**

3. Next, select the **Tunnel 1** (this is the one that has been configured in **BCS PDI Builder**) and click on **Launch**:



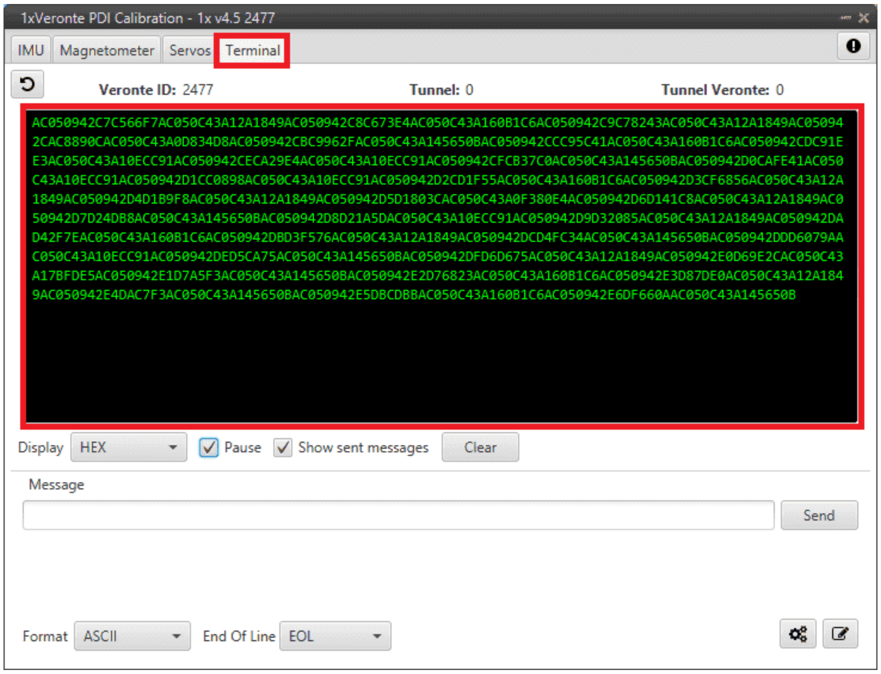
Welcome to the Terminal Manager

What you need to know about the process:

1. Veronte must be plugged over USB and in Initial (INI) phase.
2. Veronte I/O Setup must be configured beforehand.
3. Be sure to DO NOT interrupt this process or the Veronte connection can be lost.
4. Restarting the device is necessary in order to apply your changes

Fig. 4: **Terminal tab - Tunnel selected**

The tunnel console should open and the user will be able to view the message being sent:



Veronte ID: 2477 **Tunnel:** 0 **Tunnel Veronte:** 0

```
AC050942C7C566F7AC050C43A12A1849AC050942C8C673E4AC050C43A160B1C6AC050942C9C78243AC050C43A12A1849AC05094
2CAC8890CAC050C43A0D834D8AC050942CBC9962FAC050C43A145650BAC050942CCC95C41AC050C43A160B1C6AC050942CDC91E
E3AC050C43A10ECC91AC050942CECA29E4AC050C43A10ECC91AC050942CFCB37C0AC050C43A145650BAC050942D0CAFE41AC050
C43A10ECC91AC050942D1CC0898AC050C43A10ECC91AC050942D2CD1F55AC050C43A160B1C6AC050942D3CF6856AC050C43A12A
1849AC050942D4D1B9F8AC050C43A12A1849AC050942D5D1803CAC050C43A0F380E4AC050942D6D141C8AC050C43A12A1849AC0
50942D7D24DB8AC050C43A145650BAC050942D8D21A5DAC050C43A10ECC91AC050942D9D32085AC050C43A12A1849AC050942DA
D42F7EAC050C43A160B1C6AC050942DBD3F576AC050C43A12A1849AC050942DCD4FC34AC050C43A145650BAC050942DDD6079AA
C050C43A10ECC91AC050942DED5CA75AC050C43A145650BAC050942DFD6D675AC050C43A12A1849AC050942E0D69E2CAC050C43
A17BFDE5AC050942E1D7A5F3AC050C43A145650BAC050942E2D76823AC050C43A160B1C6AC050942E3D87DE0AC050C43A12A184
9AC050942E4DAC7F3AC050C43A145650BAC050942E5DBCDBBAC050C43A160B1C6AC050942E6DF660AAC050C43A145650B
```

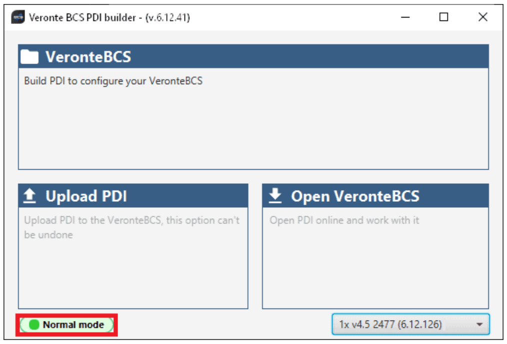
Display HEX ☑ Pause ☑ Show sent messages

Format ASCII End Of Line EOL

Fig. 5: **Tunnel console**

4.3 Maintenance mode

The user can simply enter maintenance mode via **BCS PDI Builder** by clicking on the "Normal mode" button in the initial menu. In addition, exiting maitenance mode is the same process.



Build PDI to configure your VeronteBCS

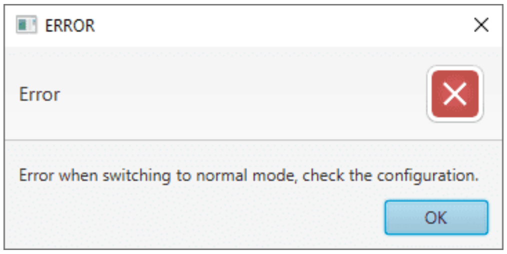
Upload PDI to the VeronteBCS, this option can't be undone

Open PDI online and work with it

Fig. 6: **Enter/Exit maintenance mode**

4.4 Maintenance mode (loaded with errors)

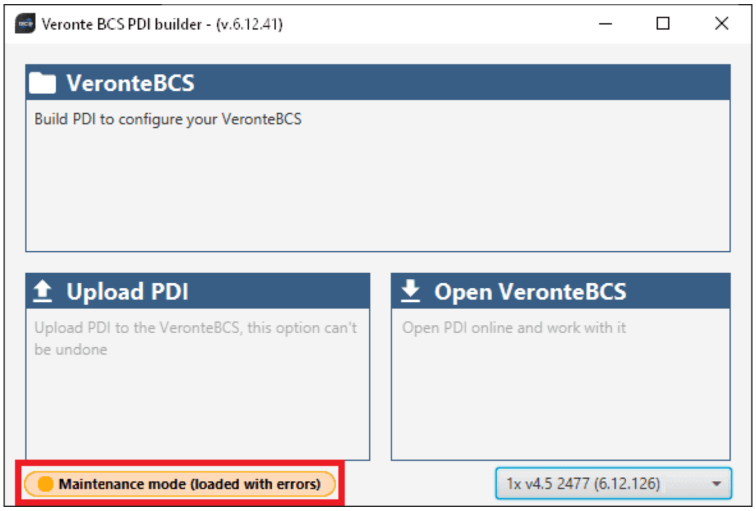
The following error message may appear when trying to save a change or import a configuration.



Error when switching to normal mode, check the configuration.

Fig. 7: **Error message**

Therefore, **BCS** will be in 'Maintenance mode (loaded with errors)':


Veronte BCS PDI builder - (v.6.12.41)
VeronteBCS
Upload PDI
Open VeronteBCS
Maintenance mode (loaded with errors)
1x v4.5 2477 (6.12.126)

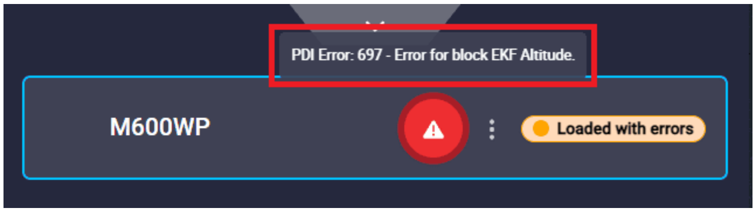
Build PDI to configure your VeronteBCS

Upload PDI to the VeronteBCS, this option can't be undone

Open PDI online and work with it

Fig. 8: **Maintenance mode (loaded with errors)**

To check what the source of the problem is, the user can consult the **Veronte Ops Platform panel**, which will show which PDI Error it is. For more information on this panel, see Platform panel section of the **Veronte Ops** user manual.



PDI Error: 697 - Error for block EKF Altitude.

Fig. 9: **PDI Error - Veronte Ops**

Then, it is possible to access the **BCS** configuration to fix this error.

In addition, a list of all PDI Errors can be accessed in the List of PDI errors section of the **1x Software Manual**.

4.5 Radios paired but 1x air unit not showing

If the radios of **1x** (air unit) and **BCS** (ground unit), are paired but the air unit does not appear connected in **Veronte Link**, check the Ports configuration on the **BCS** ground unit. To do this:

Go to **Communications** → **Ports** section. It should be similar to the configuration shown in the figure below:

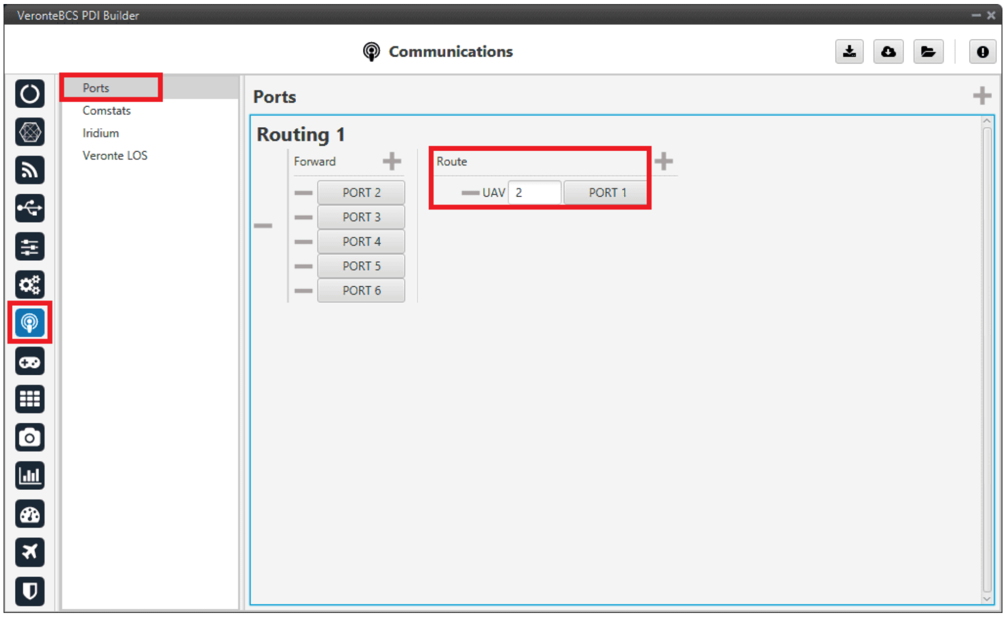


Fig. 10: **BCS (ground unit) - Ports configuration**

4.6 Reduce GNC Task frequency

400 Hz is the maximum possible **frequency**, but it can only be used in **simple configurations**, in other cases it is advisable to **reduce** it to **250-300 Hz**.

To find out if the frequency needs to be reduced in the user configuration, check the **GNC Task Average CPU Ratio variable**.

For correct operation, this variable should be at approximately **60-70%**. If it reports a **higher value**, the **frequency must be lowered**.