4x PDI Builder

Release 6.12.27

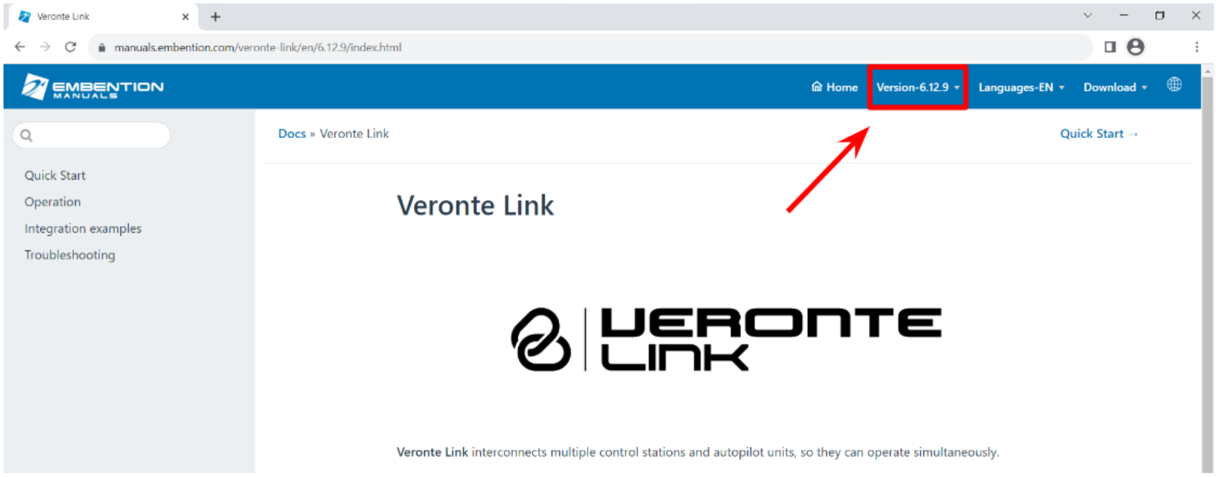
Embention

2024-04-03

CONTENTS

4X PDI Builder is an application for modifying, generating and uploading PDI files for **Arbiters**, the autopilot managers of the redundant **Veronte Autopilot 4x**.

Warning: Select your version before reading any user manual for software. The following image shows where to select a version from any Embention user manual.



Veronte Link

Veronte Link interconnects multiple control stations and autopilot units, so they can operate simultaneously.

CHAPTER

ONE

QUICK START

4x PDI Builder is the main configuration tool to adapt **Arbiters** for a specific system, being its main function to decide which **Autopilot 1x** controls the vehicle at every moment. **4x PDI Builder** includes:

- Communications: through general purpose CAN bus, RS-232, RS-485 and GPIOs.
- Arbitration criterion: **Arbiters** decide which **Autopilot 1x** controls the vehicle according to the criterion decided by the user.

All Veronte devices, including **Arbiters**, are configured with PDI files.

1.1 Download

Once **Veronte Autopilot 4x** has been purchased, a GitHub release should be created for the customer with the application.

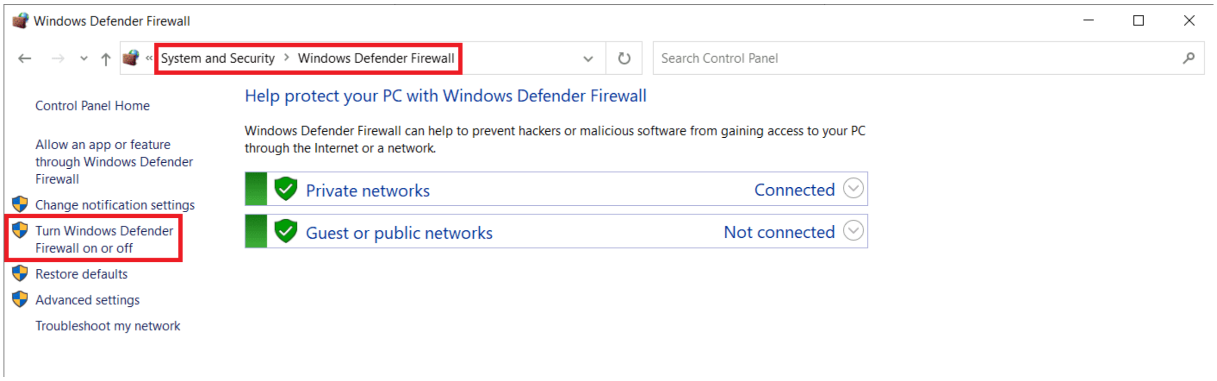
To access to the release and download the software, read the Releases section of the **Joint Collaboration Framework** manual.

1.2 Installation

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”
- Then, click on “Turn windows Defender Firewall on or off”.



Windows Defender Firewall

System and Security > Windows Defender Firewall

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

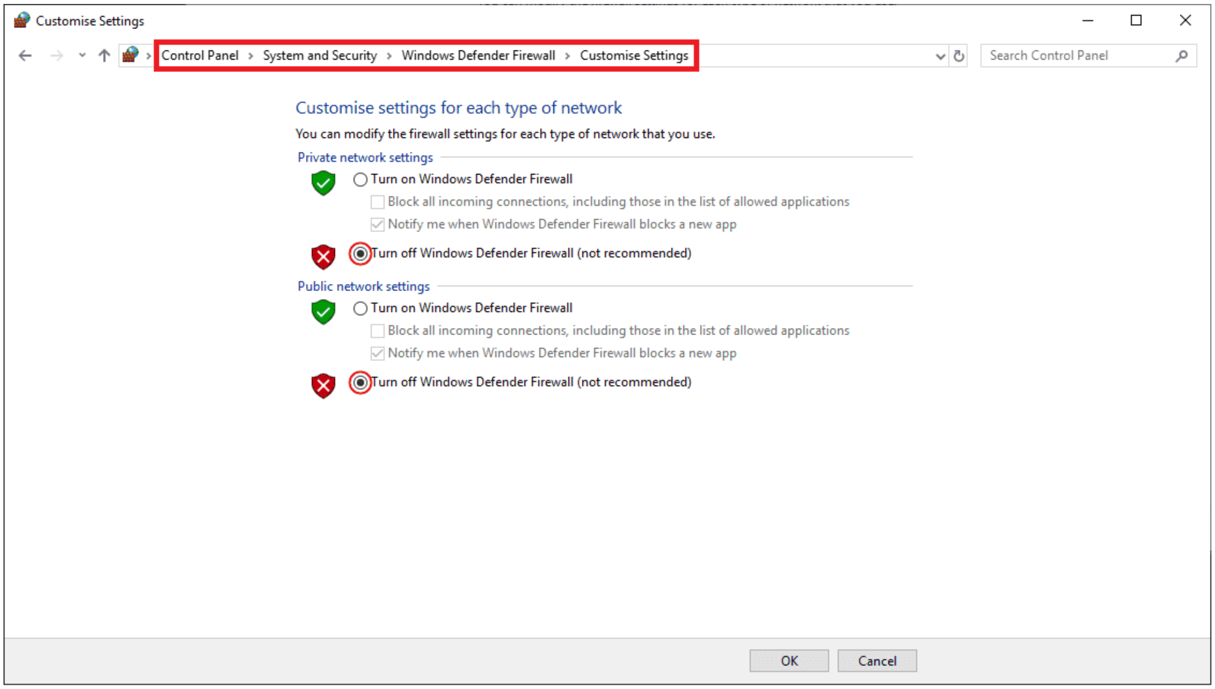
Help protect your PC with Windows Defender Firewall

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

Fig. 1: **Windows Defender Firewall**



Customise Settings

Control Panel > System and Security > Windows Defender Firewall > Customise Settings

Search Control Panel

Customise settings for each type of network

You can modify the firewall settings for each type of network that you use.

Private network settings

○ Turn on Windows Defender Firewall

□ Block all incoming connections, including those in the list of allowed applications

☑ Notify me when Windows Defender Firewall blocks a new app

● Turn off Windows Defender Firewall (not recommended)

Public network settings

○ Turn on Windows Defender Firewall

□ Block all incoming connections, including those in the list of allowed applications

☑ Notify me when Windows Defender Firewall blocks a new app

● Turn off Windows Defender Firewall (not recommended)

OK   Cancel

Fig. 2: **Windows Defender Firewall: Settings**

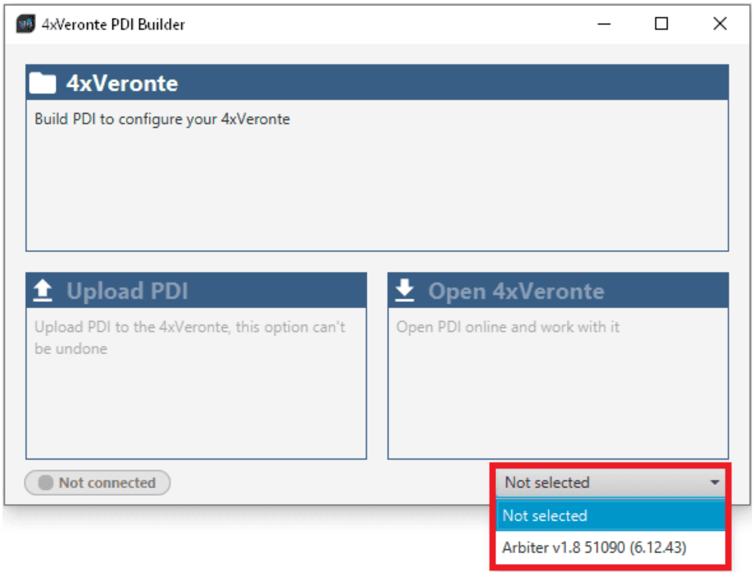
CHAPTER
TWO

CONFIGURATION

This section explains each option and parameter available in **4x PDI Builder**.

Once the installation is finished, connect the **Management Board** of the **4x** (using the **Connector 4**) with Veronte Link.

Then, open **4x PDI Builder** and select the **Arbiter** to configure.



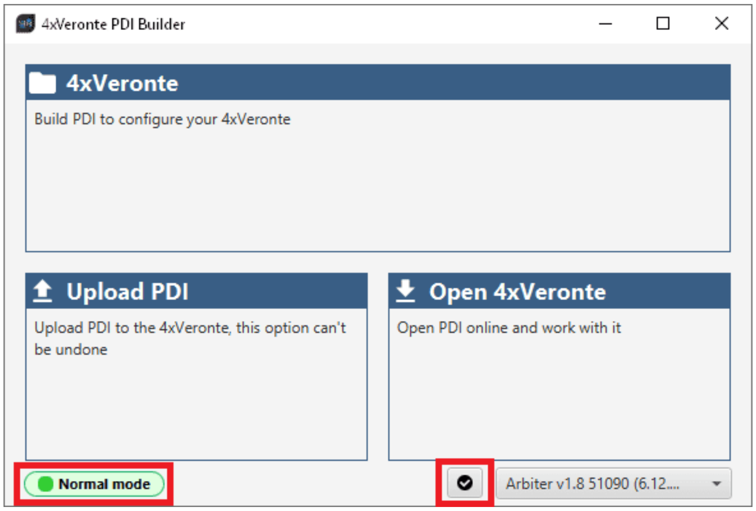
Build PDI to configure your 4xVeronte

Upload PDI to the 4xVeronte, this option can't be undone

Open PDI online and work with it

Fig. 1: **Arbiter ID**

If the **Veronte Autopilot 4x** is correctly connected, **4x PDI Builder** will display the **mode** in which the connected unit is. In addition, a **PDI error button** ✓ will appear:



4xVeronte PDI Builder

4xVeronte

Build PDI to configure your 4xVeronte

Upload PDI

Upload PDI to the 4xVeronte, this option can't be undone

Open 4xVeronte

Open PDI online and work with it

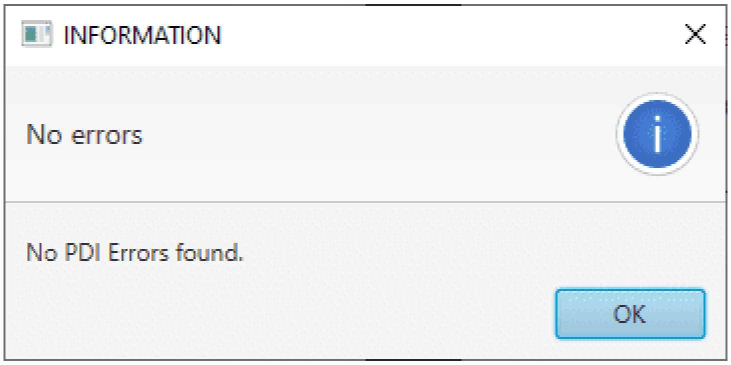
Fig. 2: **4x PDI Builder**

- **4x mode**: The 4x unit should appear in **Normal mode**, as shown in the figure above, or **Maintenance mode**.

  It can also appear as **Maintenance mode (loaded with errors)** or **Normal mode - Disconnected**.

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

- ✓ **PDI Errors** button: The user can check if the connected unit has PDI Errors by simply clicking on it. If there are no errors, the following message is displayed:



INFORMATION

No errors

No PDI Errors found.

Fig. 3: **4x PDI Builder - PDI Errors message**

The user can access now to three configuration options:

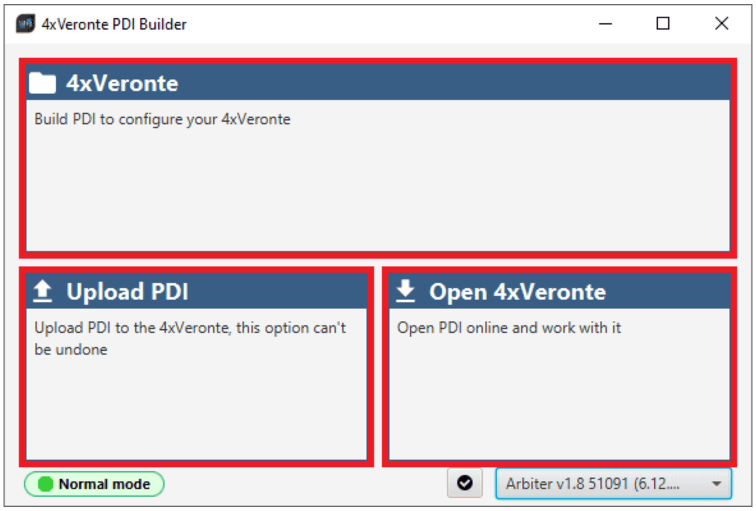

4xVeronte PDI Builder
4xVeronte
Build PDI to configure your 4xVeronte
Upload PDI
Upload PDI to the 4xVeronte, this option can't be undone
Open 4xVeronte
Open PDI online and work with it
Normal mode
Arbiter v1.8 51091 (6.12....

Fig. 4: **4x PDI Builder options**

- **4xVeronte**: It allows the user to work with **offline** configurations. A previously exported 4x PDI configuration can be opened and modified. It is also possible to build a new one from the default configuration.

Note: When an offline configuration is opened, it is possible to select the hardware version to work with on the marked button. Versions `1.8 A` and `1.8 B` are arbiters A and B of version 1.8 respectively.

It is recommended to select the same hardware version that the user's Arbiter has, so that if the user wishes to load this configuration later to the Arbiter, there will be no compatibility problems.

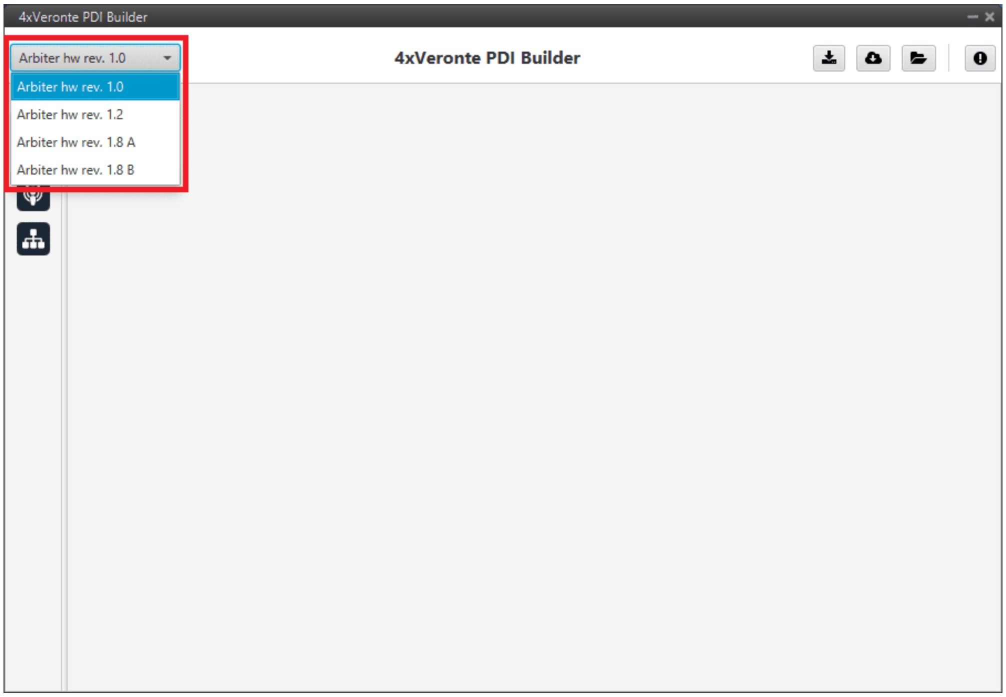
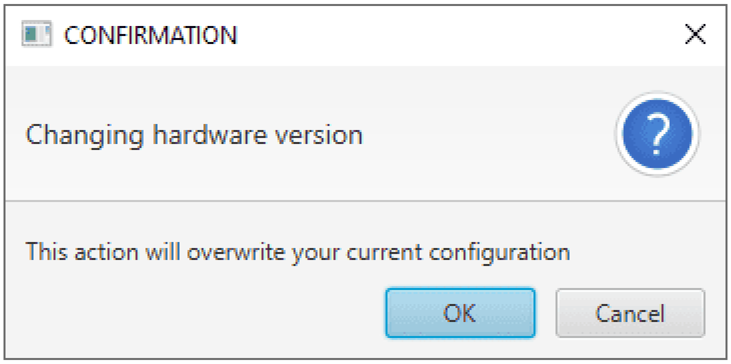


Fig. 5: **Hardware versions selector**

When switching between hardware versions, the following confirmation message appears:



CONFIRMATION

Changing hardware version

This action will overwrite your current configuration

Fig. 6: **Hardware versions selector - Confirmation message**

- **Upload PDI**: A previously exported 4x PDI configuration can be imported to the linked **Arbiter** (PDI files are transferred from computer to **Arbiter**).

> **Warning:** When a configuration is loaded into Arbiter with a version older than the software version being used, an **automatic migration** from the configuration version to the software version being used will be performed.
>
> For more information on this, see *Migrate configuration - Troubleshooting* section of this manual.

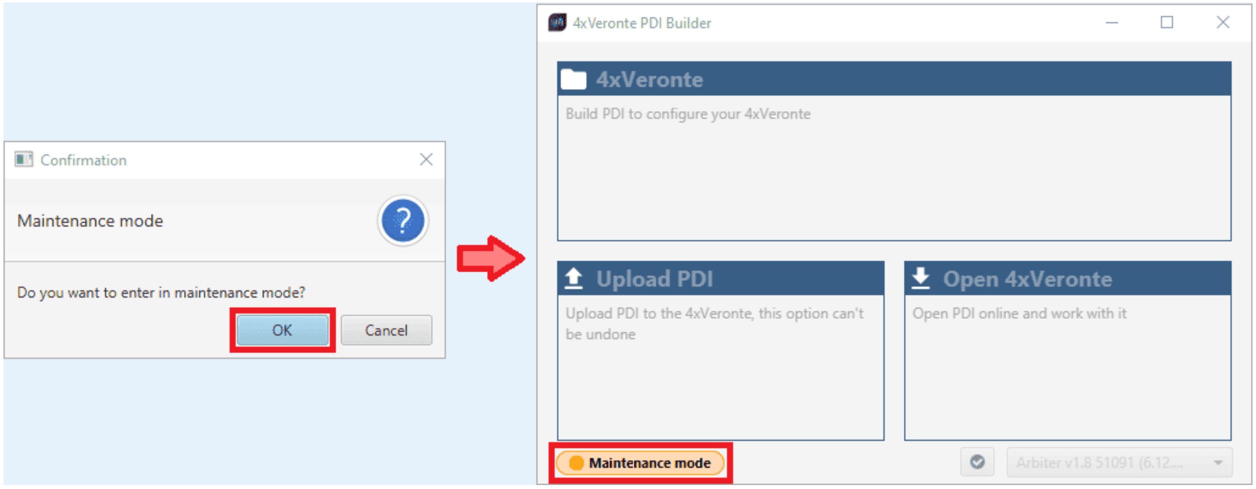
- **Open 4x Veronte**: It opens the configuration (the PDI files) loaded in the **Arbiter** (PDI files are transferred from **Arbiter** to computer). Then, the user can modify it **online**.

Note: PDI files are Arbiter configuration files. These PDI files are stored in the *setup* folder, holding several .xml files which contain all the control system and parameters.

Fig. 7: **PDI files**

Finally, click on '**4xVeronte**' to edit a configuration offline or '**Open 4xVeronte**' to open the current configuration and start editing it online.

Note: The 4x unit must enter **Maintenance mode** for the user to start editing, so it is necessary to accept the confirmation panel below.



Confirmation

Maintenance mode

Do you want to enter in maintenance mode?

4xVeronte PDI Builder

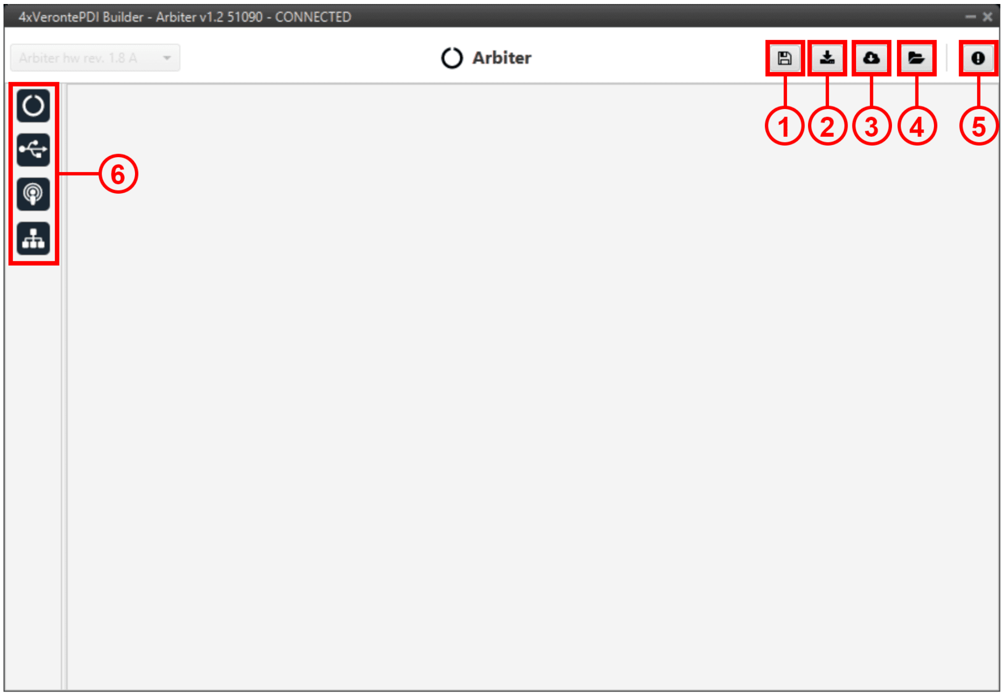
Build PDI to configure your 4xVeronte

Upload PDI to the 4xVeronte, this option can't be undone

Open PDI online and work with it

Fig. 8: **Open 4xVeronte**

Once in Maintenance mode, the user can access the *Initial menu*. The different ‘buttons’ that can be seen in this menu of the **4x PDI Builder** are explained below.

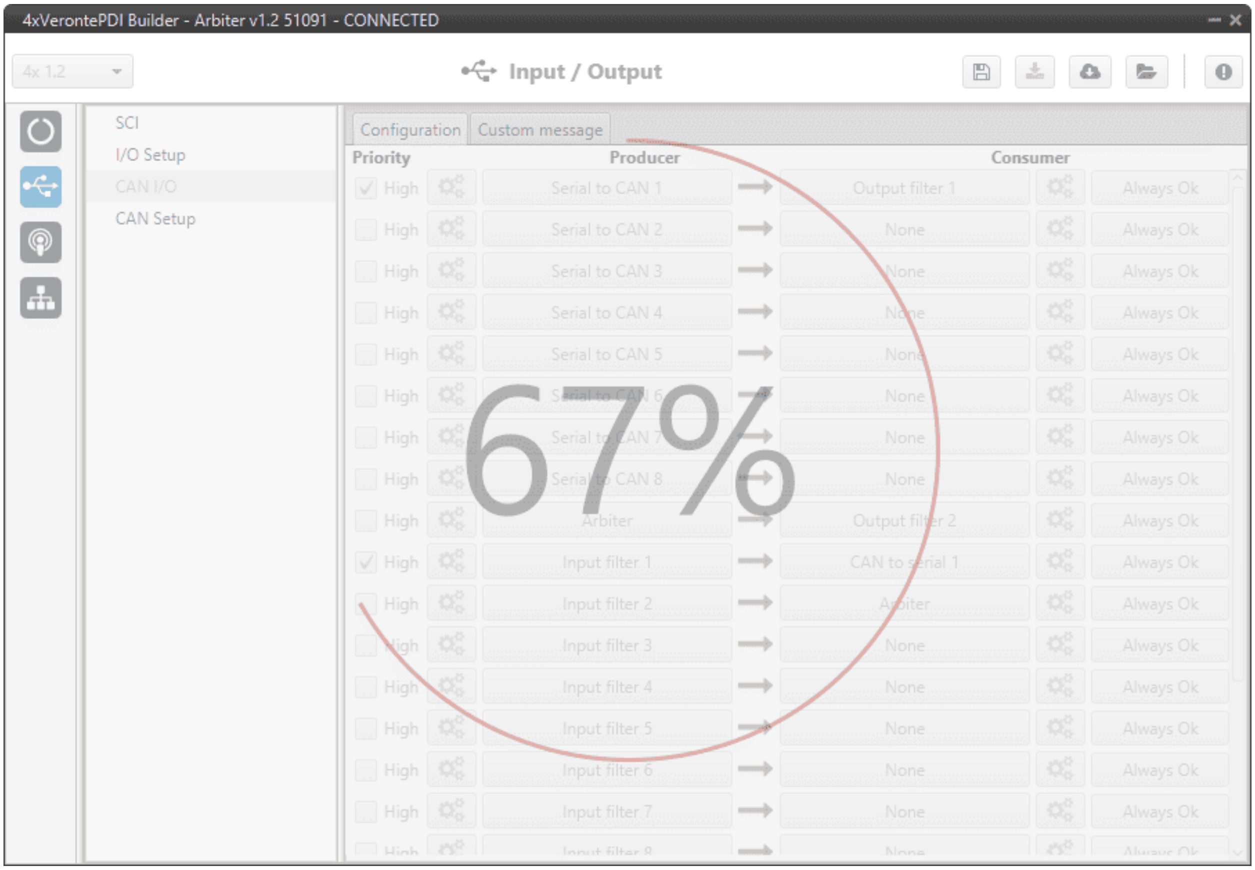


4xVerontePDI Builder - Arbiter v1.2 51090 - CONNECTED

Fig. 9: **Initial menu**

1. **Save and close**: After changes are done, press here to apply changes.

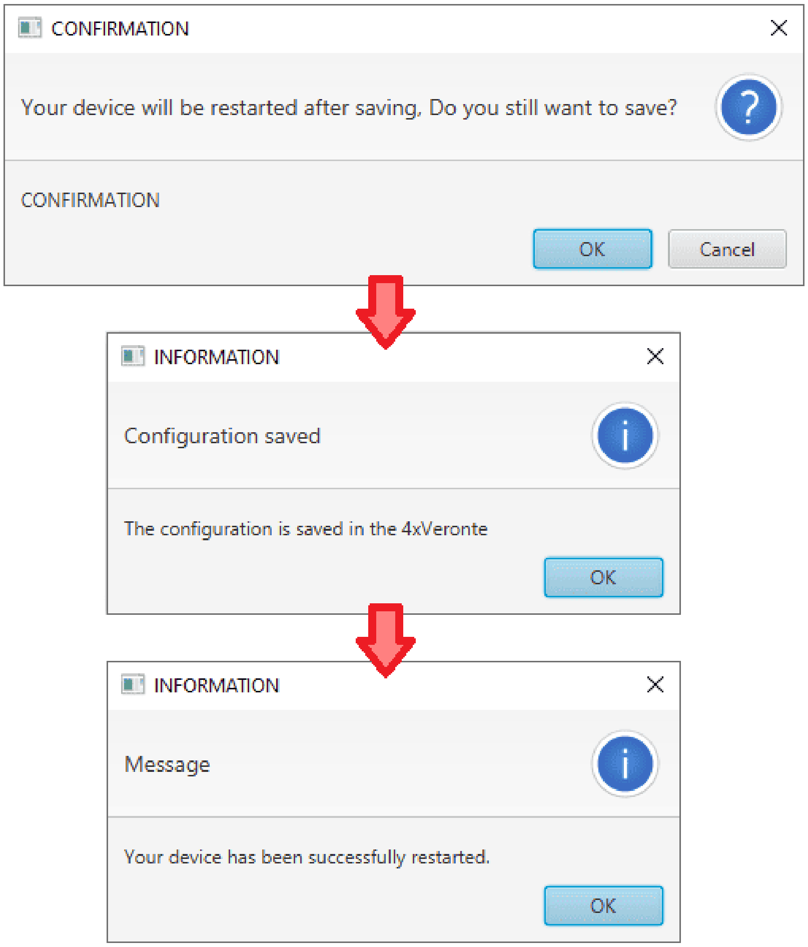
   While saving, a percentage of saving process is displayed.



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		Serial to CAN 7	→	None		Always Ok
☐ High		Serial to CAN 8	→	None		Always Ok
☐ High		Arbiter	→	Output filter 2		Always Ok
☑ High		Input filter 1	→	CAN to serial 1		Always Ok
☐ High		Input filter 2	→	Arbiter		Always Ok
☐ High		Input filter 3	→	None		Always Ok
☐ High		Input filter 4	→	None		Always Ok
☐ High		Input filter 5	→	None		Always Ok
☐ High		Input filter 6	→	None		Always Ok
☐ High		Input filter 7	→	None		Always Ok
☐ High		Input filter 8	→	None		Always Ok

Fig. 10: **Save and close**

In order to save the configuration in the 4x unit it is necessary to **RESET**, therefore the **4x PDI Builder** software will **close**. For this reason, the user must accept the following panels:



Your device will be restarted after saving, Do you still want to save?

CONFIRMATION

Configuration saved

The configuration is saved in the 4xVeronte

Message

Your device has been successfully restarted.

Fig. 11: **Save and close - Confirmation panels**

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

Note: This button will only appear if an **Arbiter** is connected. When working offline this button will not be available.

2. **Export PDI**: After modifying a configuration, press here to save the configuration in the local storage (both folders with the PDI files are downloaded). 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.
3. **Import PDI from repository**: The user can import a configuration file from the GitHub repository and modify it. After that, if the **Save and close** button (1) is pressed, this configuration will be uploaded to the connected **arbiter**.
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 and close** button (1) is pressed, this configuration will be loaded into the connected **arbiter**.
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 **arbiter** 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.

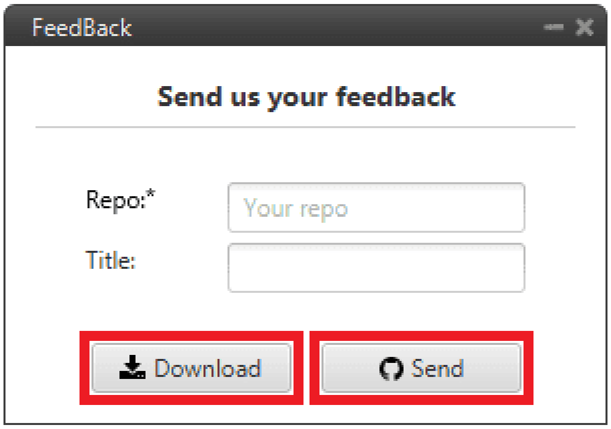


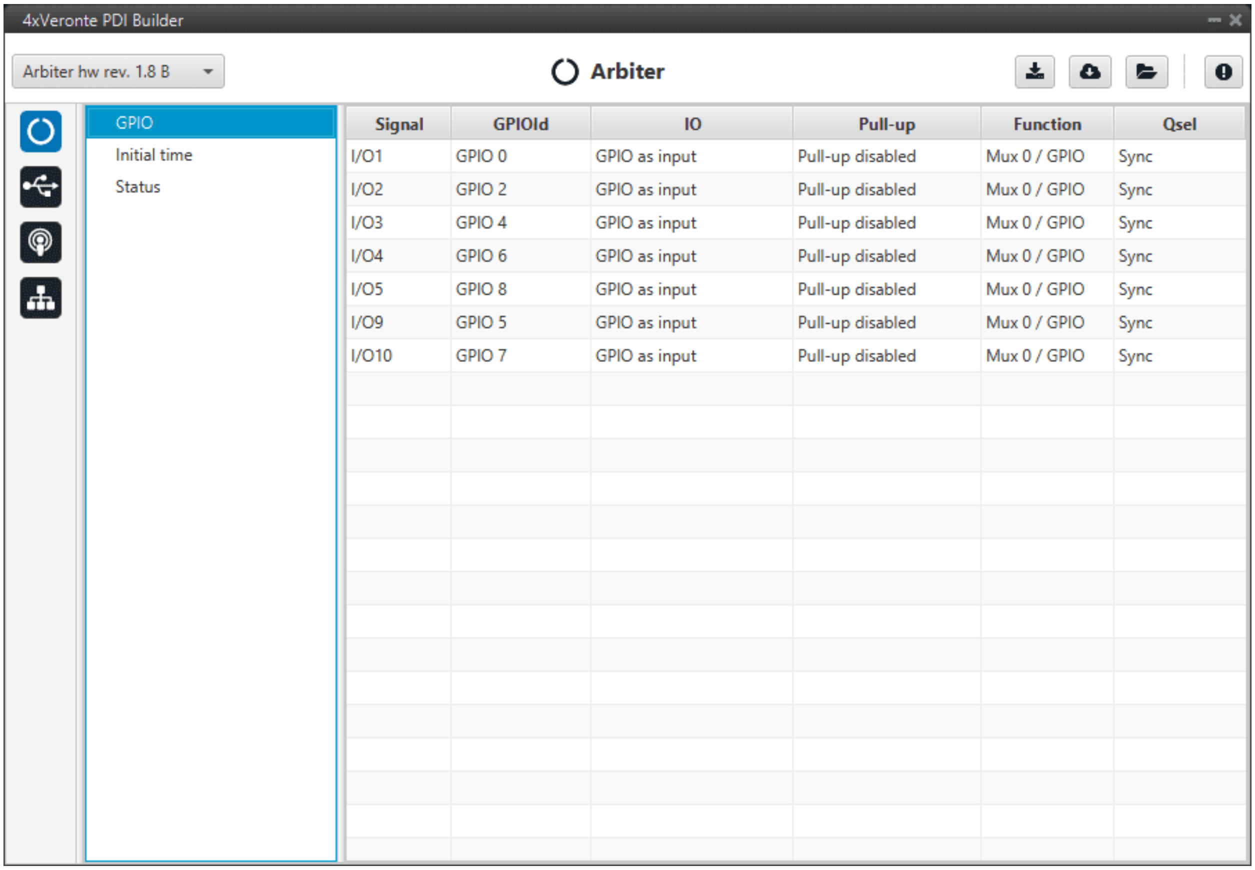
Fig. 12: **Feedback**

6. These are the different functions of the **arbiter**. They are explained in the following menus.
   - *Arbiter*
   - *Input/Output*
   - *Communications*
   - *Arbitration*

2.1 Arbiter

2.1.1 GPIO

In this window, each individual GPIO (General Purpose Input/Output) behavior can be configured:



Signal	GPIOId	IO	Pull-up	Function	Qsel
I/O1	GPIO 0	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O2	GPIO 2	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O3	GPIO 4	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O4	GPIO 6	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O5	GPIO 8	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O9	GPIO 5	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
I/O10	GPIO 7	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync

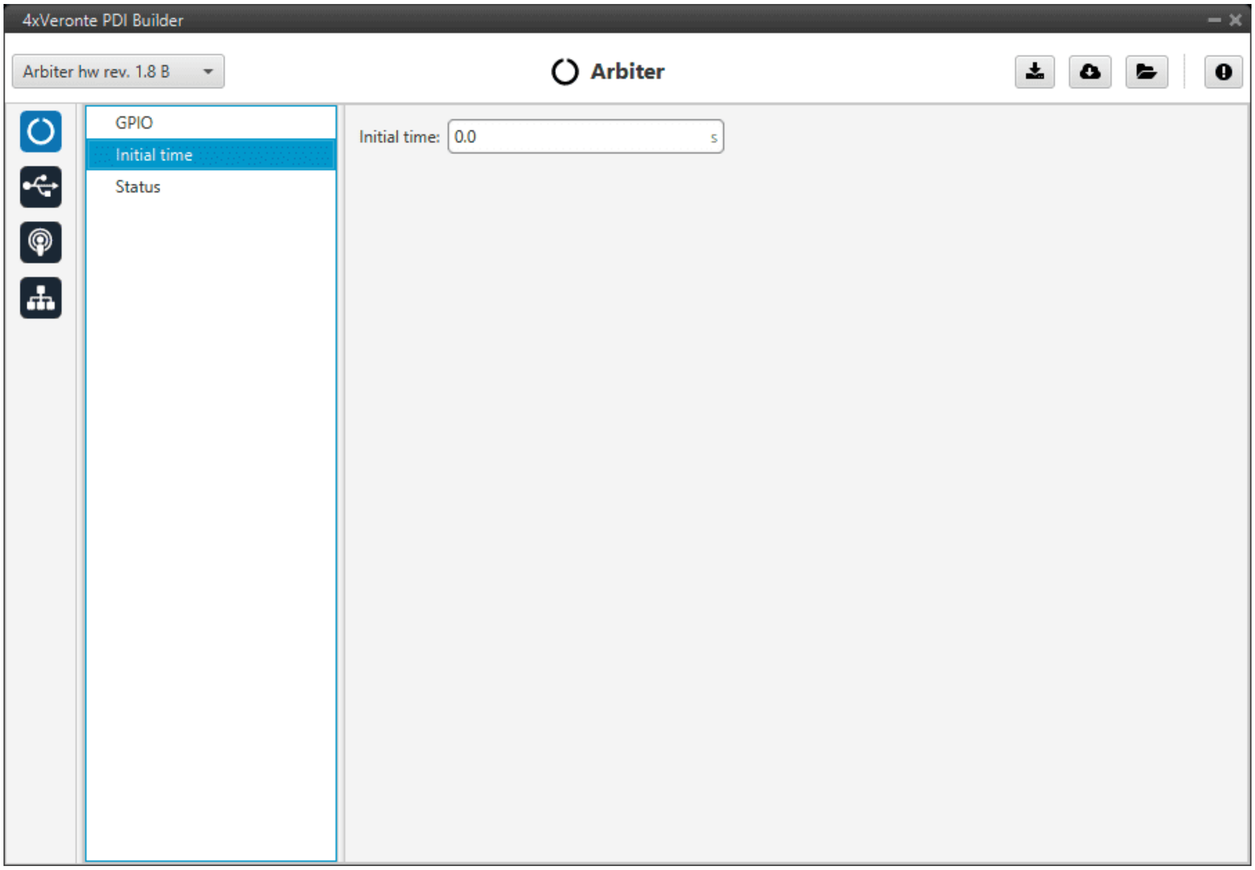
Fig. 13: **GPIO panel**

- **Signal**: Pin ID as described in Arbiter connector pinout - Hardware Installation section of the **4x Hardware Manual**.
- **GPIOId**: GPIO ID of the microcontroller.
- **IO**: Define GPIO as an input or ouput.
- **Pull-up**: Enable or disable the pull-up resistance.
- **Function**: Mux 0/GPIO, Mux 1, Mux 2 or Mux 3. These are the different functionalities that the GPIO can have, depending on the multiplexer.
- **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 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 **3 samples**, but checking 6 times instead of 3.
  - **ASync**: No checks are performed. It is used when it is not used as GPIO.

2.1.2 Initial time

If any **Autopilot 1x** has not power supply, the **Management Board** will detect it as a fault. The **Initial time** parameter is the margin of time from the moment when the **Management Board** is powered up until it starts cheking power supplies.

With this parameter it is possible to connect each **1x** one by one after the **Management Board**.



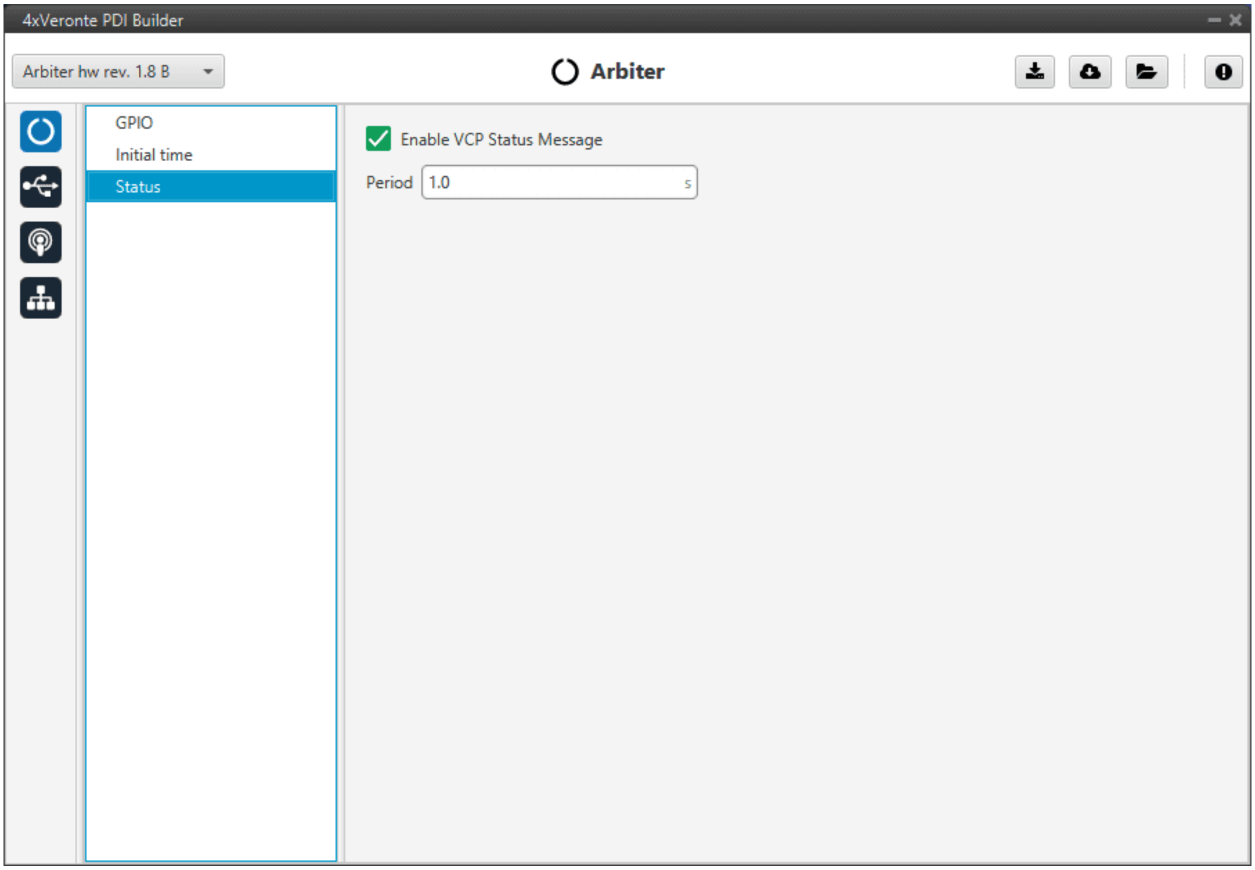
Initial time: 0.0

Fig. 14: **Initial time panel**

2.1.3 Status

- **Enable VCP Status Message:** Enables the periodic sending of the status message that **Veronte Link** uses to recognize an **arbiter**.
- **Period:** Establishes the desired period to send repeatedly the status message.

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



☑ Enable VCP Status Message

Period 1.0 s

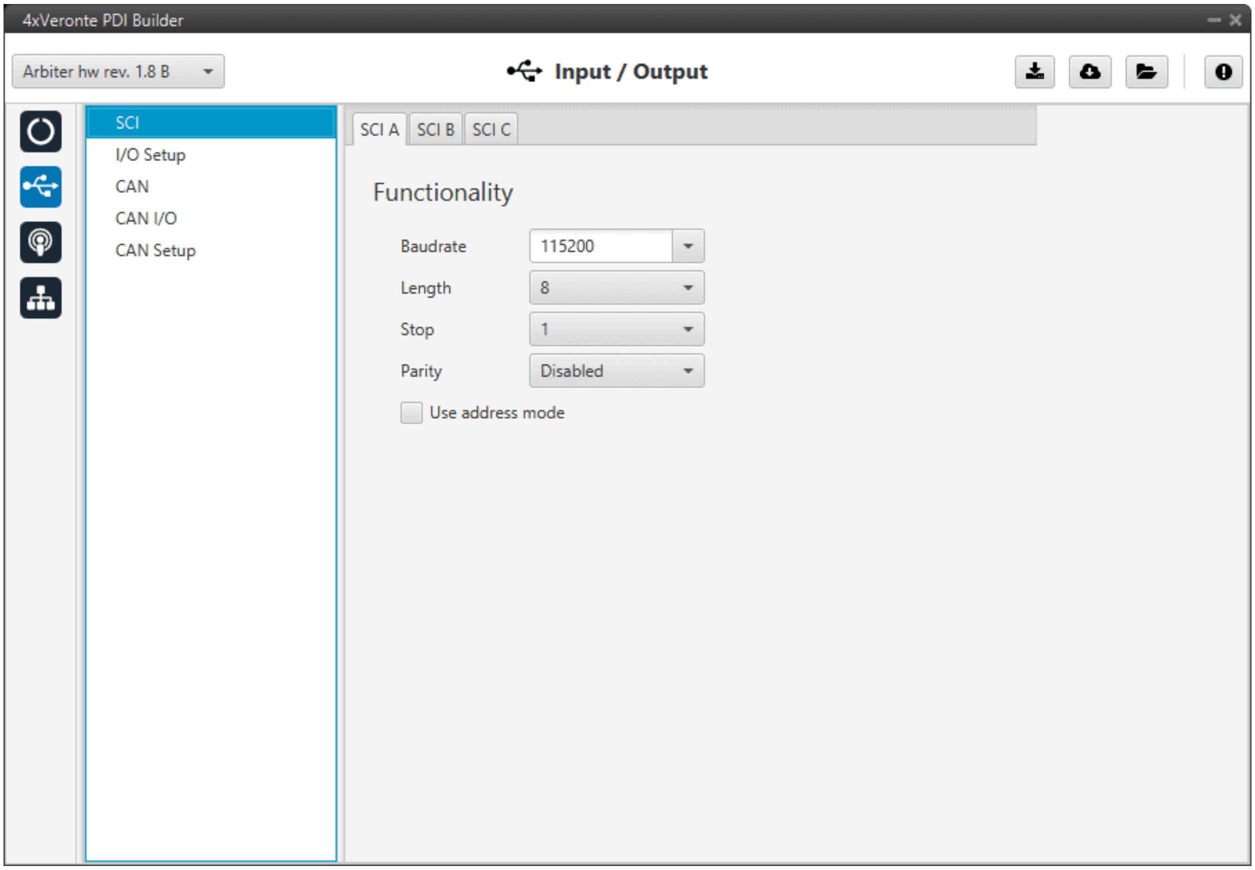
Fig. 15: **Status panel**

2.2 Input/Output

2.2.1 SCI

Arbiters can use up to three serial SCIs (Serial Communication Interface). Serial ports A, B and C parameters can be edited in this menu to fit the serial protocol requirements.

Note: SCI A corresponds to RS-232 port A (RS232-1), SCI B to RS-232 port B (RS232-2) and SCI C to RS-485.



Functionality

Baudrate: 115200

Length: 8

Stop: 1

Parity: Disabled

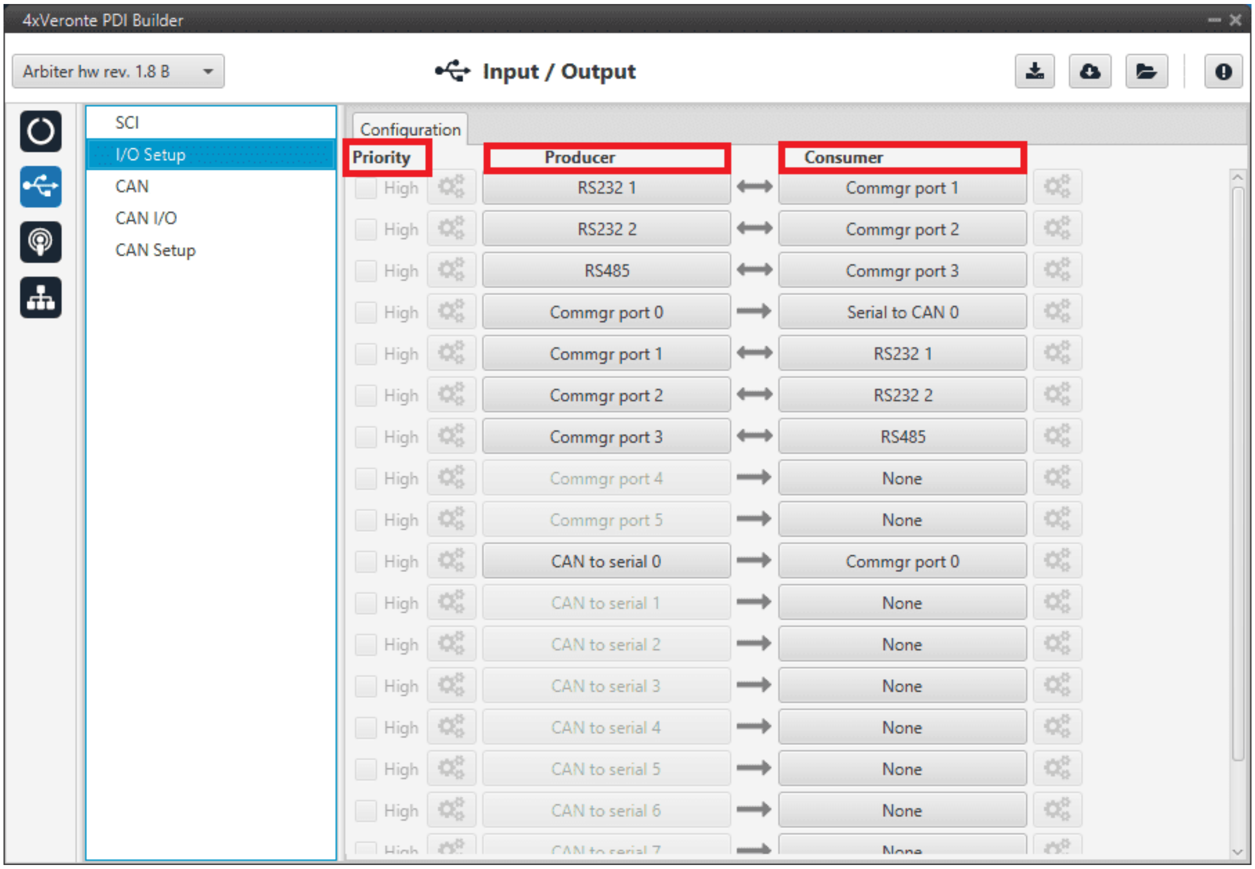
□ Use address mode

Fig. 16: **SCI panel**

- **Baudrate**: How fast data is sent over a serial line.
- **Length**: Number of data bits for each character: 4 to 8 bits.
- **Stop**: Number of stop bits sent at the end of each character: 1, 1.5 or 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**.
- **Use address mode**: 9-bit data framing uses the bit typically associated with parity error detection to identify address messages. Sent serial data that does not have the address bit set will be ignored (unless the device had previously identified an address message associated with it). This option can be disabled or enabled.

2.2.2 I/O Setup

In this panel the user can establish the relationship between a determined signal with a I/O port. This allows users to configure external sensors, custom messages, etc.



Priority	Producer		Consumer
□ High	RS232 1	↔	Commgr port 1
□ High	RS232 2	↔	Commgr port 2
□ High	RS485	↔	Commgr port 3
□ High	Commgr port 0	→	Serial to CAN 0
□ High	Commgr port 1	↔	RS232 1
□ High	Commgr port 2	↔	RS232 2
□ High	Commgr port 3	↔	RS485
□ High	Commgr port 4	→	None
□ High	Commgr port 5	→	None
□ High	CAN to serial 0	→	Commgr port 0
□ High	CAN to serial 1	→	None
□ High	CAN to serial 2	→	None
□ High	CAN to serial 3	→	None
□ High	CAN to serial 4	→	None
□ High	CAN to serial 5	→	None
□ High	CAN to serial 6	→	None
□ High	CAN to serial 7	→	None

Fig. 17: **I/O Setup panel**

- **Priority**: Certain connections between I/O ports can be marked with **high priority** with this checkbox. If it is enabled, they will **run at high frequency: 1000 Hz**.
- **Producer**: Functions to create and send messages.
- **Consumer**: Functions to receive and parse messages.

The following are the steps to setting up reception or transmission between ports:

1. Choose the **Producer** to use.
2. To configure the desired **Consumer** that will be bind to the chosen **Producer**, it is first required to establish the **relationship** between them:
   - **Bind** →: Unidirectional relationship.
   - **Bind Bidirectional** ↔: Bidirectional relationship. This enables a port to receive or send information.

   Note: Once the **Consumer** has been selected, it is possible to undo the selection by pressing the **Clear** button.

3. Select the desired **Consumer** element.

The following I/O ports are available:

Field	Description
RS232 1	Serial Port 232 A
RS232 2	Serial Port 232 B
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, read the VCP user manual
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 for serial transmission / CAN message unwrapper for serial transmission	**CAN wrapper for serial transmission** sends CAN streams over a serial Bus / **CAN message unwrapper for serial transmission** undoes this transformation. For more information on these ports, please refer to CAN wrapper/CAN unwrapper - Input/Output section of the **1x PDI Builder** user manual

Warning: By default, RS232 1-2 are set up for **Arbiter** configuration.

If these connections are removed, it will not be possible to communicate with the **Arbiter** in **Normal mode**. In this situation, force the **arbiter** into **Maintenance mode** in order to recover the communication.

2.2.3 CAN

A CAN (Controller Area Network) Bus is a robust standard communication protocol for vehicles widely used in the aviation sector. The **Management Board** has two CAN buses that can be configured independently.

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

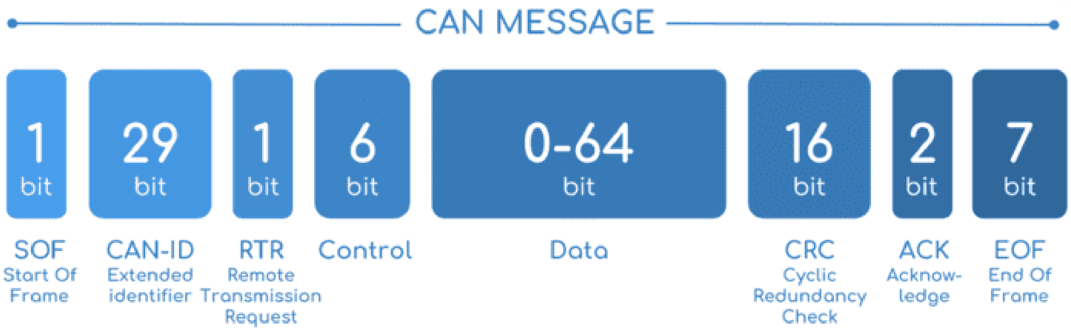
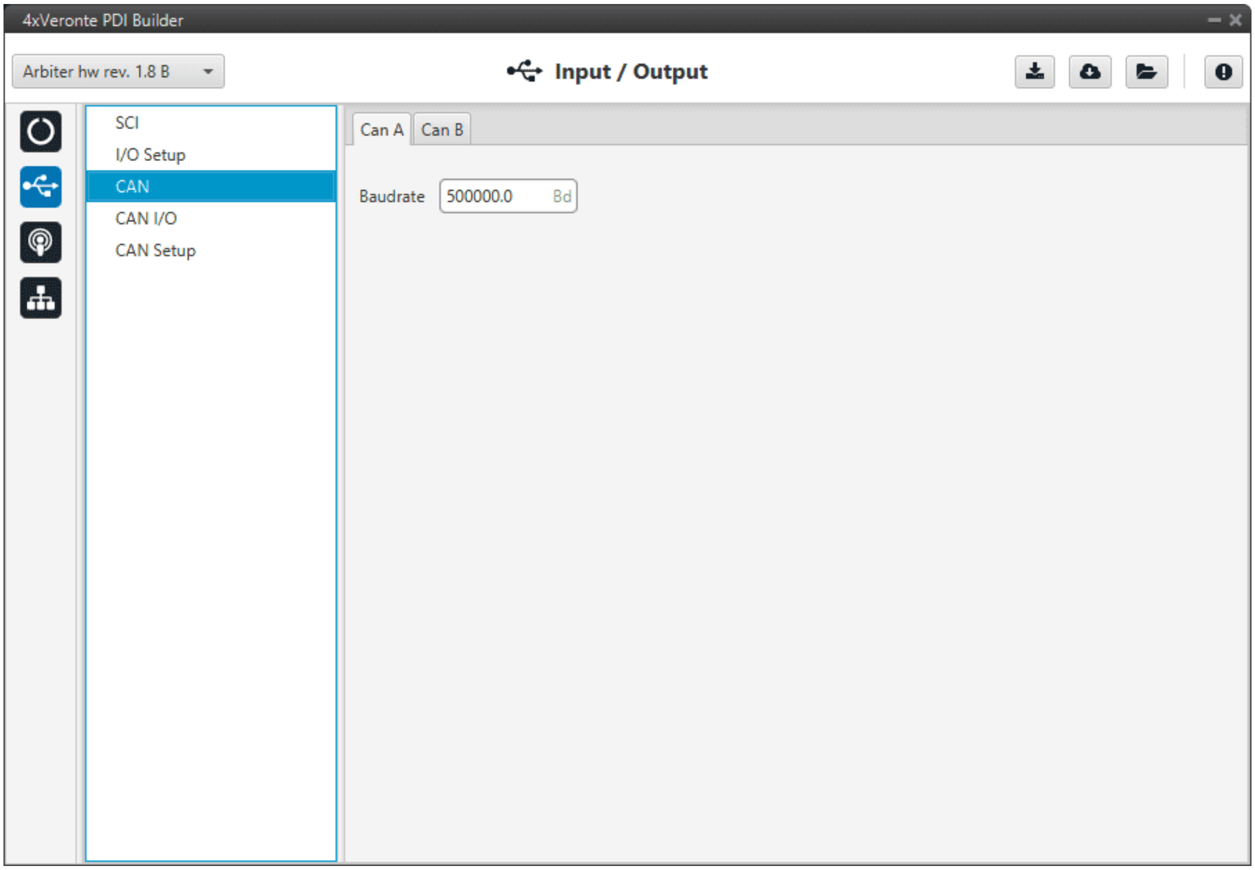


Fig. 18: **CAN message structure**

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

For more information on the CAN Bus protocol, see CAN Bus protocol section of the **4x Software Manual**.

The baudrate of both CAN buses can be configured in the **Baudrate** box.



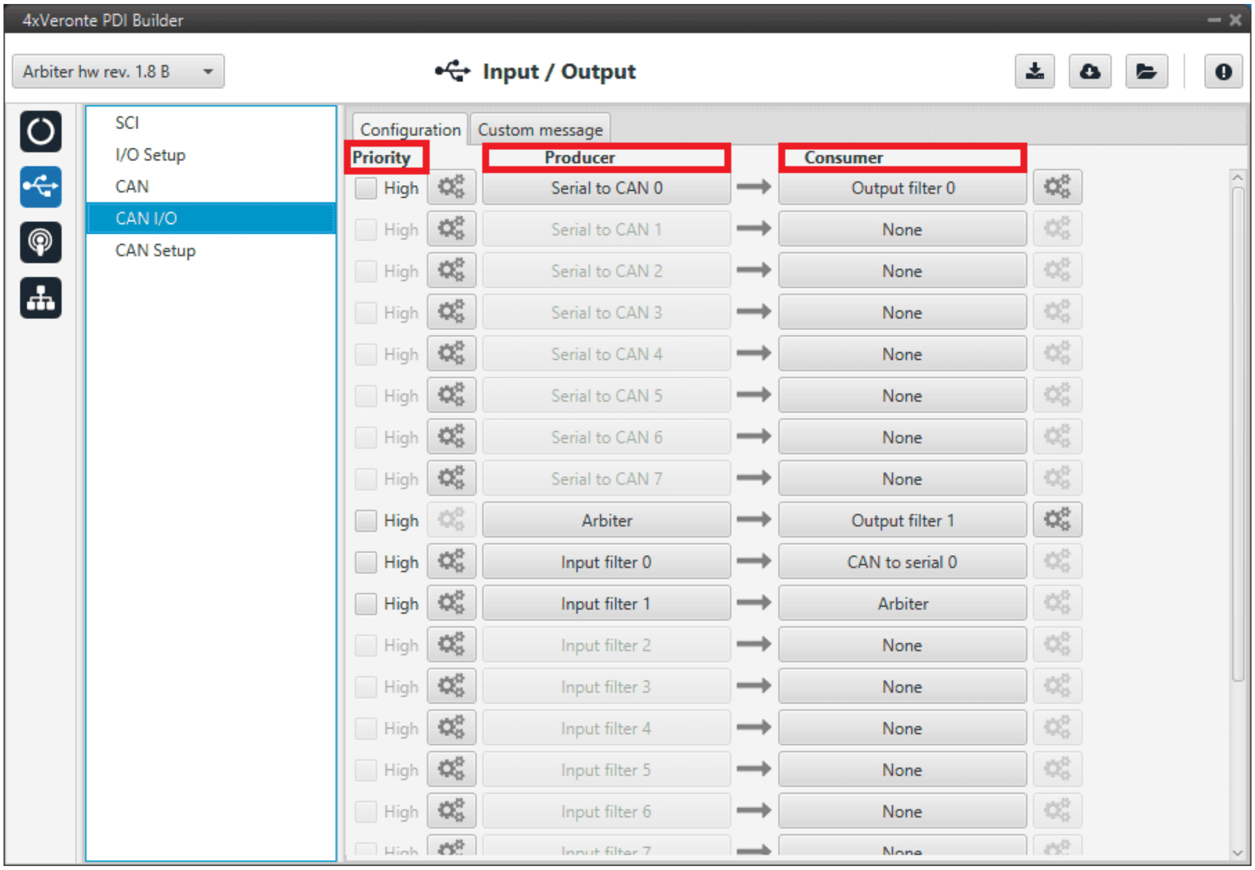
Baudrate 500000.0 Bd

Fig. 19: **CAN panel**

2.2.4 CAN I/O

2.2.4.1 Configuration

This panel allows the configuration of CAN communications between different devices.



Priority	Producer		Consumer
□ High	Serial to CAN 0	→	Output filter 0
□ High	Serial to CAN 1	→	None
□ High	Serial to CAN 2	→	None
□ High	Serial to CAN 3	→	None
□ High	Serial to CAN 4	→	None
□ High	Serial to CAN 5	→	None
□ High	Serial to CAN 6	→	None
□ High	Serial to CAN 7	→	None
□ High	Arbiter	→	Output filter 1
□ High	Input filter 0	→	CAN to serial 0
□ High	Input filter 1	→	Arbiter
□ High	Input filter 2	→	None
□ High	Input filter 3	→	None
□ High	Input filter 4	→	None
□ High	Input filter 5	→	None
□ High	Input filter 6	→	None
□ High	Input filter 7	→	None

Fig. 20: **Configuration panel**

In this panel, the user can find the same 'columns' (**Priority**, **Producer** and **Consumer**) as in the *I/O Setup panel*. In addition, the process for configuring producers and consumers is also the same as described in the *I/O Setup - Input/Output* section.

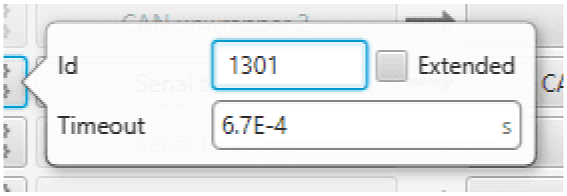
> **Warning:** In CAN, in Low state the specified period is not guaranteed, but it is guaranteed with High state.
>
> However, only those **messages that are critical for external devices** should be set as **high priority**, as this may disrupt the proper functioning of the **arbiter**.

Producers:

- **Arbiter**: It sends a **status** and a **score** mesage.

  To know more, read Arbitration messages - CAN Bus protocol section of the **4x Software Manual**.

- **Serial to CAN**: Serial messages through CAN output, it has to be connected to *I/O Setup* **consumer**. It can be configured by clicking on ⚙, a pop-up window will appear:



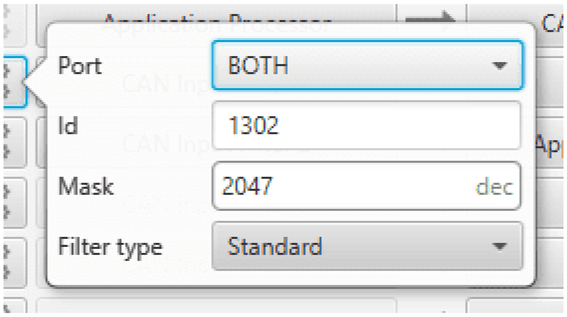
Id 1301 □ Extended

Timeout 6.7E-4 s

Fig. 21: **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 it is enabled, the frame format will be '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.

- **Input filter**: Those CAN messages received in one filter can no longer be received in subsequent filters. The following parameters need to be configured by clicking on ⚙:



Port BOTH

Id 1302

Mask 2047 dec

Filter type Standard

Fig. 22: **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**: A CAN Id mask can be set to filter messages. **The mask defines which bits of the message id (in binary) should match**.

  For example, to admit standard Ids (11 bits) from 8 to 11 (100 to 111 in binary) the user should set the mask to binary 11111111100, that is 2044 in decimal.

  > **Warning:** Make sure that mask is set properly to be able to receive the desired CAN messages.
  >
  > The mask should be **11 bits for Standard** frame format and **29 bits for Extended** frame format.
  >
  > More information about this can be found in *How to calculate a mask - FAQ* section of this manual.

- **Filter type**: The available options are *Standard* (frame format with a 11-bit identifier), *Extended* (frame format with a 29-bit identifier) and *Both*.

- **Custom messages**: CAN custom messages transmission. They are configured in the next panel, explained in thr *Custom message* section of the present manual.
- **CAN unwrapper**: This undoes the 'CAN wrapper' action, it has to be connected to *I/O Setup* **consumer** (*CAN unwrapper for serial transmission*)..

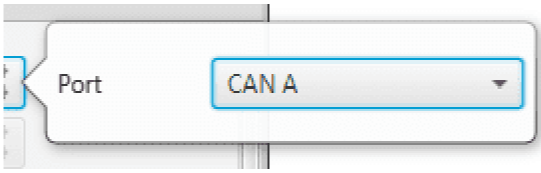
Consumers:

- **Arbiter**: It sends **ready** and **arbitration** messages.

  To know more, read Arbitration messages - CAN Bus protocol section of the **4x Software Manual**.

- **CAN to serial**: This undoes the 'Serial to CAN' action, it has to be connected to *I/O Setup* **producer**.

- **Output filter**: CAN output filters. The user can choose between *CAN A*, *CAN B* or *BOTH* in the **configuration button** ⚙.



Port CAN A

Fig. 23: **Output filter configuration**

- **Custom messages**: CAN custom messages reception. They are configured in the next panel, explained in thr *Custom message* section of the present manual.

- **CAN GPIO consumer:** CAN message from **Veronte Autopilot 1x** for GPIO inputs. An example of how to implement it can be found in the *GPIO command - Integration examples* section of this manual.

- **CAN wrapper**: CAN message through serial output, it has to be connected to *I/O Setup* **producer** (*CAN wrapper for serial transmission*).

> **Warning:** The **Input filter** connected to the **Arbiter consumer** must have a mask that allows all arbitration CAN IDs.

2.2.4.2 Custom message

In this tab, the user chooses the variables to be sent/received through CAN buses. The following elements can be configured:

- **TX Ini**: It is used to configure transmitted messages that are only sent once at the beginning of the operation (sent when the **arbiter** boots up). They can be used to initialize some devices.

- **TX**: Configuration for transmitted messages.

- **RX**: Configuration for 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.
> - Each **arbiter** has a **CAN limitation** of **40 TX** messages, **40 TX Ini** messages and **80 RX** messages. In addition, the following limits apply:
>   - Maximum number of **vectors** (fieldset): **104**
>   - Maximum number of **fields: 2000**

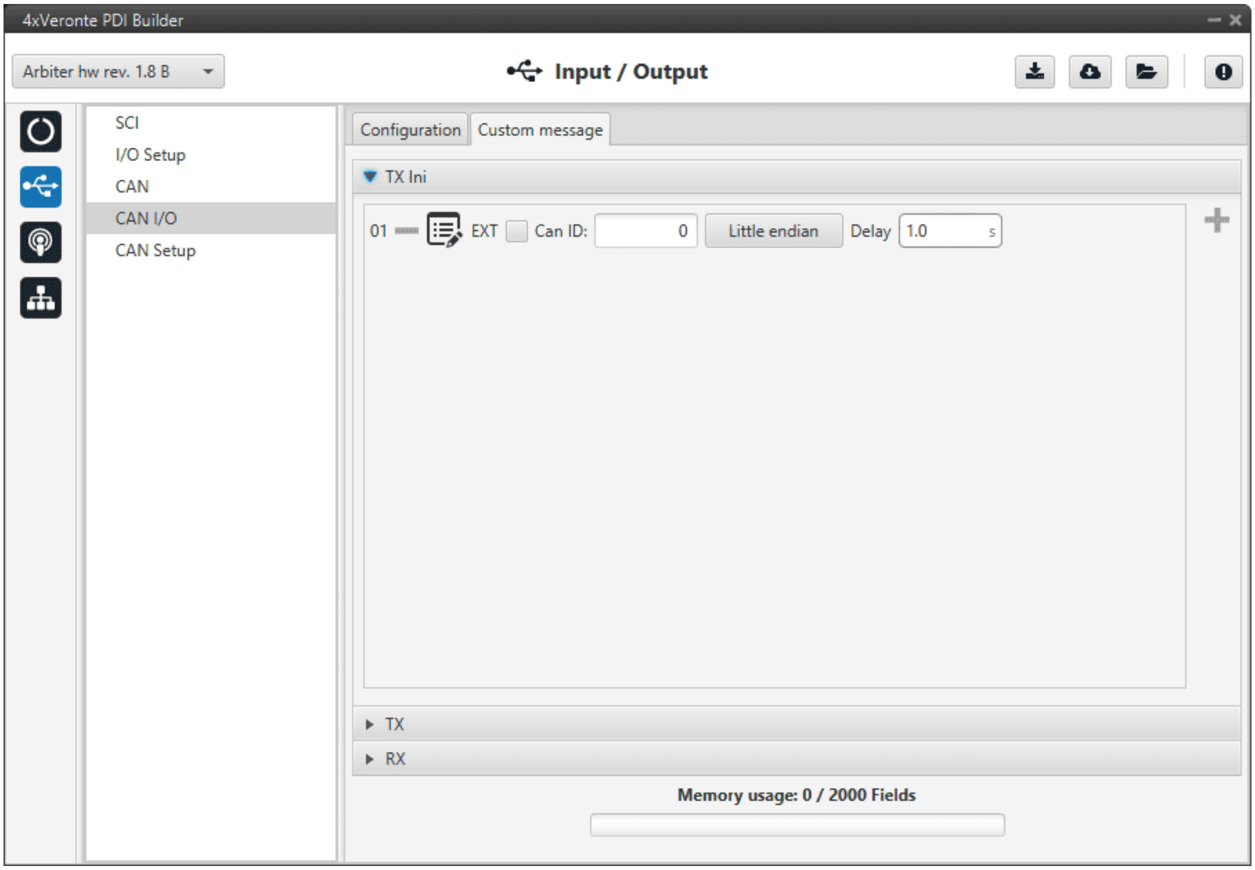


Fig. 24: **Custom message panel**

Since this section works in a similar way to the CAN Custom Message configuration in the **1x PDI Builder** software, the explanation on how to configure the telemetry messages via CAN is reflected in the Custom Messages - Input/Output section of the **1x PDI Builder** user manual.

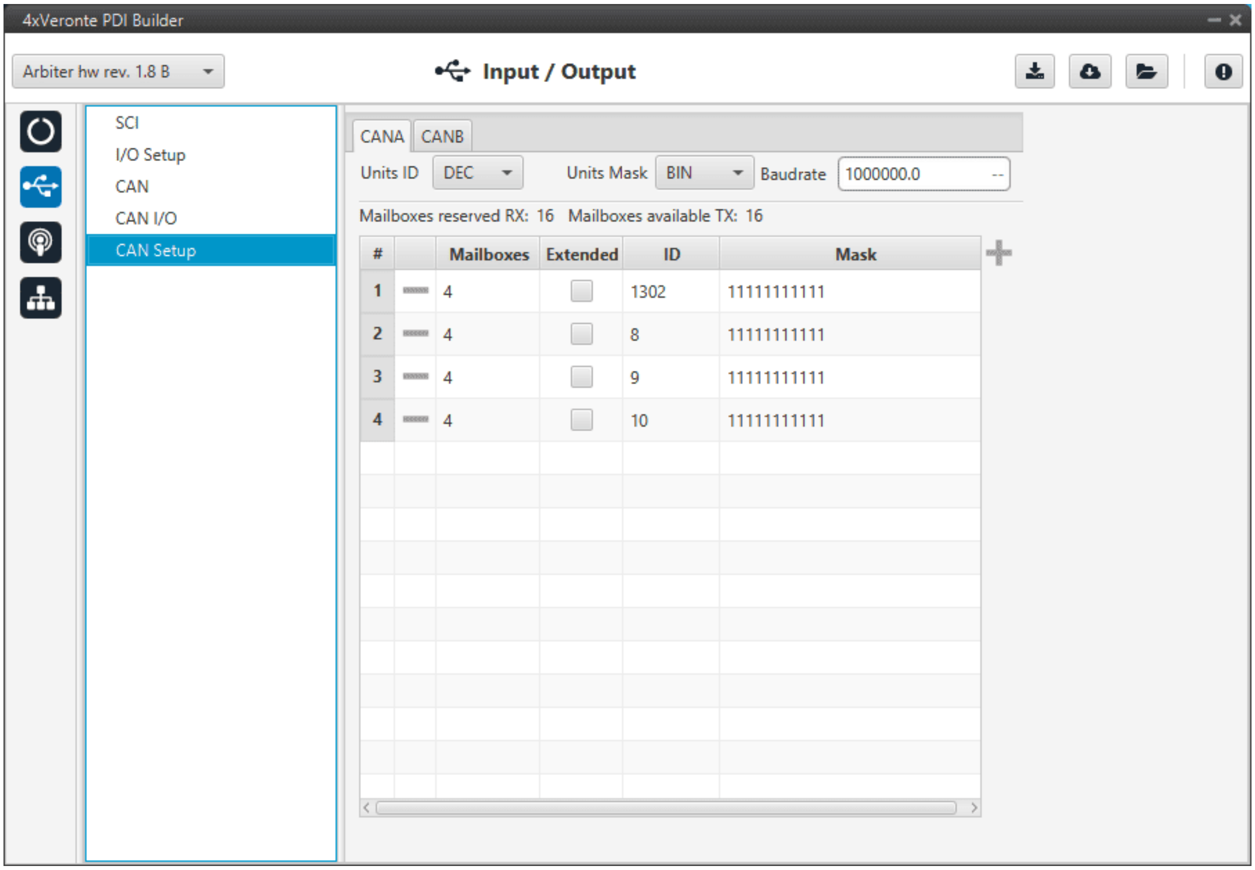
2.2.5 CAN Setup

In this screen, users can configure the **baudrate** and **reception mailboxes** of each **CAN Bus**.

Since the **arbiter** is going to receive data on the CAN Bus, it is mandatory to configure a certain number of mailboxes to store that data until the arbiter reads it.

A mailbox can be configured for **multiple CAN message IDs** as long as the mask is configured correctly and these messages are **sent spaced out** with enough time between them to allow the high priority core to read each one individually. More information on masks can be found in *How to calculate a mask - FAQ* section of this manual.

> **Warning:** Since **4x PDI Builder allows up to 32 mailboxes**, users should make sure to **leave at least one mailbox free for transmission (TX)**.
>
> If any mailbox is **full** and another message arrives, the **new** message is **discarded**.



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

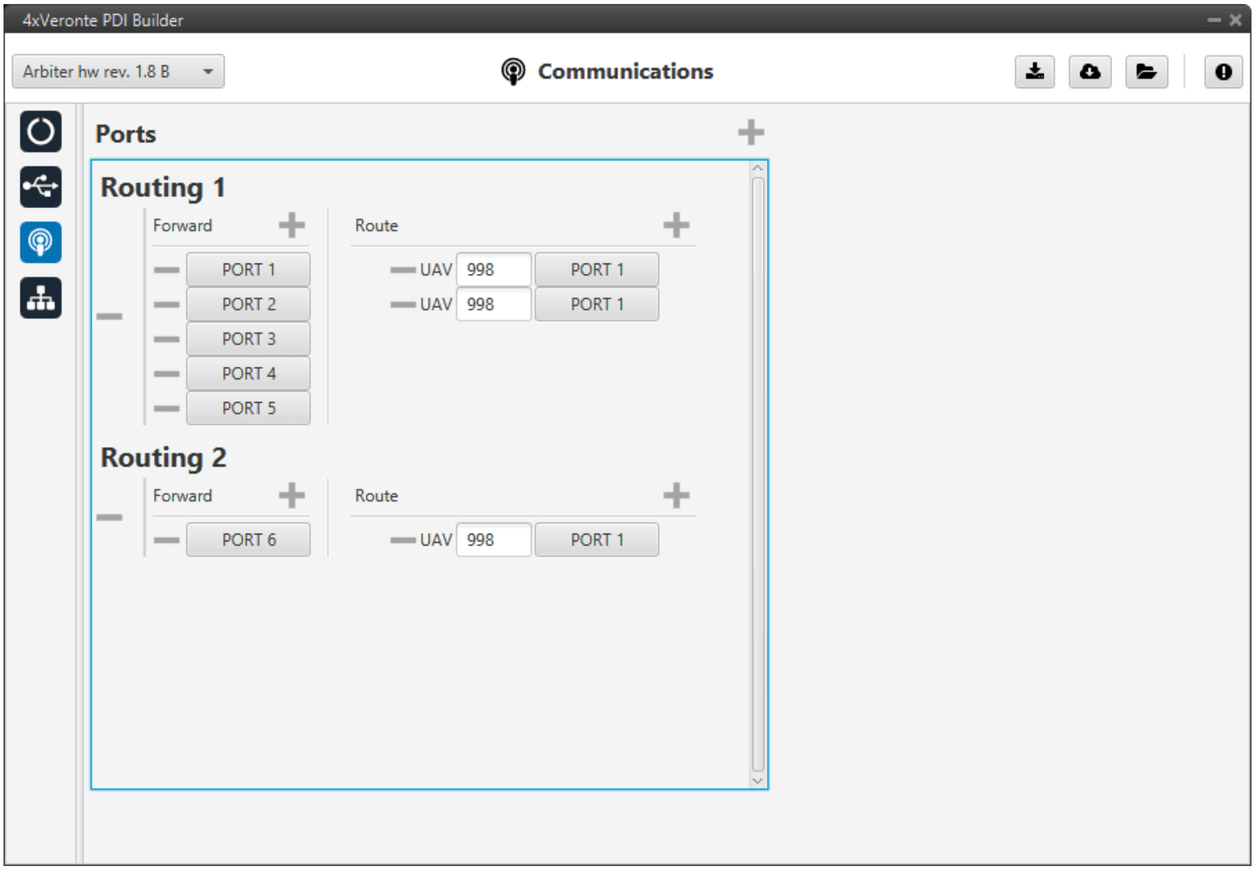
Fig. 25: **CAN Setup panel**

More information about mailboxes can be found in the Mailboxes - Input/Output section of the **1x PDI Builder** user manual.

2.3 Communications

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, **arbiters** can be configured to **route** VCP messages for an external Veronte device with a known address (ID) through a given port.

Note: To know more about VCP messages, read the VCP user manual.



Routing 1

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

Route: UAV 998 PORT 1; UAV 998 PORT 1

Routing 2

Forward: PORT 6

Route: UAV 998 PORT 1

Fig. 26: **Communications panel**

Each port can be configured with the following options:

- **Forward**: Any message generated by this unit (i.e. telemetry or response messages to certain commands) will be sent through these ports.
- **Route**: Any message 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 using the Ports action of **1x PDI Builder** software. Routing 1 will always be selected by default when booting an **arbiter**.

Warning: An incorrect port configuration can disable USB communications. If this happens, the **arbiter** will not be detected through Veronte Link.

If this is the case, please visit How to enter in maintenance mode - Troubleshooting section of the **4x Hardware Manual**.

2.4 Arbitration

The arbitration algorithm in **Autopilot 4x** is based on a **scoring system**. Each autopilot must send continuously a set of arbitration variables that will be used by the arbiter in order to calculate the score for each unit. Then, based on the scores and the current arbitration mode, the arbiter will choose to keep the current selected autopilot, or switching to one of the other units.

A **Score** is a 32 bit, single precission, floating-point value. This parameter is first computed resulting in a range between **0** and **0xFFFFFFFF**, where **0** is a perfect score (*Score'*). To achieve a better understanding, *Score'* is converted to a value comprehended between 0 an 100 (*Score*), being **100** the best possible score in the end.

Scores are calculated using the **arbitration variables** received from each autopilot at their dedicated addresses. After receiving the value, the following formula is used to compute the score for their respective unit:

$$score'_i = \frac{\sum_{j}^{N} min\left[\left(\frac{x_{i,j}-\mu_j}{d_{j,max}}\right)^2, 1\right]\cdot w_j}{\sum_{j}^{N} w_j}$$

$$score_i = 100 \cdot (1 - score'_i)$$

Where:

j	Variable index
$x_{i,j}$	*j* variable from autopilot *i*
μ_j	*j* variable reference
$d_{j,max}$	Maximum allowed error
w_j	Weight of *j* variable
N	Number of variables
$score'_i$	Score' of unit *i*
$score_i$	Score of unit *i*

Tip: The Arbitration weights should be used to increase or decrease the relevance that a certain arbitration variable has over the calculation of the score.

Any variable in Veronte can be used as an arbitration variable. Depending on the platform, operation, application etc. the more relevant variables can be selected for its use as arbitration references.

2.4.1 Config

2.4.1.1 Absolute Arbitration Variables

Absolute arbitration variables are indicators that are **inherently good** or **bad**, and so they are added directly to the score.

Examples of absolute arbitration variables are **Link Quality**, **GNSS accuracy** or warnings such as **Sensors error** or **Position not fixed**.

2.4.1.2 Relative Arbitration Variables

Relative arbitration variables are **not inherently good** or **bad**, and hence need to be compared against the other autopilots in order to calculate its score contribution.

The contribution to the score from a relative arbitration variable will be its **Deviation** from the **Average** of the same variable from each autopilot.

Examples of relative arbitration variables are **Attitude**, **Position**, measurements from sensors, etc.

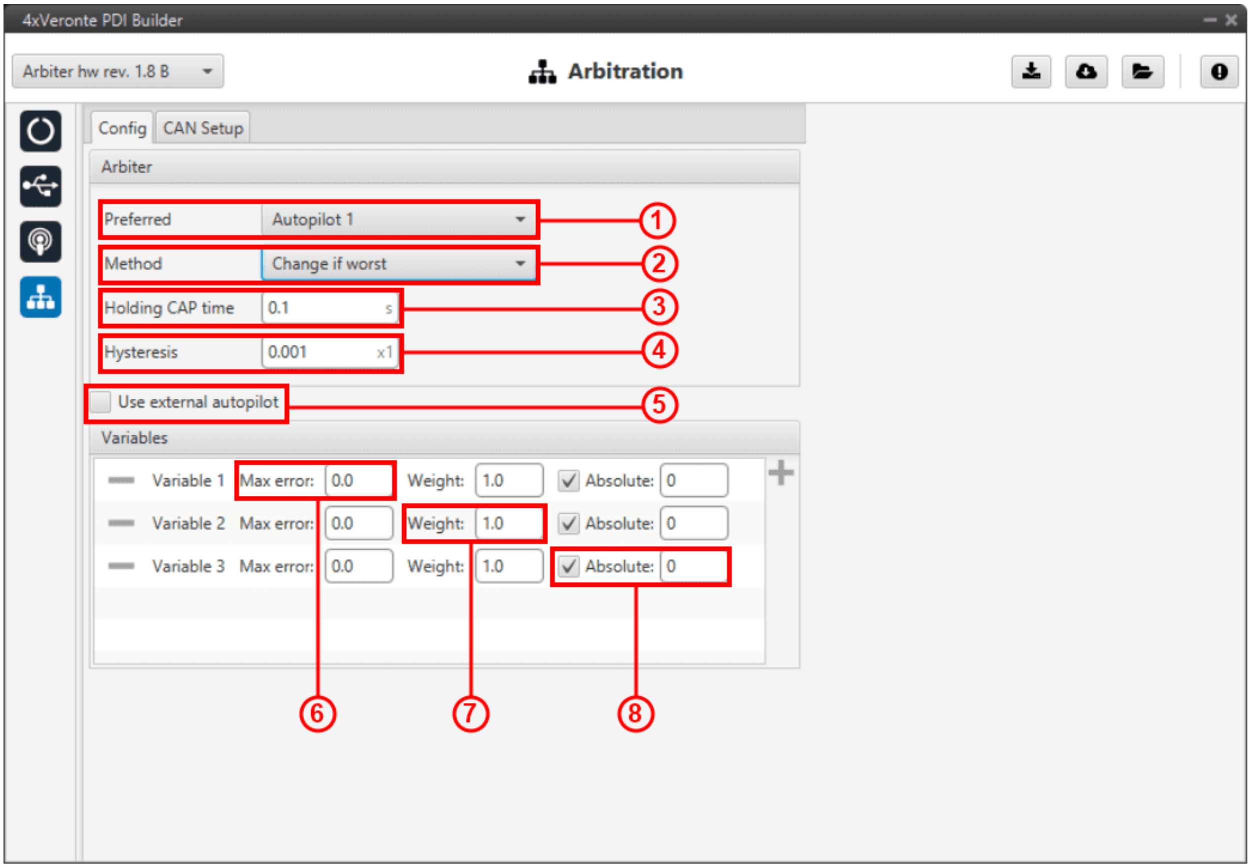
2.4.1.3 Arbitration Example

Autopilot	Var. Nº	Veronte Variable	Type	$x_{i,j}$	μ_j	w_j	Relative score	$Score'_i$	$Score_i$
1	1	Roll	Relative	0.12	0.096	1	0.054	0.043	95.67
	2	Pitch	Relative	0.30	0.283	1	0.027		
	3	GNSS 1 Accuracy	Absolute	1.7		0.01	0.004		
2	1	Roll	Relative	0.10	0.096	1	0.0011	0.0039	99.60
	2	Pitch	Relative	0.28	0.283	1	0.0011		
	3	GNSS 1 Accuracy	Absolute	1.9		0.01	0.005		
3	1	Roll	Relative	0.07	0.096	1	0.071	0.046	95.39
	2	Pitch	Relative	0.27	0.283	1	0.017		
	3	GNSS 1 Accuracy	Absolute	1.5		0.01	0.0036		

In the above example, **AP2** is considered to be **the best**, since it has the **highest total score**. Even though its **GNSS accuracy** is the worst of all 3, its values for pitch and roll are the ones with the lower deviation from the mean.

2.4.1.4 Config menu

In Config menu the parameters of the arbiter algorithm are adjusted.



4xVeronte PDI Builder

Arbiter hw rev. 1.8 B

Arbitration

Config | CAN Setup

Arbiter

Preferred: Autopilot 1

Method: Change if worst

Holding CAP time: 0.1 s

Hysteresis: 0.001 x1

☐ Use external autopilot

Variables

Variable 1 Max error: 0.0 Weight: 1.0 ☑ Absolute: 0

Variable 2 Max error: 0.0 Weight: 1.0 ☑ Absolute: 0

Variable 3 Max error: 0.0 Weight: 1.0 ☑ Absolute: 0

Fig. 27: **Config panel**

1. **Preferred**: The preferred autopilot will be chosen in case of a score draw. **Fixed while ok** mode will always select this autopilot first.
2. **Method**: The method of arbitration can be chosen by the user. The available options are:
   - **Always best**: It chooses always the best autopilot.
   - **Change if worst**: The arbiter will only switch if the currently selected autopilot has the worst score. In that case, it will switch to the one with the best score.
   - **Round robin control**: Using the **Holding CAP** time parameter, the **arbiter** will periodically switch between autopilots. This mode is meant for testing purposes only.
   - **Fixed**: Arbitration is disabled and one autopilot is selected. In this mode **Autopilot 4x** will behave as an **Autopilot 1x**.
     - **Fixed 0**: Autopilot 1 is selected.
     - **Fixed 1**: Autopilot 2 is selected.
     - **Fixed 2**: Autopilot 3 is selected.
   - **Fixed while ok**: This mode does not take into account scores. In this mode, the **Preferred** autopilot will be selected by default. A switch will only happen if the current autopilot is considered **Dead**.
3. **Holding CAP time**: Amount of time needed from last switch in order to allow a new switch.
4. **Hysteresis**: When comparing scores, the difference between them needs to be bigger than this proportional value, in order to assess scores. The difference is proportional to the score of the selected autopilot.

i.e: If current selected autopilot is the number 1, arbitration mode is **Always best**, hysteresis is **0.5** and score for AP 1 is **0.3**, AP 2 will need a score lower than **0.15** in order to be selected.

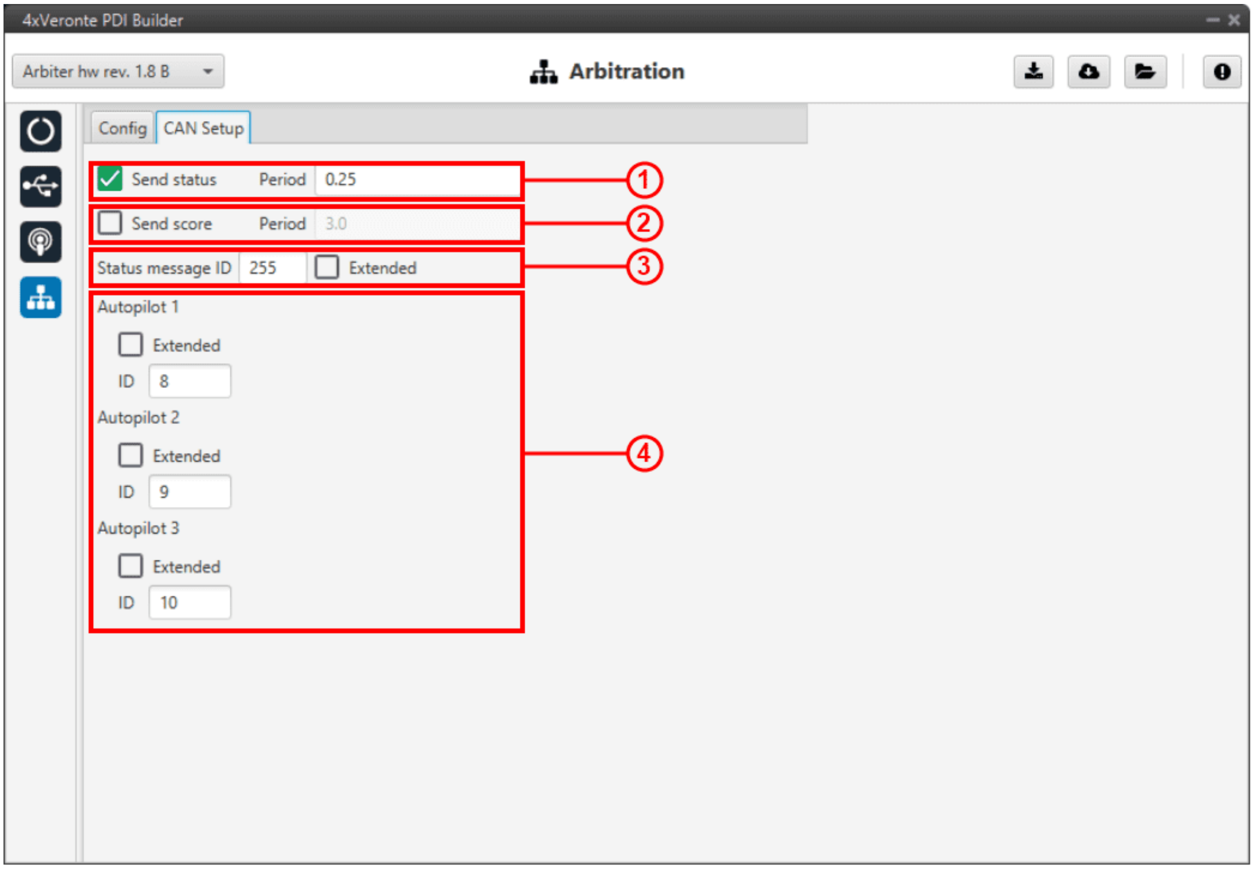
5. Enable arbitration of **external autopilot**.
6. **Variable - Max error**: Arbitration maximum error for each variable.
7. **Variable - Weight**: Arbitration weight for each variable.
8. **Variable - Absolute**: If it is enabled, it will indicate that it is an absolute variable. The value set here will be used to compare the variables in order to choose the best autopilot.

Click on + to add variables and − to delete them. Currently, the **maximum** amount of **arbitration variables** supported is **32**.

The number for each variable is assigned with the **CAN I/O** configuration in the *arbiter consumer*. With this consumer, arbitration variables are selected with their corresponding numbers.

2.4.2 CAN Setup

Here users can configure the received CAN Ids for each one of the 3 possible **Autopilots 1x** that are sending data to **arbiters**. Therefore, to send data from any of them to **arbiters**, these Ids must be specified.



☑ Send status Period 0.25

□ Send score Period 3.0

Status message ID 255 □ Extended

Autopilot 1
□ Extended
ID 8

Autopilot 2
□ Extended
ID 9

Autopilot 3
□ Extended
ID 10

Fig. 28: **CAN Setup panel**

1. **Send status**: The user can enable status sending and set a status message period.
2. **Send score**: Score sending can be enabled and a period of scoring messages can be set.

3. **Status message ID**: CAN ID to which Status and Score messages are sent.

   - **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.

4. **Autopilot 1-3**: CAN IDs used for the reception of autopilot 4x arbitration messages for each AP.

   - **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.

CHAPTER

THREE

INTEGRATION EXAMPLES

3.1 Arbitration configuration

The following arbitration configuration has been done **in conjunction** with the integration example detailed in the Autopilots 1x configuration - Integration examples section of the **1x PDI Builder** user manual. Also, this is the **recommended configuration** and a good starting point.

The following schema broadly summarizes the configuration that will be explained:

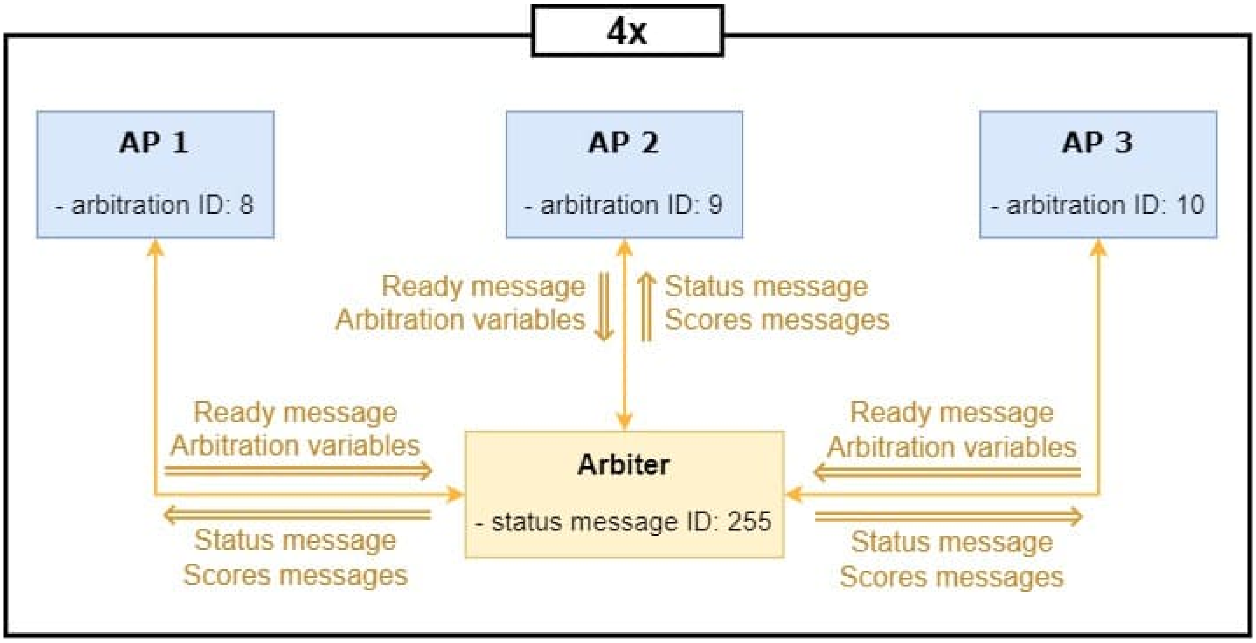


Fig. 1: **Arbitration configuration diagram**

Important: All the **Ids** represented in the schema are the **default** ones and those that will be used throughout this example, but users can change them as they wish.

Nonetheless, the Ids designated for arbitration in this configuration must **match** those entered in the **Autopilots 1x configuration** of the **1x PDI Builder** software.

AP 1 arbitration Id	AP 2 arbitration Id	AP 3 arbitration Id	Status message Id
8	9	10	255

Users should follow the steps below:

1. Go to Input/Output menu → CAN I/O panel → **Configuration tab**.

    - For **sending the status and score messages** to the **Autopilots 1x**, connect the **Arbiter** producer to an **Output filter** consumer, in this case *Output filter 2* has been selected.

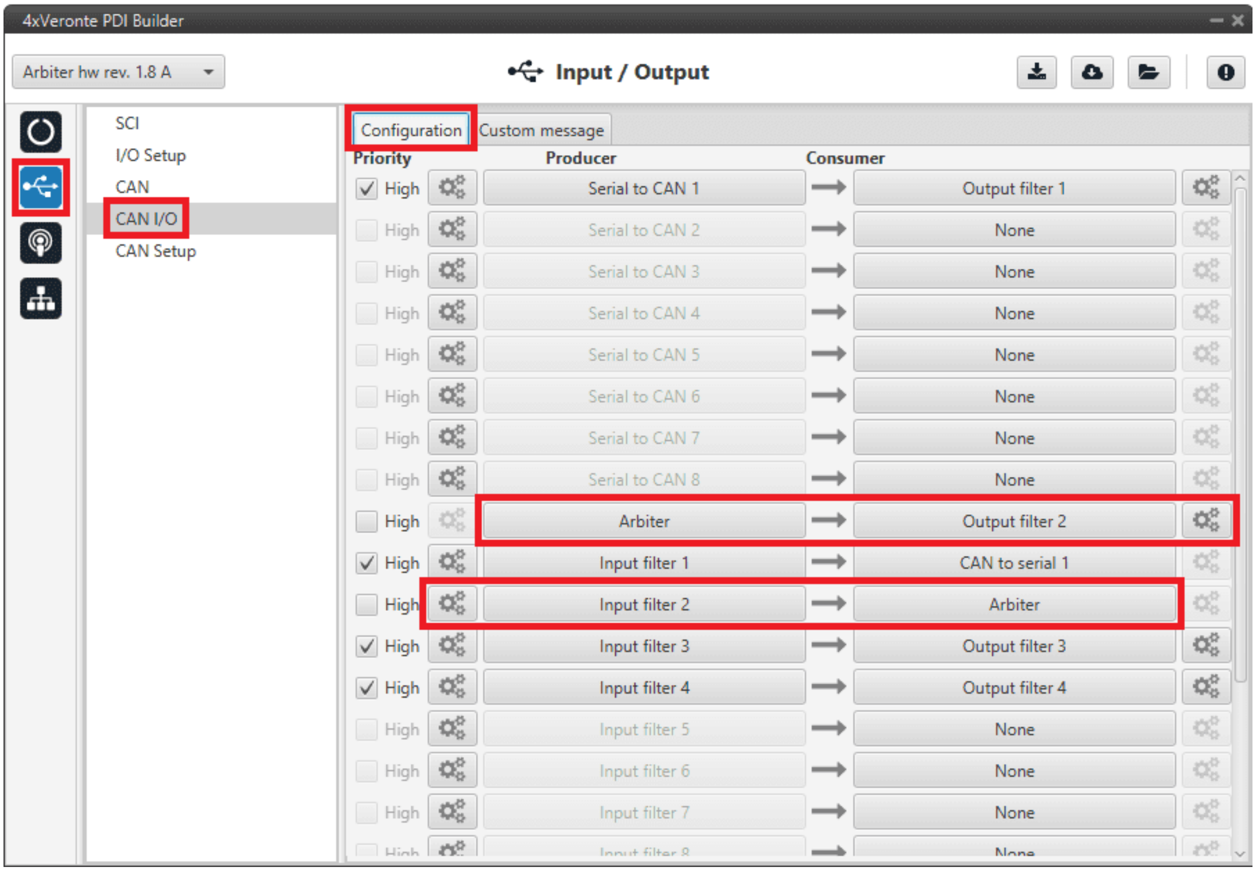
      This **Output filter** must be set to **CAN A** port, as this is the CAN Bus employed by the arbiter to communicate with **Autopilots 1x**. For more information on the CAN architecture of an **Autopilot 4x**, see the General description - Technical section of the **4x Hardware Manual**.

    - For **receiving the ready and arbitration variables messages** from the **Autopilots 1x**, connect an **Input filter** producer to the **Arbiter** consumer, in this case *Input filter 2* has been selected.

      To read these message correctly, it is necessary to configure this Input filter to **CAN A** port, with Id **8** and mask **2044** (decimal format), this will read messages with Ids from **8** to **10**.

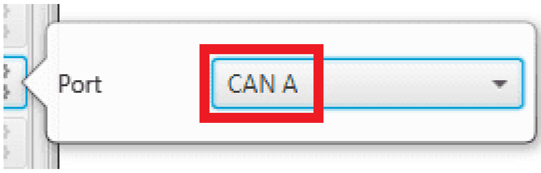
      Note: The following table shows the Ids and mask in binary and decimal format used in this example.

	Decimal format	Binary format
AP 1 arbitration Id	8	000 0000 1000
AP 2 arbitration Id	9	000 0000 1001
AP 3 arbitration Id	10	000 0000 1010
Mask	2044	111 1111 1100



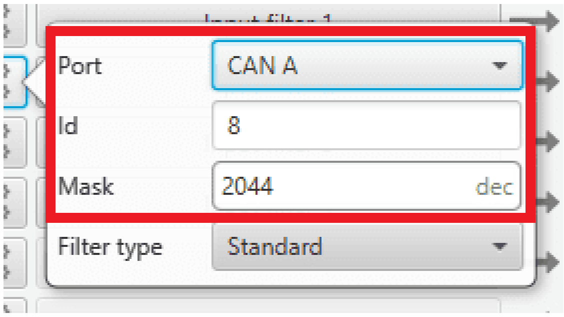
Priority	Producer		Consumer
☑ High	Serial to CAN 1	→	Output filter 1
☐ High	Serial to CAN 2	→	None
☐ High	Serial to CAN 3	→	None
☐ High	Serial to CAN 4	→	None
☐ High	Serial to CAN 5	→	None
☐ High	Serial to CAN 6	→	None
☐ High	Serial to CAN 7	→	None
☐ High	Serial to CAN 8	→	None
☐ High	Arbiter	→	Output filter 2
☑ High	Input filter 1	→	CAN to serial 1
☐ High	Input filter 2	→	Arbiter
☑ High	Input filter 3	→	Output filter 3
☑ High	Input filter 4	→	Output filter 4
☐ High	Input filter 5	→	None
☐ High	Input filter 6	→	None
☐ High	Input filter 7	→	None
☐ High	Input filter 8	→	None

Fig. 2: **Arbitration configuration - CAN I/O configuration**



Port: CAN A

Fig. 3: **Arbitration configuration - Output filter configuration**



Port: CAN A

Id: 8

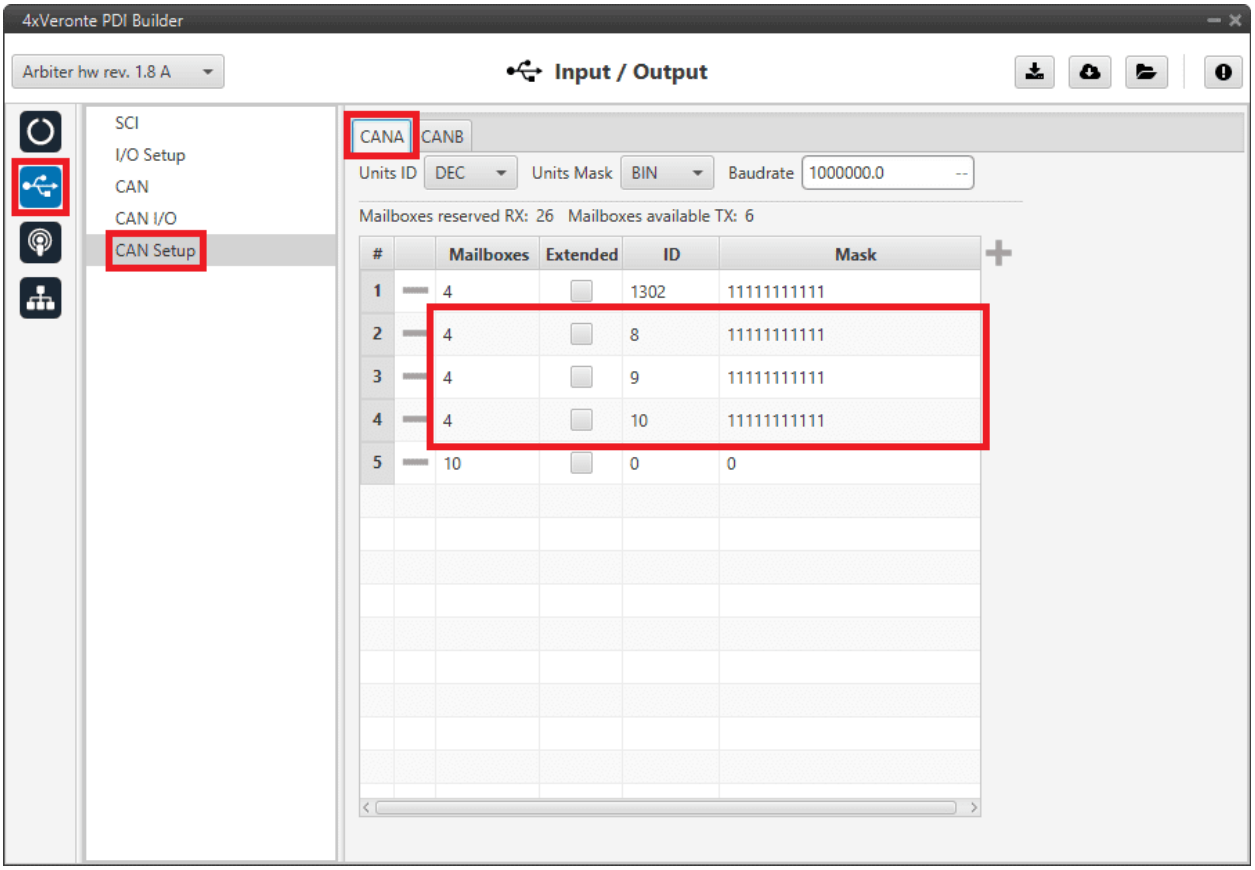
Mask: 2044 dec

Filter type: Standard

Fig. 4: **Arbitration configuration - Input filter configuration**

2. Go to Input/Output menu → CAN Setup panel → **CAN A tab**.

Configure at least **4** mailboxes in **CAN A** for each *arbitration Id*:



Units ID DEC Units Mask BIN Baudrate 1000000.0

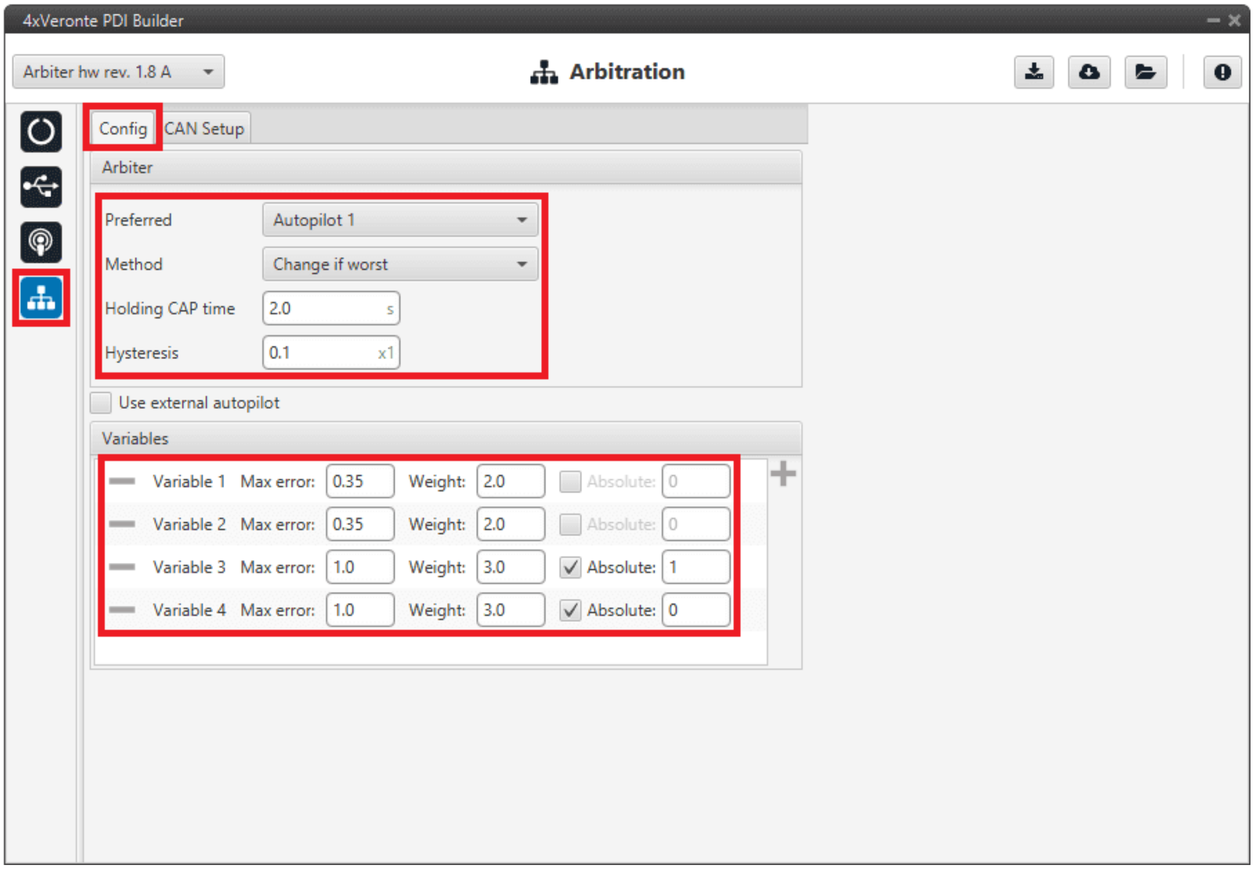
Mailboxes reserved RX: 26 Mailboxes available TX: 6

#		Mailboxes	Extended	ID	Mask
1	—	4	☐	1302	11111111111
2	—	4	☐	8	11111111111
3	—	4	☐	9	11111111111
4	—	4	☐	10	11111111111
5	—	10	☐	0	0

Fig. 5: **Arbitration configuration - CAN Setup configuration**

3. Go to Arbitration menu → **Config panel**.

   Here users have to configure the necessary parameters for the arbitration algorithm. For more information on this algorithm, see *Arbitration* section of the present manual.



Preferred: Autopilot 1

Method: Change if worst

Holding CAP time: 2.0 s

Hysteresis: 0.1 x1

☐ Use external autopilot

Variable 1 Max error: 0.35 Weight: 2.0 ☐ Absolute: 0

Variable 2 Max error: 0.35 Weight: 2.0 ☐ Absolute: 0

Variable 3 Max error: 1.0 Weight: 3.0 ☑ Absolute: 1

Variable 4 Max error: 1.0 Weight: 3.0 ☑ Absolute: 0

Fig. 6: **Arbitration configuration - Config (Arbitration) configuration**

- **Arbiter**: These are the recommended values for each parameter:
  - Preferred: **Autopilot 1**
  - Method: **Change if worst**
  - Holding CAP time: **2.0 s**
  - Hysteresis: **0.1**
- **Variables**: Add as many variables as have been configured in the **1x PDI Builder** software as **arbitration variables**.

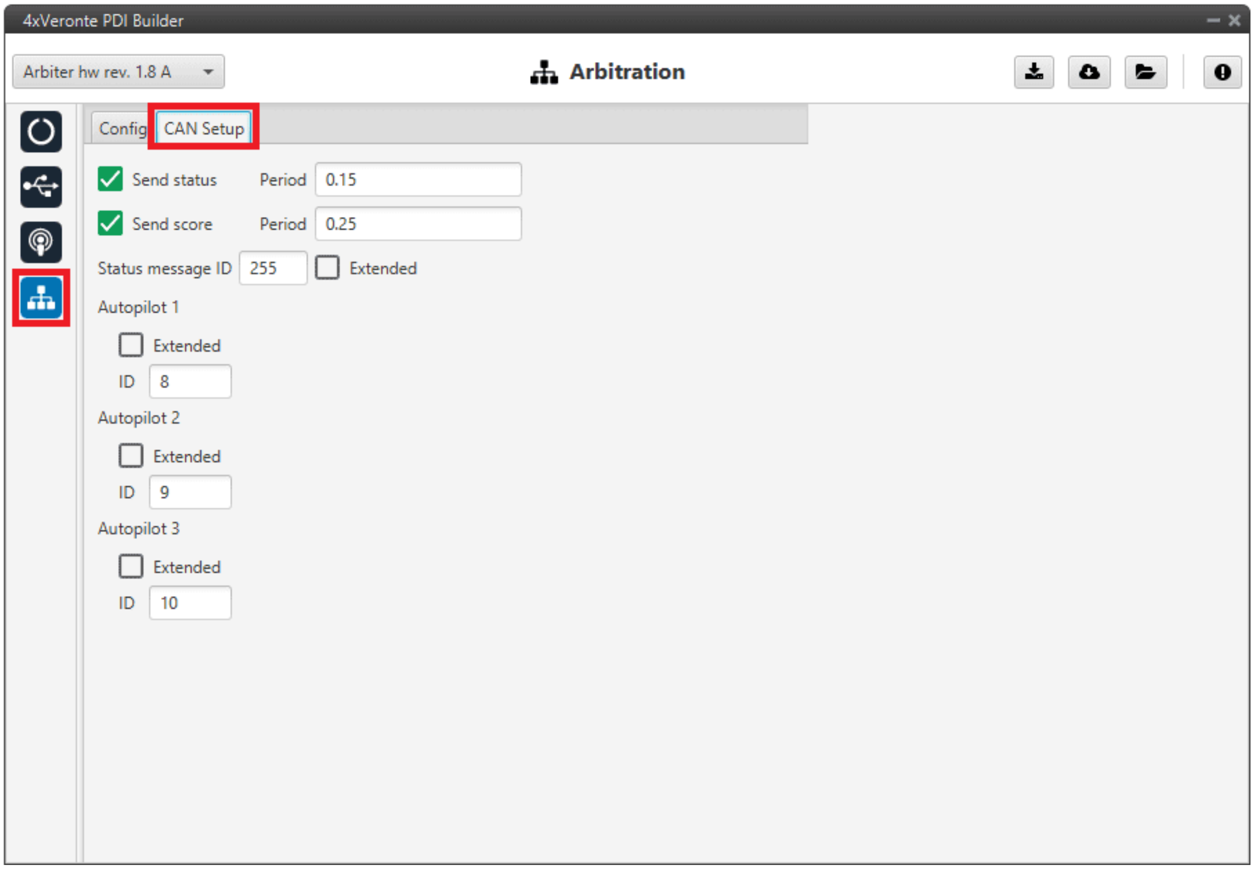
  The variables chosen in this example, as well as their configuration parameters, are presented below:

Variable	Max error	Weight	Absolute
Roll	0.35	2.0	-
Pitch	0.35	2.0	-
Position not fixed	1.0	3.0	Enabled | 1
Custom Arbitration Variable	1.0	3.0	Enabled | 0

Note: Remember that the number for each variable is assigned with the CAN I/O configuration in the *arbiter consumer*. With this consumer, arbitration variables are selected with their corresponding numbers.

4. Go to Arbitration menu → **CAN Setup panel**.

Finally, users must enable the sending of **status** and **scores messages** from the arbiter and configure the CAN Ids described above.



☑ Send status Period 0.15

☑ Send score Period 0.25

Status message ID 255 □ Extended

Autopilot 1

□ Extended

ID 8

Autopilot 2

□ Extended

ID 9

Autopilot 3

□ Extended

ID 10

Fig. 7: **Arbitration configuration - CAN Setup (Arbitration) configuration**

- Send status: **Enabled**
  - Period: **0.15**
- Send score: **Enabled**
  - Period: **0.25**
- **Status message ID**: **255**
- **Autopilot 1**: **ID 8**
- **Autopilot 2**: **ID 9**
- **Autopilot 3**: **ID 10**

3.2 GPIO command

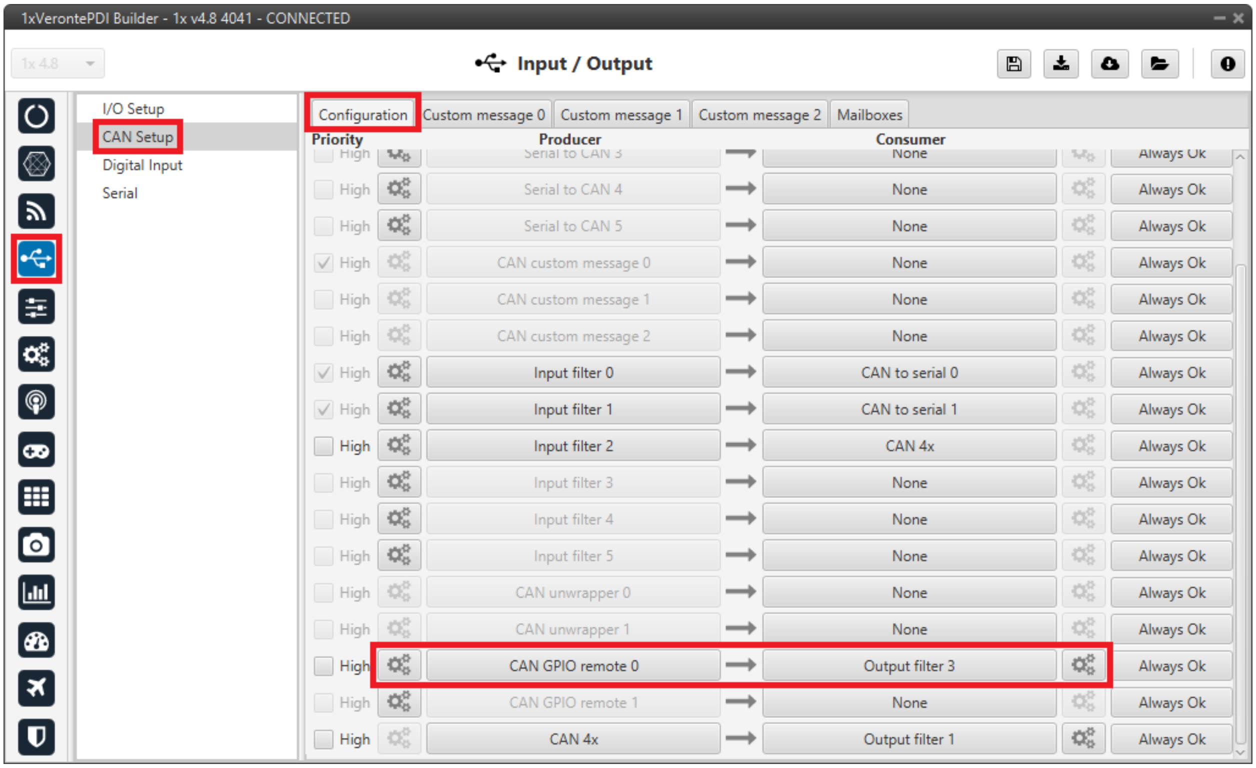
The following are the steps to send a GPIO command from the **Veronte Autopilot 1x**, receive it at **Arbiter** and process it, so that **Arbiter** carries out the command. GPIO Command is very similar to PWM command with a few exceptions.

> **Warning:** For the reception of CAN messages, Mailboxes need to be configured accordingly.

3.2.1 1x PDI Builder side

1. Go to Input/Output menu → CAN Setup panel → **Configuration tab**.

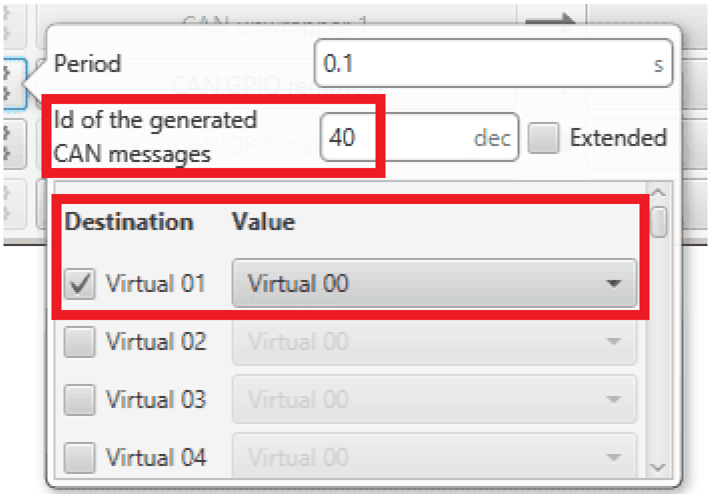
   Connect a **CAN GPIO remote** producer to an **Output filter** consumer:



Priority	Producer	Consumer	
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	CAN custom message 0	None	Always Ok
□ High	CAN custom message 1	None	Always Ok
□ High	CAN custom message 2	None	Always Ok
☑ High	Input filter 0	CAN to serial 0	Always Ok
☑ High	Input filter 1	CAN to serial 1	Always Ok
□ High	Input filter 2	CAN 4x	Always Ok
□ High	Input filter 3	None	Always Ok
□ High	Input filter 4	None	Always Ok
□ High	Input filter 5	None	Always Ok
□ High	CAN unwrapper 0	None	Always Ok
□ High	CAN unwrapper 1	None	Always Ok
□ High	CAN GPIO remote 0	Output filter 3	Always Ok
□ High	CAN GPIO remote 1	None	Always Ok
□ High	CAN 4x	Output filter 1	Always Ok

Fig. 8: **1x PDI Builder - CAN Setup configuration**

CAN GPIO remote must be configured. For more information on its configuration, see CAN Setup - Input/Output section of the **1x PDI Builder** user manual.



Period 0.1 s

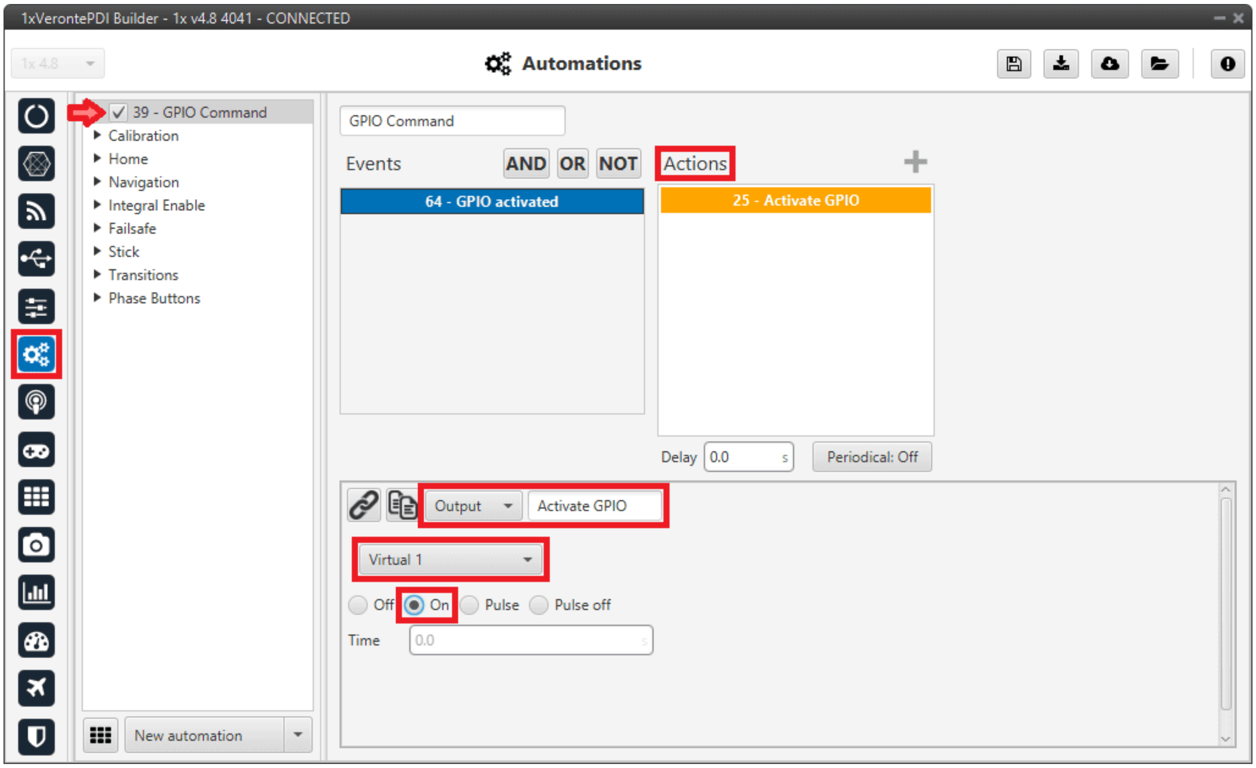
Id of the generated CAN messages 40 dec ☐ Extended

Destination	Value
☑ Virtual 01	Virtual 00
☐ Virtual 02	Virtual 00
☐ Virtual 03	Virtual 00
☐ Virtual 04	Virtual 00

Fig. 9: **1x PDI Builder - CAN GPIO remote configuration**

2. Go to **Automations menu**.

   Create a new automation to activate the GPIO using an **Output action**:



1xVerontePDI Builder - 1x v4.8 4041 - CONNECTED

Automations

GPIO Command

Events: 64 - GPIO activated

Actions: 25 - Activate GPIO

Delay 0.0 s Periodical: Off

Output Activate GPIO

Virtual 1

○ Off ● On ○ Pulse ○ Pulse off

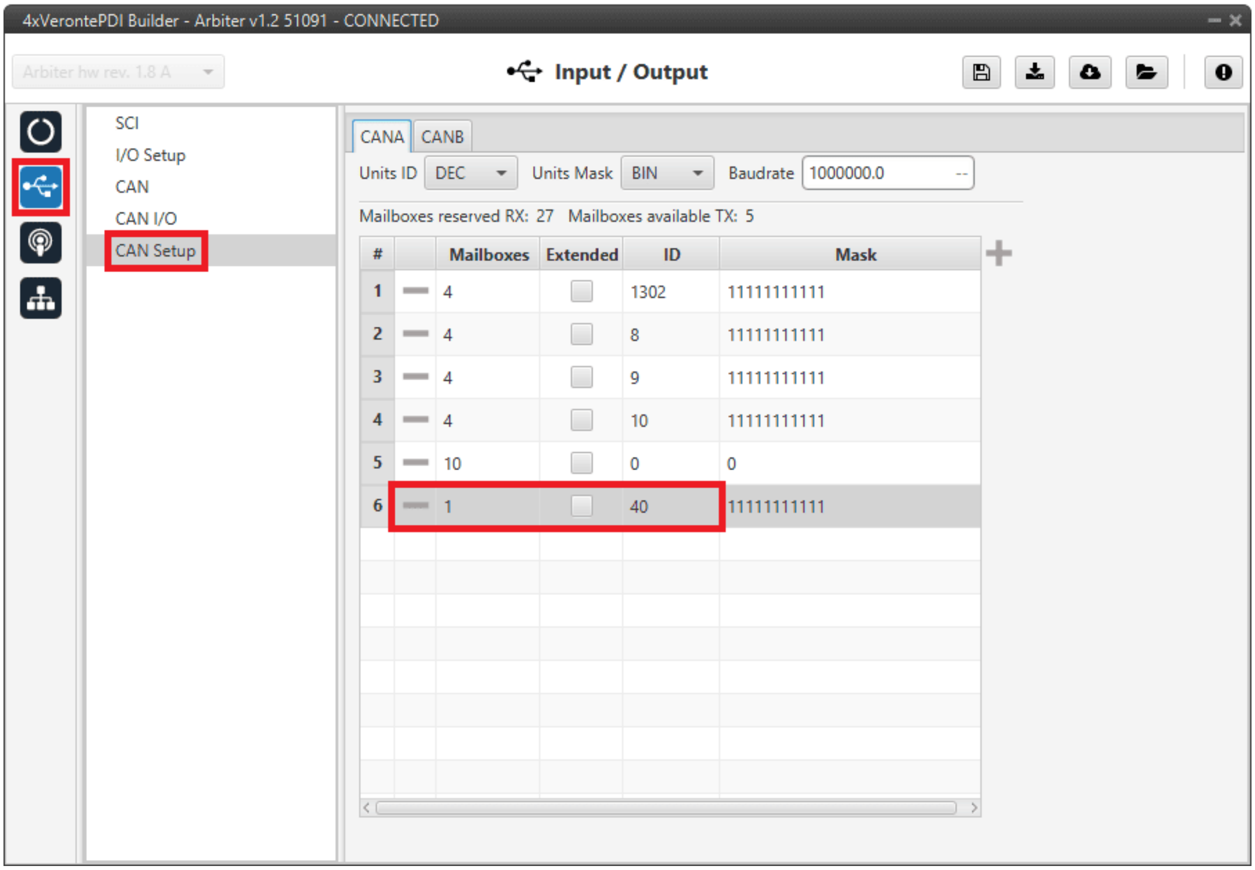
Time 0.0 s

Fig. 10: **1x PDI Builder - Automation configuration**

3.2.2 4x PDI Builder side

3. Go to Input/Output menu → **CAN Setup panel**.

   Create the mailbox to receive the message configured in the **1x PDI Builder** (ID 40):



Units ID: DEC Units Mask: BIN Baudrate: 1000000.0

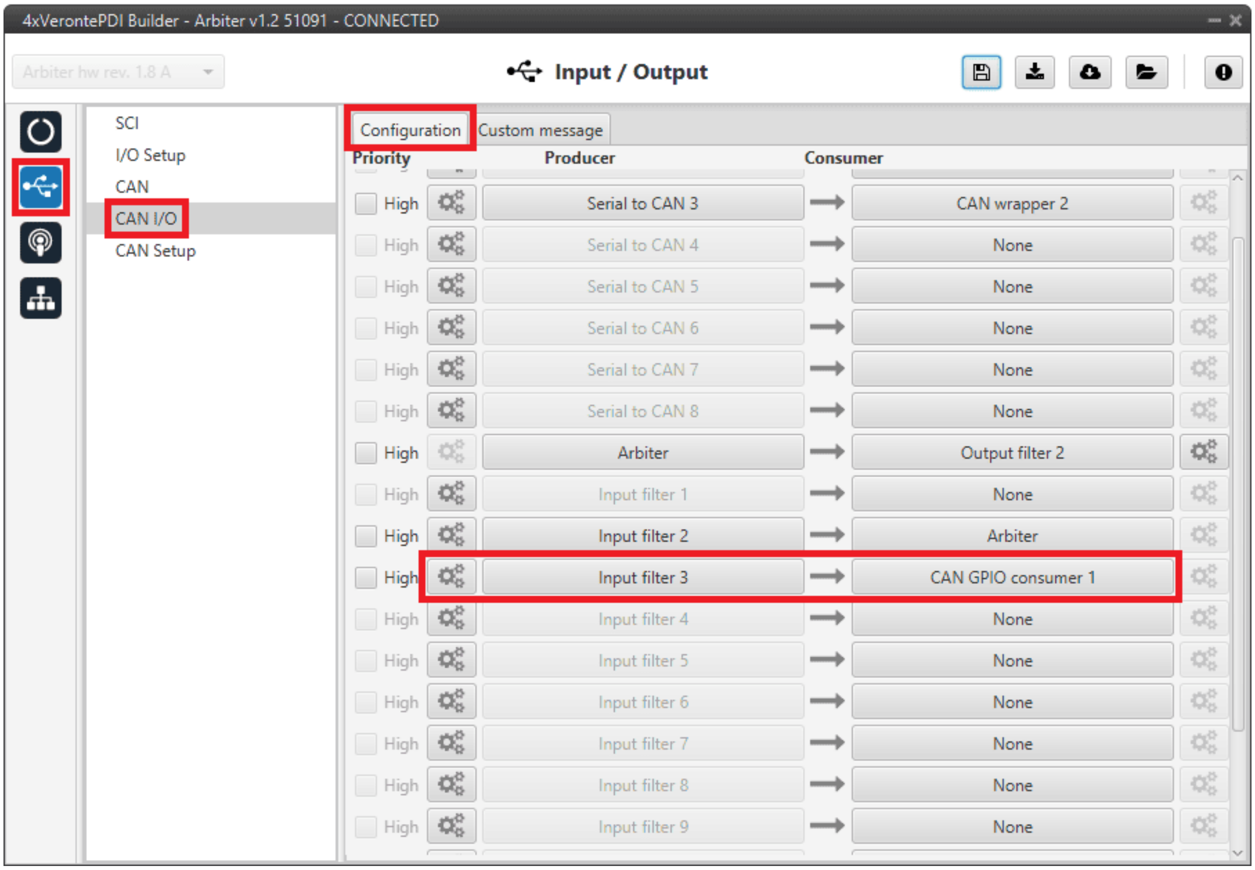
Mailboxes reserved RX: 27 Mailboxes available TX: 5

#	Mailboxes	Extended	ID	Mask
1	4	□	1302	11111111111
2	4	□	8	11111111111
3	4	□	9	11111111111
4	4	□	10	11111111111
5	10	□	0	0
6	1	□	40	11111111111

Fig. 11: **4x PDI Builder - CAN Setup configuration**

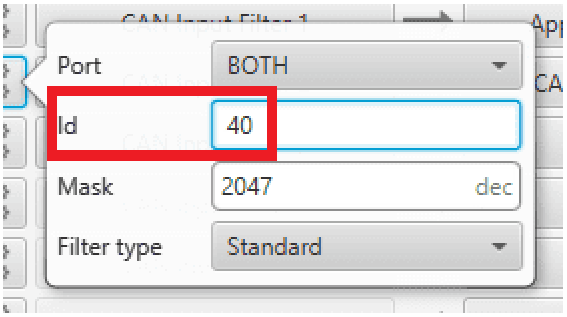
4. Go to Input/Output menu → CAN I/O panel → **Configuration tab**.

   Finally, connect an **Input filter** producer with the **right CAN ID** to a **CAN GPIO consumer**.



	Priority	Producer		Consumer
☐	High	Serial to CAN 3	→	CAN wrapper 2
☐	High	Serial to CAN 4	→	None
☐	High	Serial to CAN 5	→	None
☐	High	Serial to CAN 6	→	None
☐	High	Serial to CAN 7	→	None
☐	High	Serial to CAN 8	→	None
☐	High	Arbiter	→	Output filter 2
☐	High	Input filter 1	→	None
☐	High	Input filter 2	→	Arbiter
☐	High	Input filter 3	→	CAN GPIO consumer 1
☐	High	Input filter 4	→	None
☐	High	Input filter 5	→	None
☐	High	Input filter 6	→	None
☐	High	Input filter 7	→	None
☐	High	Input filter 8	→	None
☐	High	Input filter 9	→	None

Fig. 12: **4x PDI Builder - CAN I/O configuration**



Port: BOTH

Id: 40

Mask: 2047 dec

Filter type: Standard

Fig. 13: **4x PDI Builder - Input filter configuration**

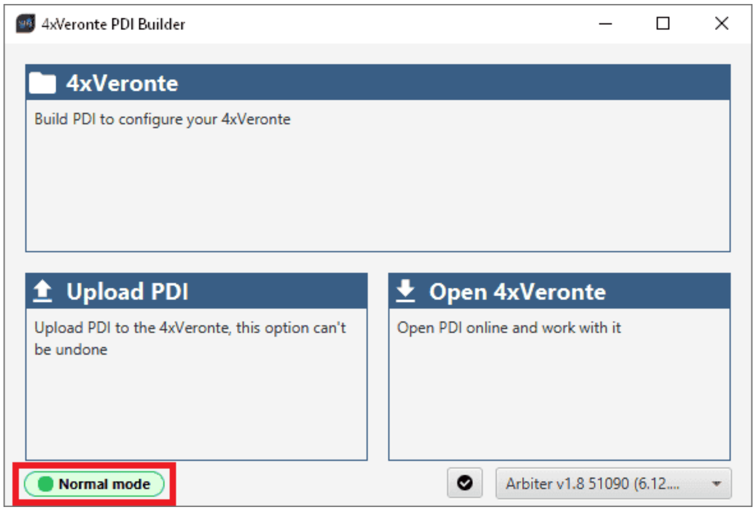
CHAPTER

FOUR

TROUBLESHOOTING

4.1 Maintenance mode

The user can simply enter maintenance mode with **4x PDI Builder** by clicking on the “Normal mode” button in the initial menu. Press the same button to return to “normal mode”.



Build PDI to configure your 4xVeronte

Upload PDI to the 4xVeronte, this option can't be undone

Open PDI online and work with it

Fig. 1: **Enter in maintenance mode**

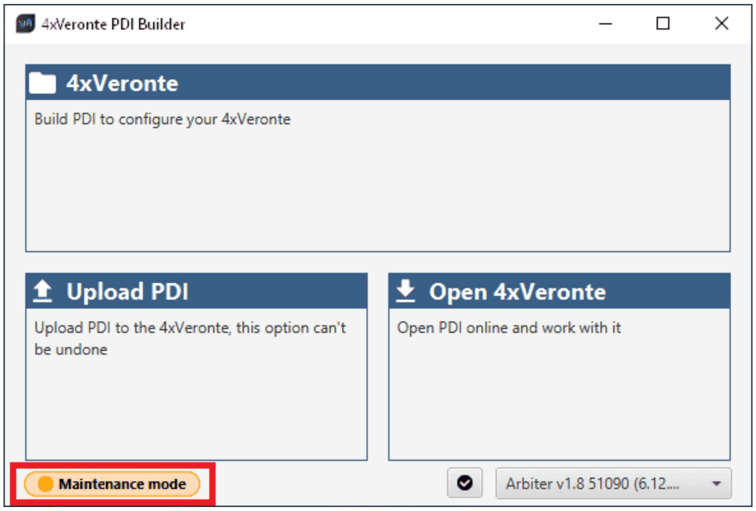
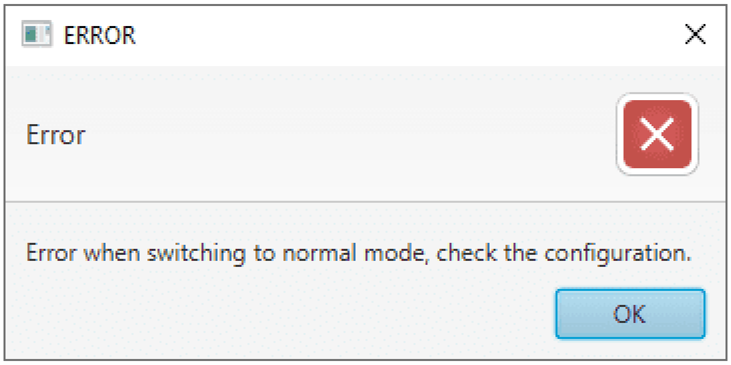

4xVeronte PDI Builder
4xVeronte
Build PDI to configure your 4xVeronte
Upload PDI
Upload PDI to the 4xVeronte, this option can't be undone
Open 4xVeronte
Open PDI online and work with it
Maintenance mode
Arbiter v1.8 51090 (6.12....

Fig. 2: **Exit maintenance mode**

4.2 Maintenance mode (loaded with errors)

The following error message may appear when trying to save a change or import a configuration.

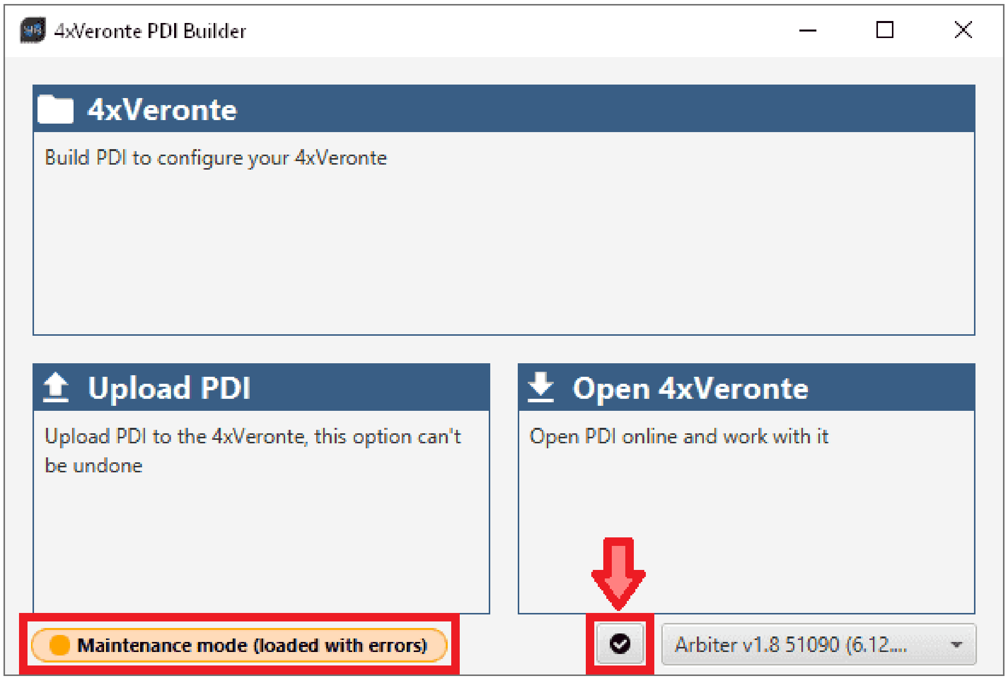


Error

Error when switching to normal mode, check the configuration.

Fig. 3: **Error message**

Therefore, the **arbiter** will be in ‘Maintenance mode (loaded with errors)’:



4xVeronte PDI Builder

4xVeronte

Build PDI to configure your 4xVeronte

Upload PDI

Upload PDI to the 4xVeronte, this option can't be undone

Open 4xVeronte

Open PDI online and work with it

Maintenance mode (loaded with errors)
Arbiter v1.8 51090 (6.12....

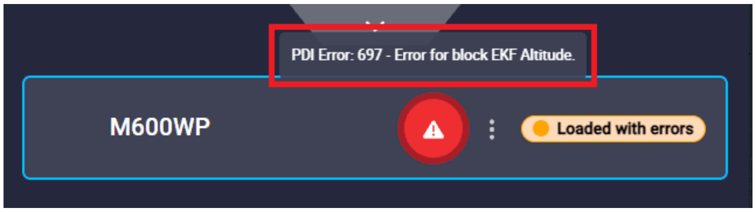
To check what the source of the problem is, the user can simply click on the **PDI Error button** ✓, which will show what the PDI Error is in the **PDI Errors panel**.

In this panel, users can found:

- **PDI ID**: ID of the PDI Error.
- **PDI Error Description**: Description of this PDI Error. A list of all PDI Errors can also be accessed in the List of PDI errors section of the **4x Software Manual**.
- **Config ID**: ID of the configurable (.xml file) containing the data in which the PDI Error has been caused.
- **Config Description**: Description of the configurable (.xml file) containing the data in which the PDI Error has been caused.

Clicking the **Export** button will export a `.csv` file with the same information shown in this PDI Errors panel.

This is useful while the configuration is in progress, however, if the user encounters this situation during the operation, it is also possible to look up the cause of the PDI Error directly on the **Platform panel** of **Veronte Ops**. For more information about this panel, see Platform panel section of the **Veronte Ops** user manual.



PDI Error: 697 - Error for block EKF Altitude.

Fig. 4: **Error message**

Then, it will be possible to modify the **Arbiter** configuration to fix the error.

In addition, the list of PDI Errors can be read in the List of PDI errors section of the **4x Software Manual**.

4.3 Migrate configuration

Warning: When performing automatic migration from a previous version to the current version of the software, errors may occur.

It is then the responsibility of the user to check the subsequent result.

CHAPTER

FIVE

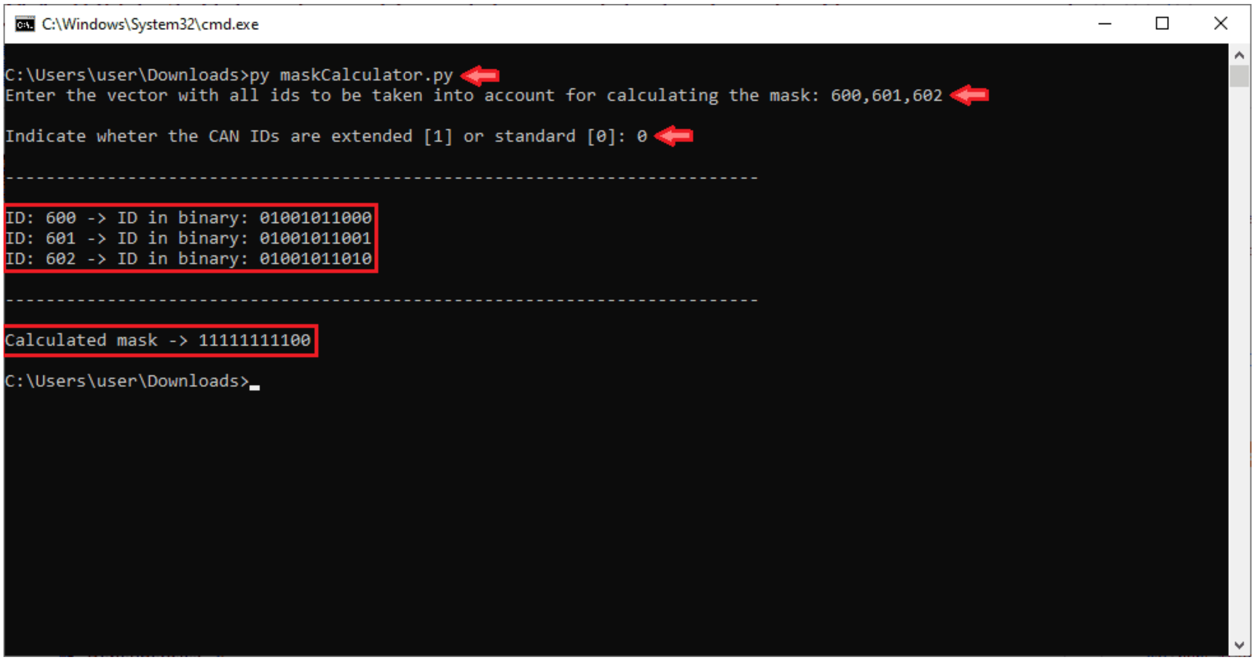
FAQ

5.1 How to calculate a mask

This section attaches a python program that allows users to easily calculate their mask in standard or extended frame format by simply entering the CAN Ids as a **vector**. In addition, this program also converts each Id entered into binary.

`maskCalculator.py`

An example of the execution of this program is shown below:



C:\Windows\System32\cmd.exe

```
C:\Users\user\Downloads>py maskCalculator.py
Enter the vector with all ids to be taken into account for calculating the mask: 600,601,602

Indicate wheter the CAN IDs are extended [1] or standard [0]: 0

--------------------------------------------------------------------------

ID: 600 -> ID in binary: 01001011000
ID: 601 -> ID in binary: 01001011001
ID: 602 -> ID in binary: 01001011010

--------------------------------------------------------------------------

Calculated mask -> 11111111100

C:\Users\user\Downloads>_
```

Fig. 1: **Example of maskCalculator program**