MEX PDI Builder

Release 6.14/1.0

Embention Sistemas Inteligentes, S.A.

2026-02-11

Contents

Scope of Changes 5

Quick Start 6

- System Requirements 6
- Download and Installation 6

Configuration 8

- MEX 14
  - MEX Base 14
  - Status 15
- Sensors 16
  - RPM 16
  - Magnetometer 17
- Input/Output 19
  - GPIO 19
  - PWM 20
  - I/O Setup 23
    - Tunnel 27
    - Serial Custom Messages 28
- 32
    - JETI box 32
  - CAN I/O 35
    - Configuration 36
    - CAN Telemetry 39
  - Digital Input 40
  - Serial 45
  - CAN Setup 46
  - Custom Msg Trigger 48
- Communications 50
- Stick 52
  - PPM 52
  - Exponential 54
  - Trim 55
  - Output 56
- Devices 57

Jetibox ... 57
Scorpion tribunus ... 58
Arbitration ... 59
Arbitration ... 59
Config ... 60
CAN Setup ... 62
Integration examples ... 64
CAN communication ... 64
CAN messages transmission ... 64
CAN messages reception ... 66
CAN Isolator ... 68
Filtered CAN subnet ... 69
CAN tunnel ... 72
Commanding/Reading PWMs ... 75
1x PDI Builder side ... 75
MEX PDI Builder side ... 77
GPIO Command ... 80
1x PDI Builder side ... 75
MEX PDI Builder side ... 77
MEX as external magnetometer for Autopilot 1x ... 82
Serial ... 82
MEX PDI Builder side ... 77
1x PDI Builder side ... 75
CAN ... 86
MEX PDI Builder side ... 77
1x PDI Builder side ... 75
Reading/Sending RPMs ... 91
MEX PDI Builder side ... 77
1x PDI Builder side ... 75
Serial communication ... 97
MEX PDI Builder side ... 77
1x PDI Builder side ... 75
External devices ... 101
Jetibox ... 102
Scorpion tribunus ... 107

Veronte products 108

Connection with Autopilot 1x via CAN 108

Connection with Autopilot 4x via CAN 108

CAN Reception IDs 108

CAN I/O Interconnections 109

Troubleshooting 111

Maintenance mode (loaded with errors) 111

Migrate configuration 112

FAQ 113

How to calculate a mask 113

Scope of Changes

- Version 1.0
    - Added
        - First version issued

Quick Start

MEX PDI Builder is the main configuration tool to adapt a **MEX** to a specific vehicle, including user-defined commnication protocols. It includes:

- Telemetry: real-time onboard UAV metrics, such as sensors, actuators and control states.
- Communications: through General Purpose Inputs and Outputs, PWMs and CAN channels.
- Stick control signal management: compatible with **Stick Expander**, Futaba, Jeti, FrSky and TBS. It includes custom configuration for other sticks.
- Arbitration: **MEX** is able to send PWM signals using arbitration in the same way **Veronte Autopilot 4x** does.

System Requirements

Before executing this software, users should check the following sections with the minimum and recommended PC hardware requirements.

Minimum requirements

- CPU: Intel Core i5-8365UE
- RAM: 8 GB DDR4
- STO: 256 GB SSD

Recommended requirements

- CPU: 12th Gen Intel(R) Core(TM) i7-12700H 14 cores up to 4,70 GHz
- RAM: 32 GB
- STO: 1 TB SSD M.2 NVMe PCIe

Download and Installation

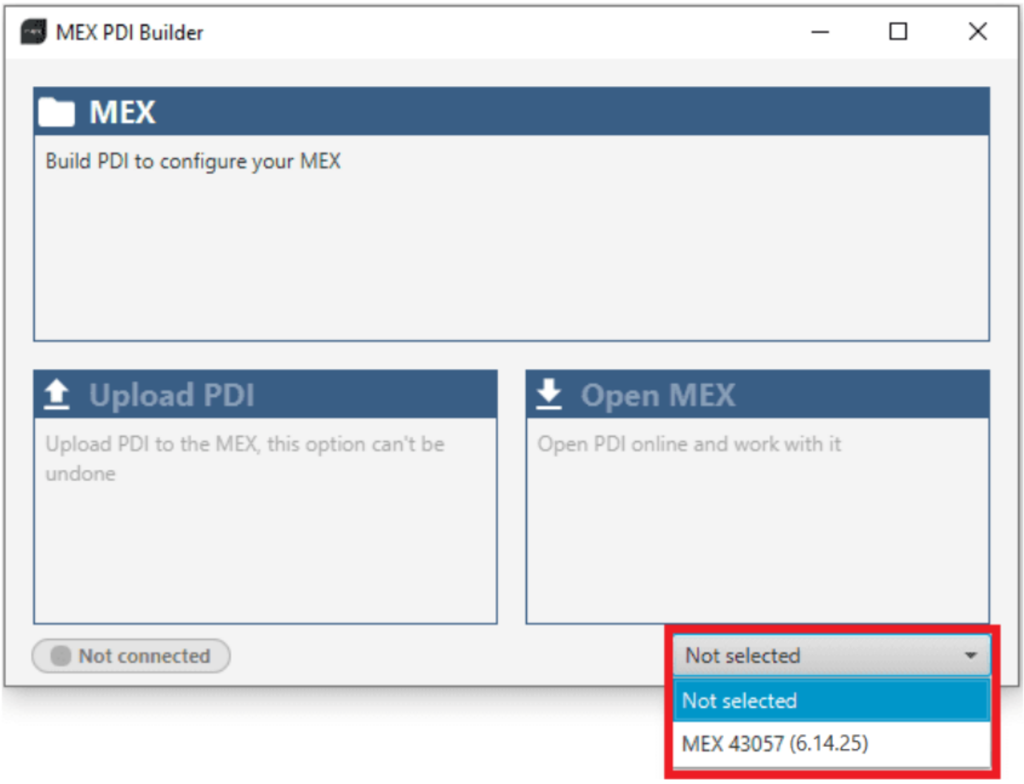
MEX PDI Builder software is available in the **Veronte Toolbox** platform. From there, users can download and install the application. For more information, please refer to the Veronte Toolbox user manual.

A **personal account** is required to access **Veronte Toolbox**; create a Ticket in the user's **Joint Collaboration Framework** and the support team will create it for you.

Configuration

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

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



MEX PDI Builder

MEX

Build PDI to configure your MEX

Upload PDI

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

Open MEX

Open PDI online and work with it

Not connected

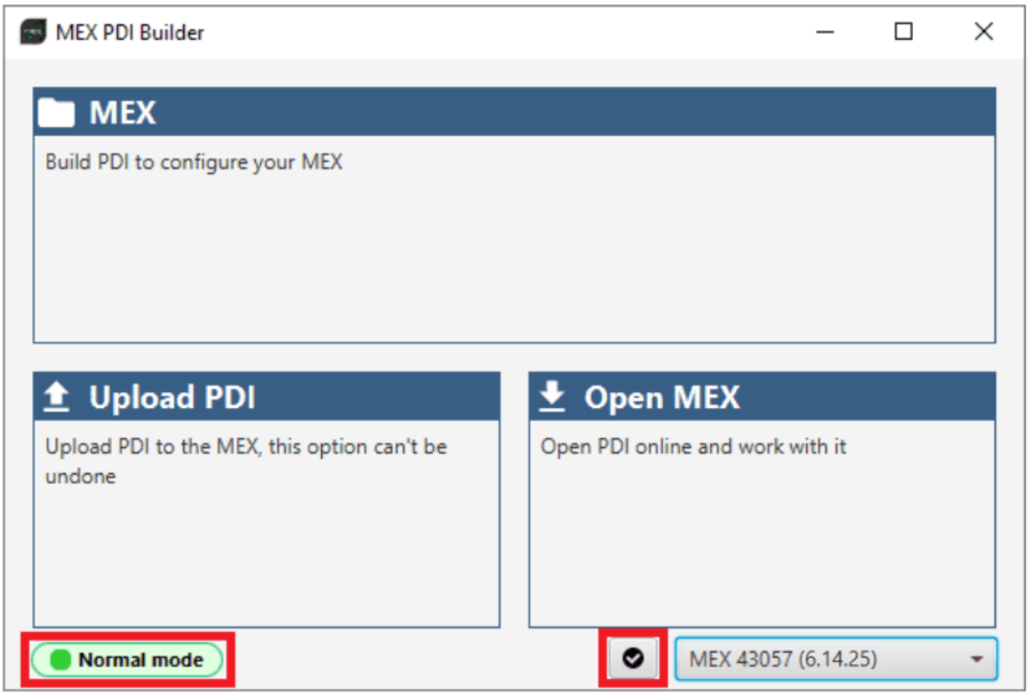
Not selected

Not selected

MEX 43057 (6.14.25)

MEX ID

If correctly connected, **MEX PDI Builder** will display the **mode** in which the connected unit is. In addition, a **PDI error button** ✔ will appear:



Build PDI to configure your MEX

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

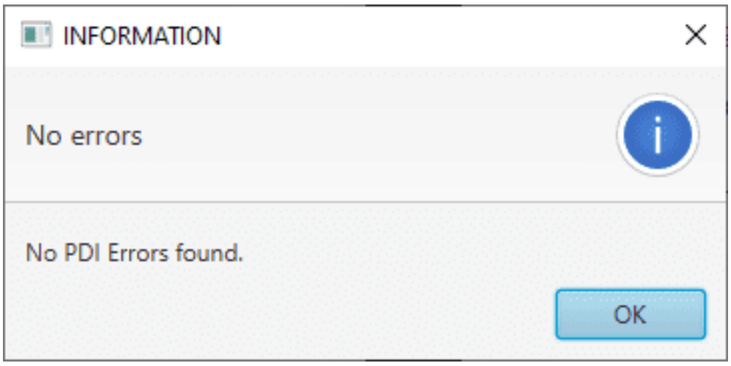
Open PDI online and work with it

MEX PDI Builder

- **MEX mode**: The MEX 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 went wrong in 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 appears:

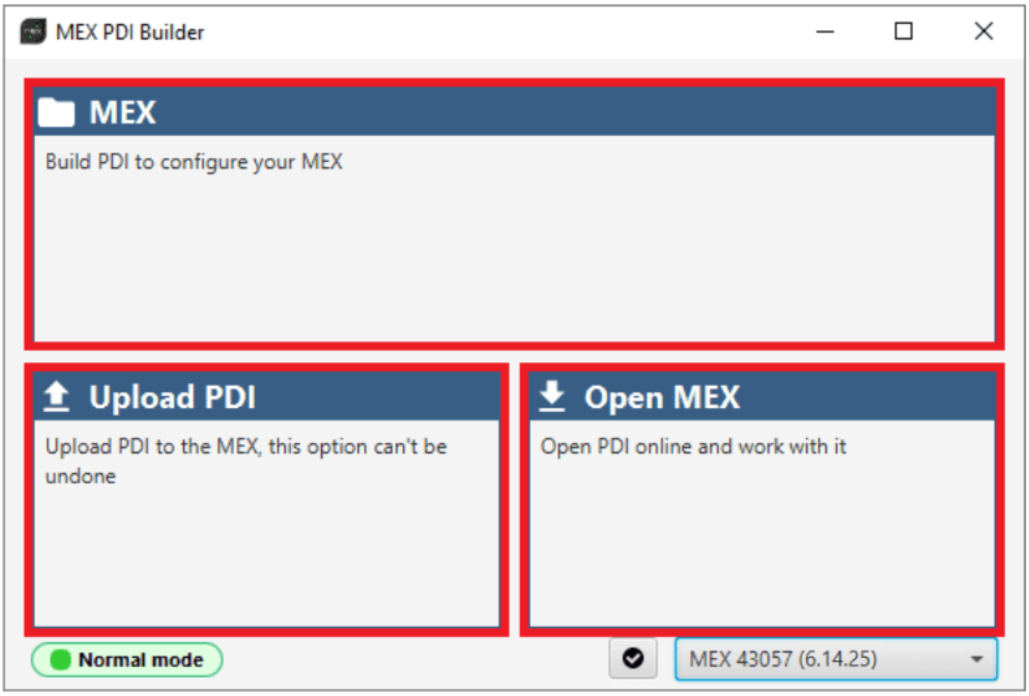


No errors

No PDI Errors found.

MEX PDI Builder - PDI Errors message

The user can access now to 3 configuration options:


MEX PDI Builder
MEX
Build PDI to configure your MEX
Upload PDI
Upload PDI to the MEX, this option can't be undone
Open MEX
Open PDI online and work with it
Normal mode
MEX 43057 (6.14.25)

MEX PDI Builder options

- **MEX**: It allows the user to work with **offline** configurations. A previously exported MEX PDI configuration can be opened and modified or it is also possible to build a new one from the default configuration.
- **Upload PDI**: A previously exported **MEX PDI configuration** can be imported to the linked **MEX**.

  > ⚠ **Warning**
  >
  > When a configuration is loaded into MEX 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 MEX**: By clicking on this option, **MEX PDI Builder** configuration menu opens with the configuration (the PDI files) loaded in the MEX. Then, the user can modify it **online**.

ⓘ **Note**

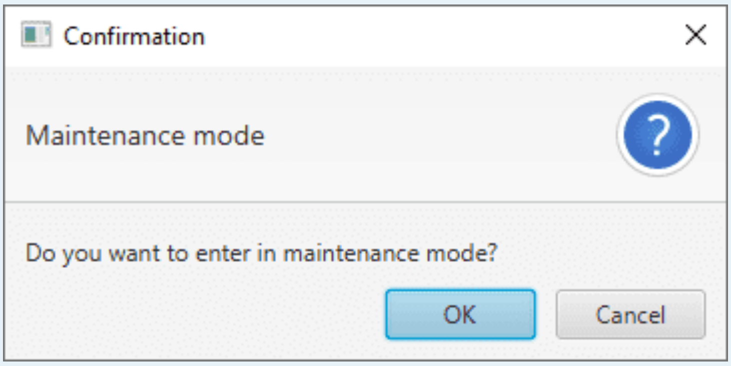
PDI files are the files that contain the configuration of the **MEX**. These files are located in a single folder, setup, which contains several .xml files, where all parameters and the control system are stored.

MEX ID

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

ⓘ **Note**

MEX unit must enter in **Maintenance mode** for the user to start editing, so it is necessary to accept the confirmation panel below.

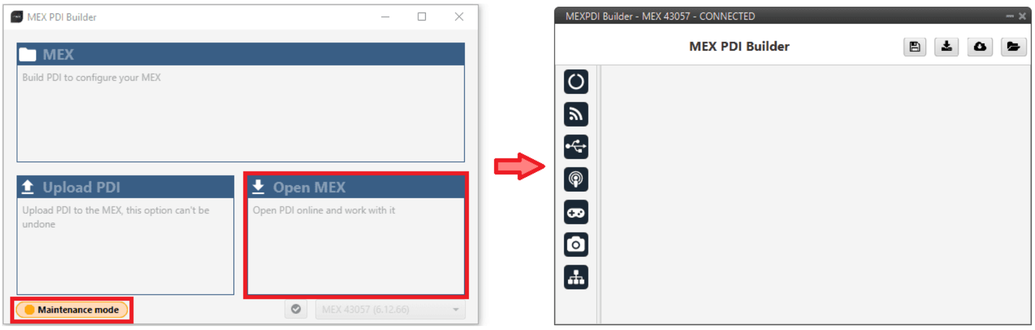


Confirmation

Maintenance mode

Do you want to enter in maintenance mode?

Enter maintenance mode - Confirmation panel



MEX PDI Builder

MEX
Build PDI to configure your MEX

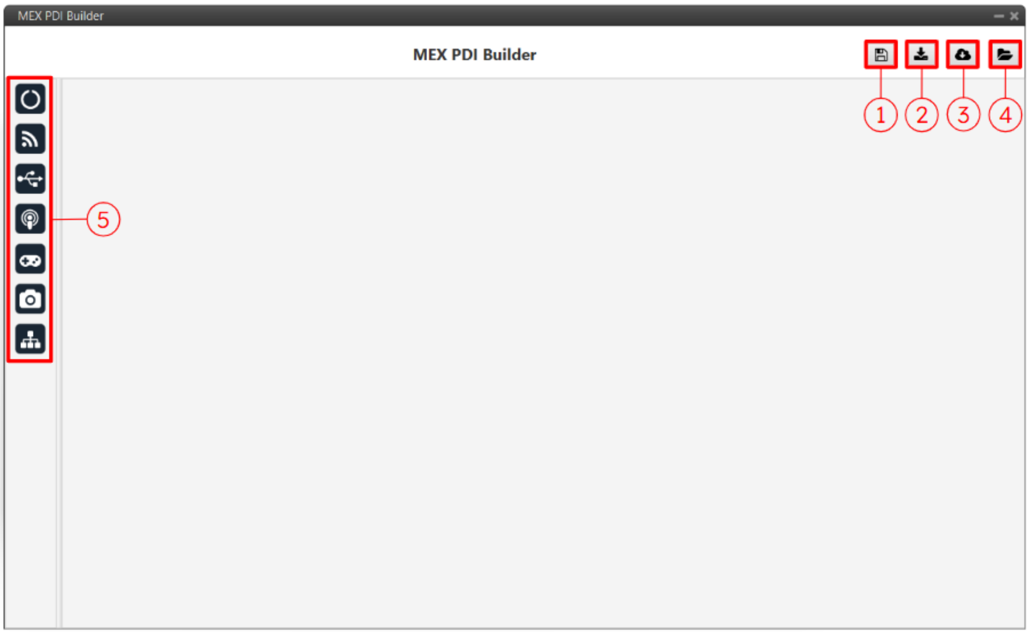
Upload PDI
Upload PDI to the MEX, this option can't be undone

Open MEX
Open PDI online and work with it

MEXPDI Builder - MEX 43057 - CONNECTED

Open MEX

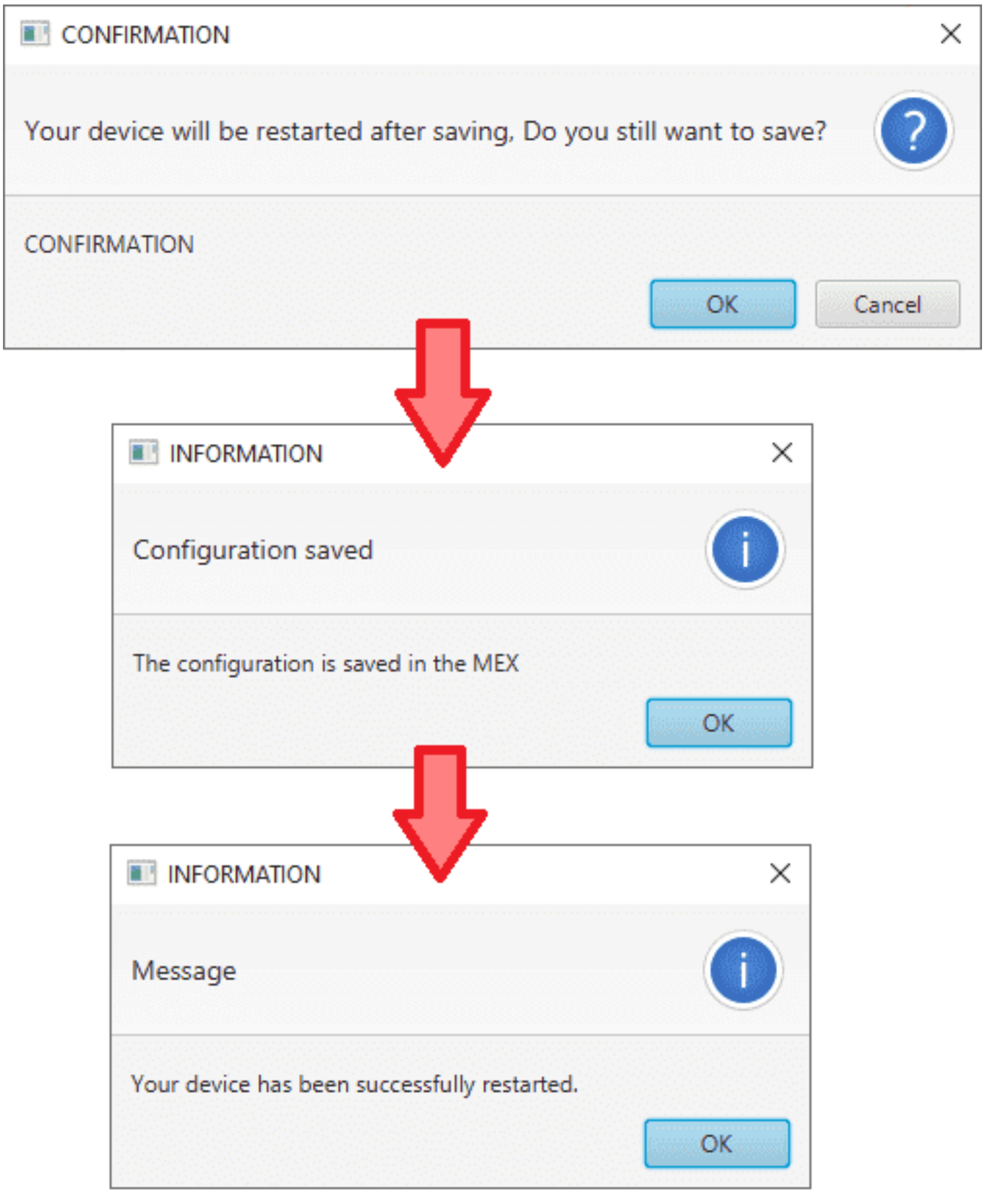
Once in Maintenance mode, the user can access the Initial menu. The different 'buttons' that can be seen in this menu of the **MEX PDI Builder** are explained below.



Initial menu

1. **Save and close**: After changes are done, press on the save button to apply the changes. While saving, a percentage of saving process is displayed.

   In order to save the configuration in the **MEX** unit it is necessary to **RESET**, therefore the **MEX 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 MEX

Message

Your device has been successfully restarted.

Save and close - Confirmation panels

> ⓘ **Danger**
>
> As MEX is **reset**, it is **not advisable to save changes during flight tests**.

> ⓘ **Note**
>
> This button will only appear if a **MEX** 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.

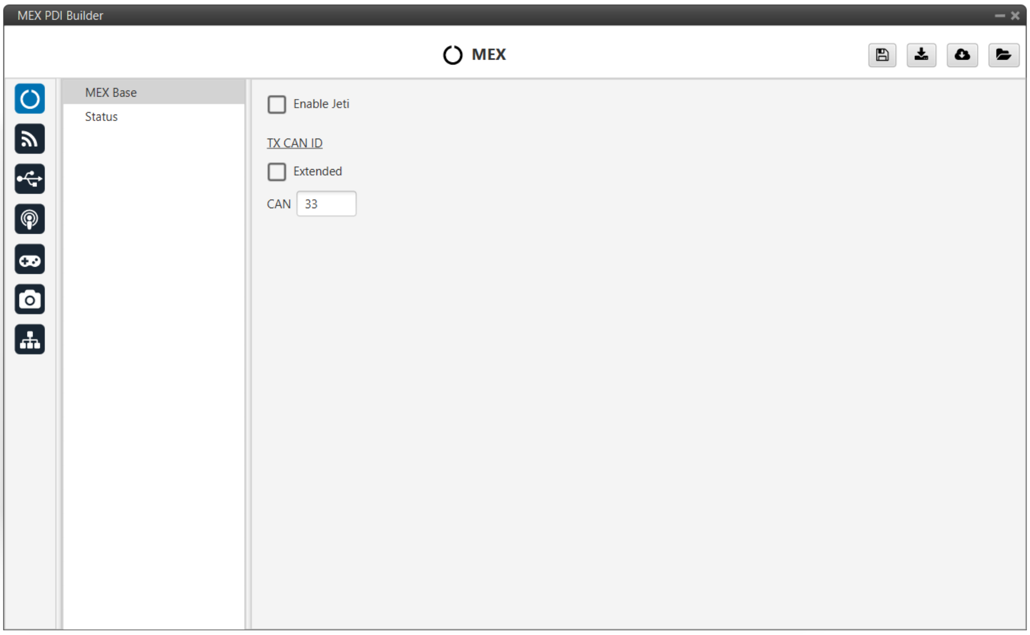
3. **Import PDI from repo**: The user can import a configuration file from the GitHub repository and modify it. After that, if the **Save and close** button (1) is pressed, this configuration will be uploaded to the connected **MEX**.
4. **Import PDI from local storage**: The user can import a configuration 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 MEX.
5. These are the different functions of MEX. They are explained in the following sections.
    - MEX
    - Sensors
    - Input/Output
    - Communications
    - Stick
    - Devices
    - Arbitration

MEX

MEX Base

MEX is able to send information about its version and Jeti telemetry from devices connected to **MEX**, this is the "status message". It is possible to define the CAN Id used for those messages.

Enable the **Extended** checkbox to use the extended CAN protocol (with a 29-bit identifier), disable it to use the standard protocol (11-bit identifier).



☐ Enable Jeti

TX CAN ID

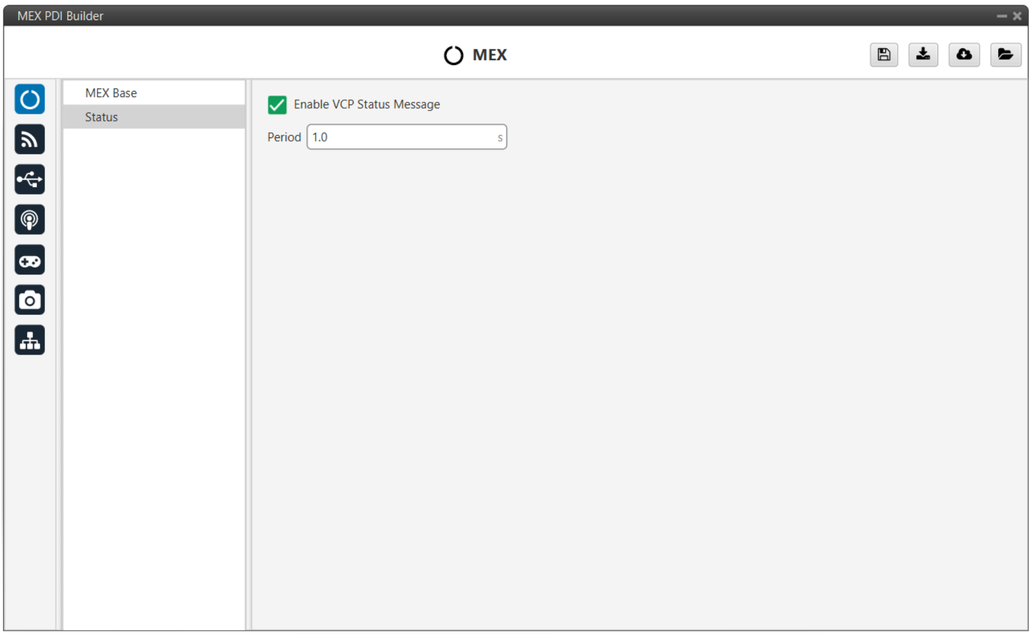
☐ Extended

CAN 33

Mex base panel

Status

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



☑ Enable VCP Status Message

Period 1.0 s

Status panel

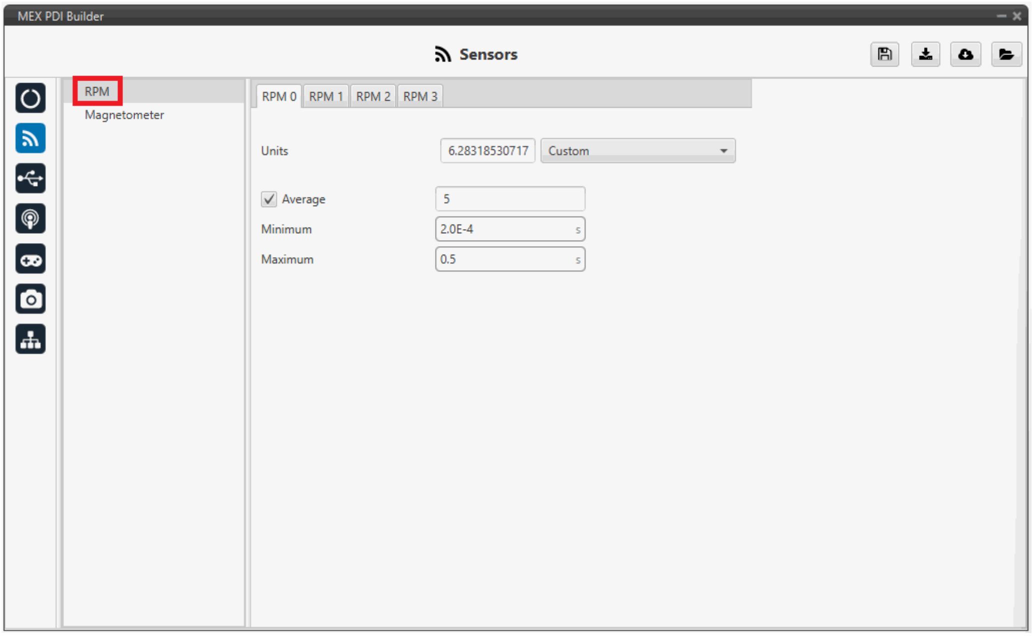
- **Period**: Enter a desired period to send repeatedly the status message.

> ⓘ **Note**
>
> VCP is the Veronte Communication Protocol. To know more, read the VCP user manual.

Sensors

RPM

MEX can measure RPMs by reading froms up to four input sources:



Units: 6.28318530717, Custom

☑ Average: 5

Minimum: 2.0E-4 s

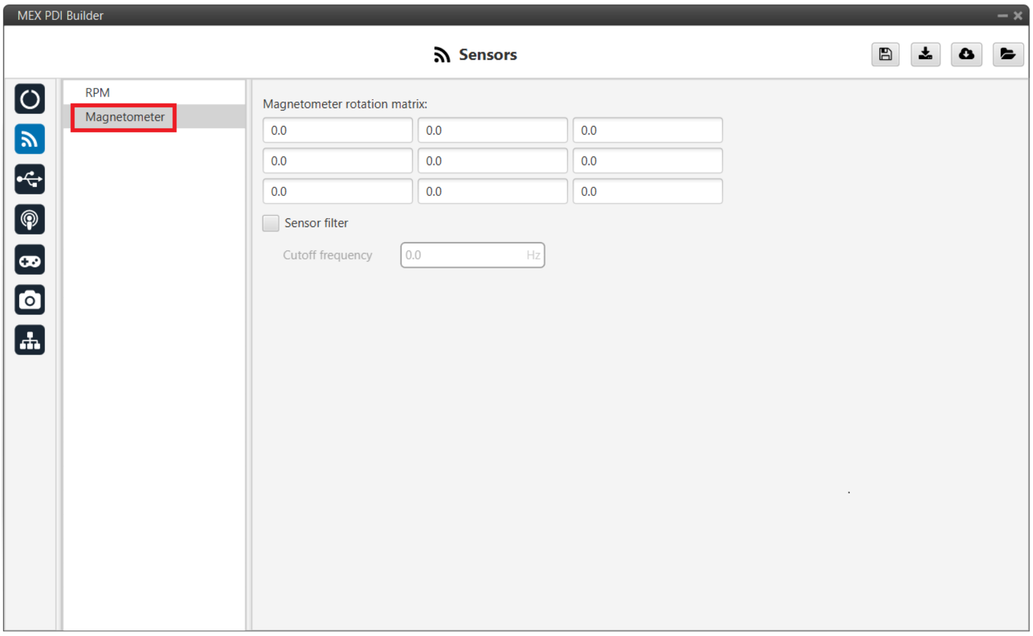
Maximum: 0.5 s

RPM panel

- **Units**: Sensor conversion factor. It can be Pulse per cycle, Radians per pulse or Custom.
- **Average**: Filter to prevent voltage 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**: 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.

An example about sending RPMs can be found in Reading/Sending RPMs - Integration examples section of the present manual.

Magnetometer



Magnetometer rotation matrix:

0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

□ Sensor filter

Cutoff frequency 0.0 Hz

Magnetometer panel

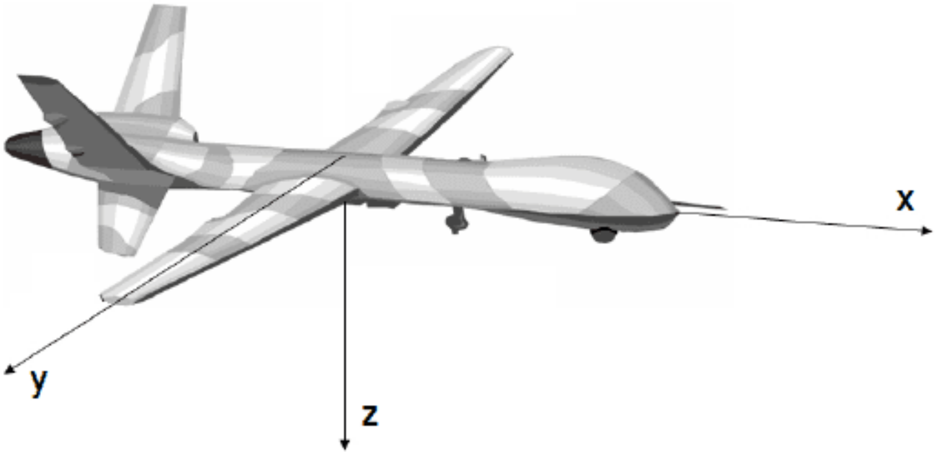
Magnetometer matrix: This rotation matrix represents the rotation of **MEX** coordinates with respect to the aircraft body. The behavior of the matrix varies depending on how **MEX** is used.

- **If MEX is used independently (without Veronte Autopilot 1x):** The rotation matrix should convert **MEX** coordinates to the **Aircraft Body** coordinates. That is, the conversion is directly from **MEX** to the aircraft coordinates.
- **If MEX is used together with Veronte Autopilot 1x:** The rotation matrix should convert **MEX** coordinates to **Veronte Autopilot 1x** coordinates, and then **Veronte Autopilot 1x** will convert the data to the **Aircraft Body** coordinates; for more information on the rotation matrix of **Autopilot 1x**, please refer to the Magnetometer - Sensors section of the **1x PDI Builder** user manual. Therefore, the coordinate transformation sequence is: **MEX** → **1xVeronte** → **Aircraft Body**.

ⓘ Error

The rotation matrix **cannot** be a **zero matrix** and must respect **the orthogonality of the axes**. Not complying with this requirement means an invalid rotation and, consequently, the calibration of this magnetometer will not be possible.

The axis of **MEX** are physically drawn on the device. The **coordinates axis of the aircraft** are like other Veronte products, which are **defined by the Standard Aeronautical Convention**.



Aircraft coordinates (Standard Aeronautical Convention)

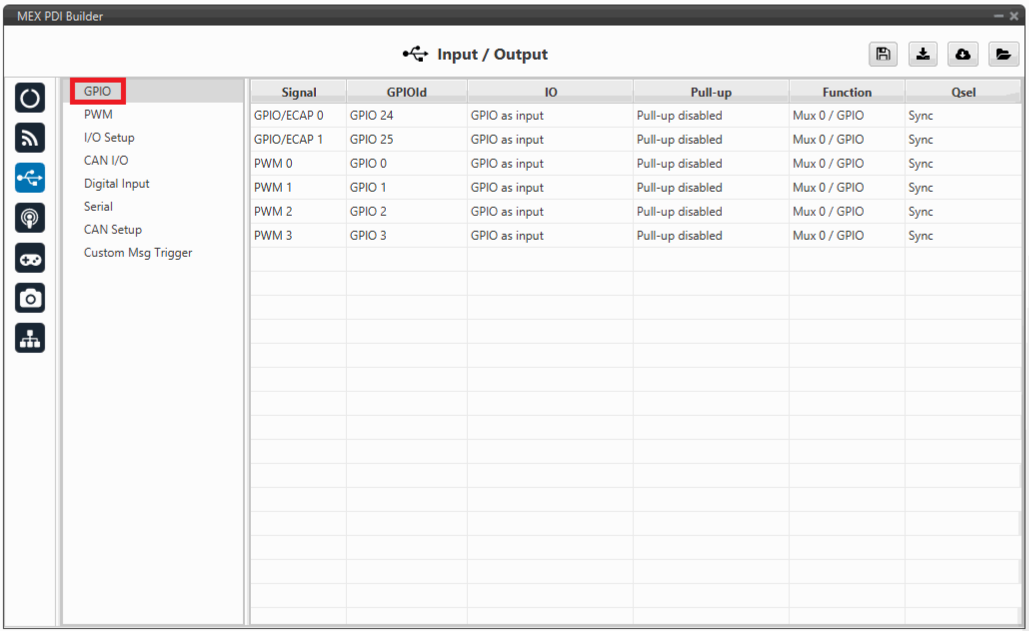
- **Sensor filter**: The magnetometer measurements may vary with high frequencies and be difficult to read.

  To solve this problem, **MEX** has this second order low-pass filter to mitigate high frequency variations. It is a **software filter**. This filter can be enabled clicking on this checkbox.
- **Cutoff frequency**: Sensor filter cutoff frequency is manually configured, allowing the user to enter any desired value in Hz. Frequency variations higher than this value will be filtered out (if the filter is enabled).

Input/Output

GPIO

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



Signal	GPIOId	IO	Pull-up	Function	Qsel
GPIO/ECAP 0	GPIO 24	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
GPIO/ECAP 1	GPIO 25	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 0	GPIO 0	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 1	GPIO 1	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 2	GPIO 2	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 3	GPIO 3	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync

GPIO panel

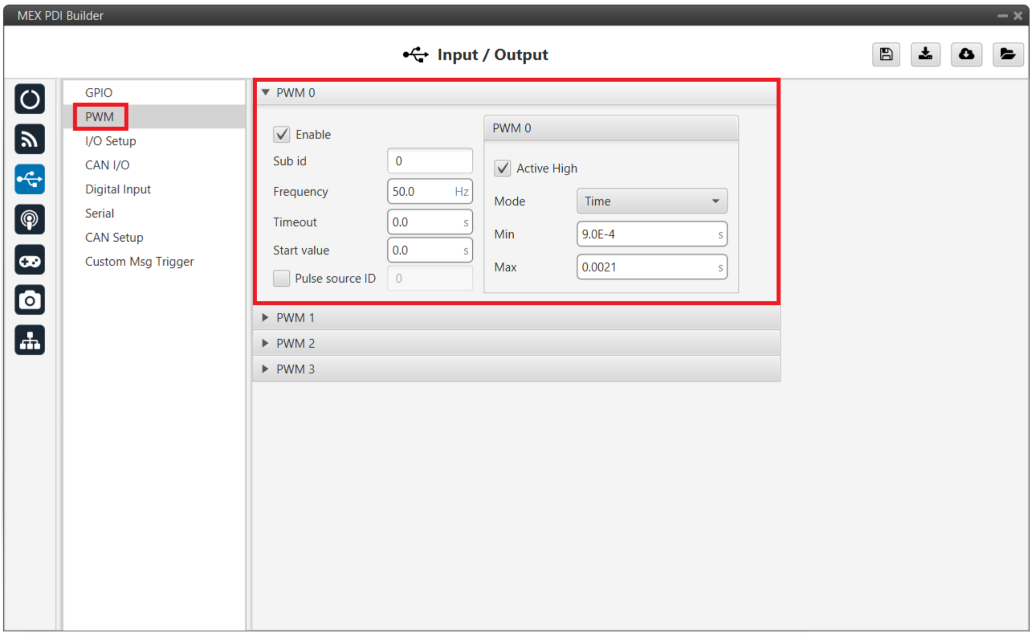
- **Signal**: Pin ID as described in the Pinout - Hardware installation section of the **MEX Hardware Manual**.
- **GPIOId**: GPIO ID of the microcontroller.
- **IO**: Define GPIO as an input or ouput.
- **Pull-up**: Enable or disable the pull-up resistance.
- **Function**: Mux 0/GPIO: GPIO, Mux 1: PWM, Mux 2 or Mux 3. These are the different functionalities that the GPIO can have, depending on the multiplexer.

> ⓘ **Note**
>
> When users set **Function** to "Mux 1", it indicates that the corresponding pin is disabled as GPIO and enabled as PWM. Consequently, the **Enable** checkbox in the PWM menu for that pin should be activated automatically.
>
> > ⚠ **Warning**
> >
> > Check that the corresponding **Enable** checkbox in the PWM menu is changed.

- **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.

PWM

In this panel, each PWM can be configured:



PWM 0

☑ Enable

Sub id: 0

Frequency: 50.0 Hz

Timeout: 0.0 s

Start value: 0.0 s

□ Pulse source ID: 0

PWM 0

☑ Active High

Mode: Time

Min: 9.0E-4 s

Max: 0.0021 s

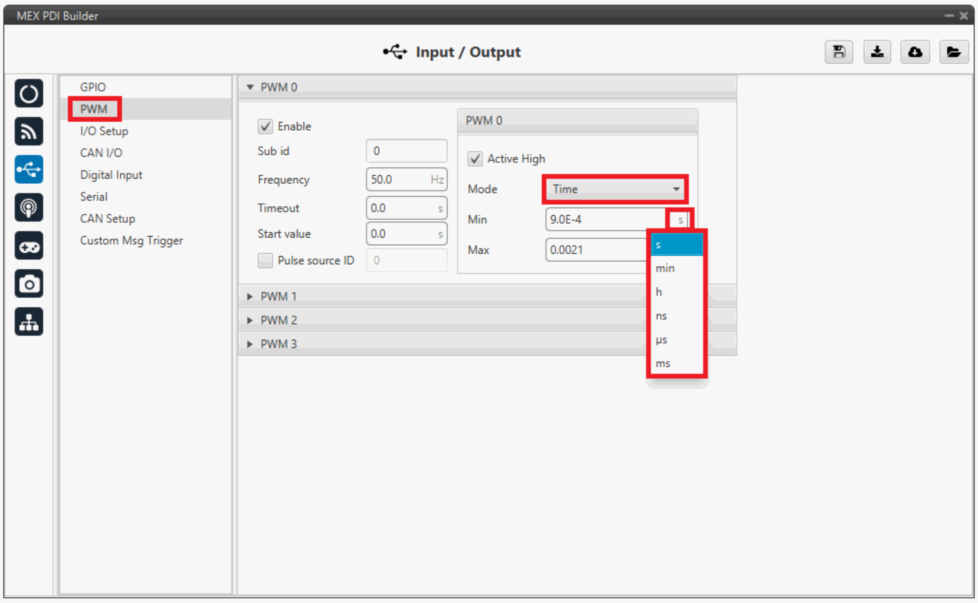
PWM panel

> ⓘ **Note**
>
> PWMs in MEX work in normalized mode, when the input value is 0 the output value will be the minimum configured, and when the input value is 4095 (12 bits all with ones), the output will be the maximum configured. This approach allows usage of the maximum resolution for the commanded value.

- **Enable**: Defines if the PWM is enabled or not.

  > ⓘ **Note**
  >
  > This checkbox disables the pin as GPIO and enables it as PWM. Hence, in the GPIO menu the "Function" parameter shall change to "**Mux 1**".

- **Sub id**: Identifies the sub-id of the PWM and, according to this value, determines the command to be used using one type of CAN message or another.
- **Frequency**: PWM output frequency. This determines the period of the pulses sent by the MEX
- **Timeout**: If a PWM message is not received in less than this time, the PWM will send as output the start value.
- **Start value**: Value used before any PWM message arrives and on timeout.
- **Pulse source ID**: PWM input ID [0,3], defined in the Digital Input panel.
- **Active High**: Polarity high or low (high if enabled).
- **Mode**: The avilable options are **Time** and **Duty cycle**.
- **Time**: The values indicated in **Min** and **Max** parameters are expressed in time units.



☑ Enable

Sub id 0

Frequency 50.0 Hz

Timeout 0.0 s

Start value 0.0 s

□ Pulse source ID 0

PWM 0

☑ Active High

Mode Time

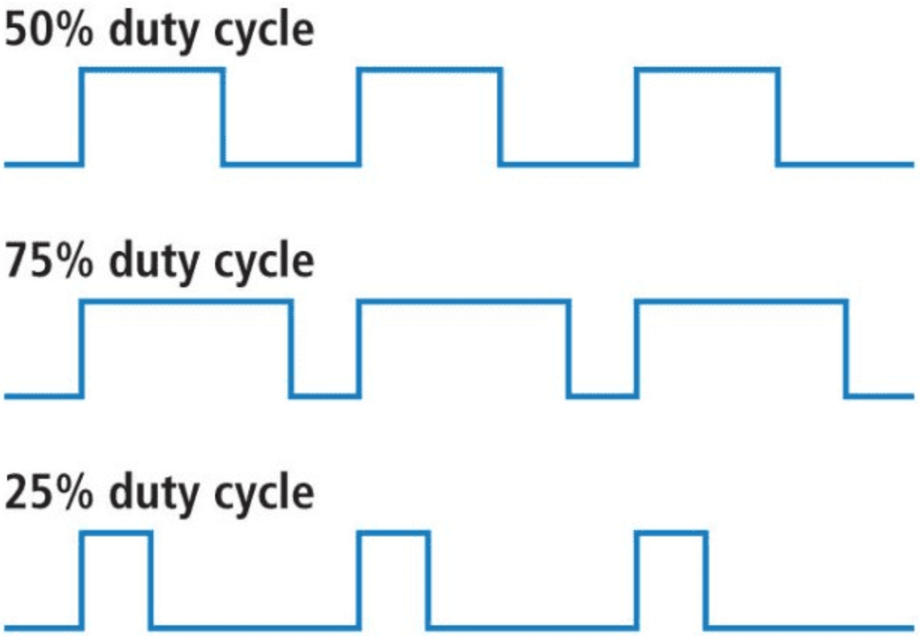
Min 9.0E-4 s

Max 0.0021

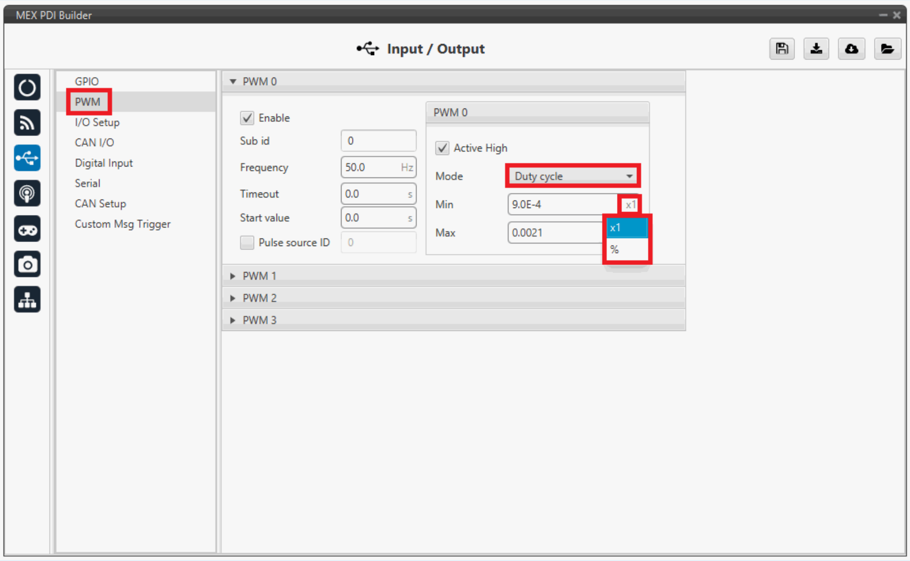
PWM panel - Time units

- **Duty cycle**: This option is a a different way of indicating the pulse width. Now the value indicated in **Min** and **Max** parameters is a percentage which corresponds to the relation 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.



Duty cycle

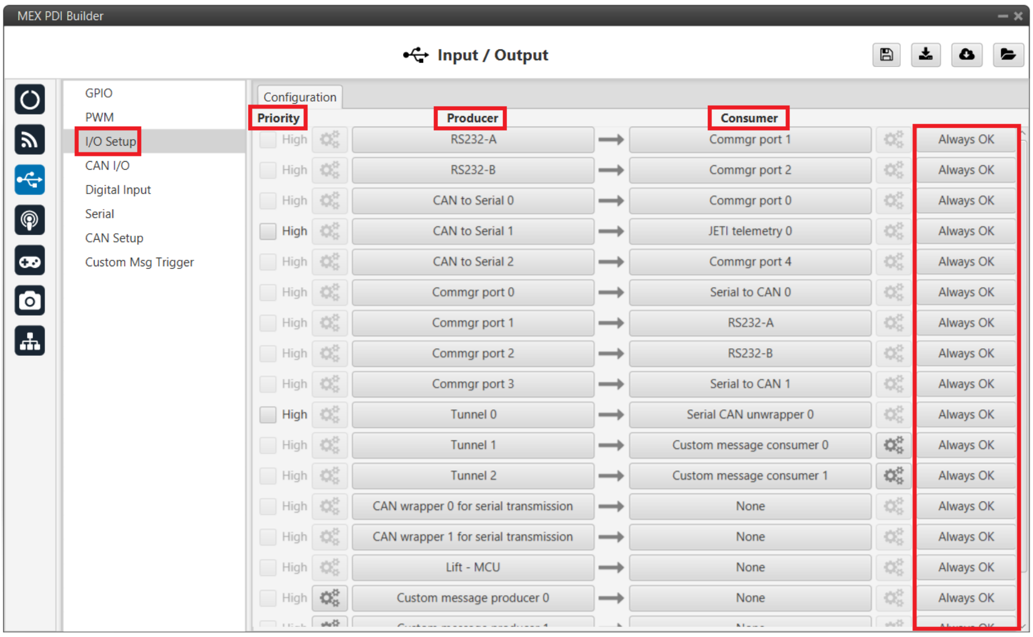
> **Note**
>
> Duty cycle percentages can be expressed in percent and per unit.
>
> 
>
>
> PWM 0
>
> ☑ Enable
>
> Sub id: 0
>
> Frequency: 50.0 Hz
>
> Timeout: 0.0 s
>
> Start value: 0.0 s
>
> ☐ Pulse source ID: 0
>
> PWM 0
>
> ☑ Active High
>
> Mode: Duty cycle
>
> Min: 9.0E-4 x1
>
> Max: 0.0021
>
> **PWM panel - Duty cycle units**

- **Min**: This parameter is the pulse width value that will make the servo/ actuator go to its **lowest position**. It will be the output when the PWM message specifies **0**.
- **Max**: This parameter is the pulse width value that will make the servo/ actuator go to its **highest position**. It will be the output when the PWM message specifies **4095**.

An example about reading PWM signals can be found in Commanding/Reading PWMs - Integration examples section of the present manual.

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-A	→	Commgr port 1	Always OK
☐ High	RS232-B	→	Commgr port 2	Always OK
☐ High	CAN to Serial 0	→	Commgr port 0	Always OK
☐ High	CAN to Serial 1	→	JETI telemetry 0	Always OK
☐ High	CAN to Serial 2	→	Commgr port 4	Always OK
☐ High	Commgr port 0	→	Serial to CAN 0	Always OK
☐ High	Commgr port 1	→	RS232-A	Always OK
☐ High	Commgr port 2	→	RS232-B	Always OK
☐ High	Commgr port 3	→	Serial to CAN 1	Always OK
☐ High	Tunnel 0	→	Serial CAN unwrapper 0	Always OK
☐ High	Tunnel 1	→	Custom message consumer 0	Always OK
☐ High	Tunnel 2	→	Custom message consumer 1	Always OK
☐ High	CAN wrapper 0 for serial transmission	→	None	Always OK
☐ High	CAN wrapper 1 for serial transmission	→	None	Always OK
☐ High	Lift - MCU	→	None	Always OK
☐ High	Custom message producer 0	→	None	Always OK
☐ High	Custom message producer 1	→	None	Always OK

I/O Setup panel

- **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 to a bit. Thus, the connection is 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.

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.
4. (optional) Configure the **Bit** parameter. By pressing on the bit button, the user can select the bit to assign the connection to.

The following I/O ports are available:

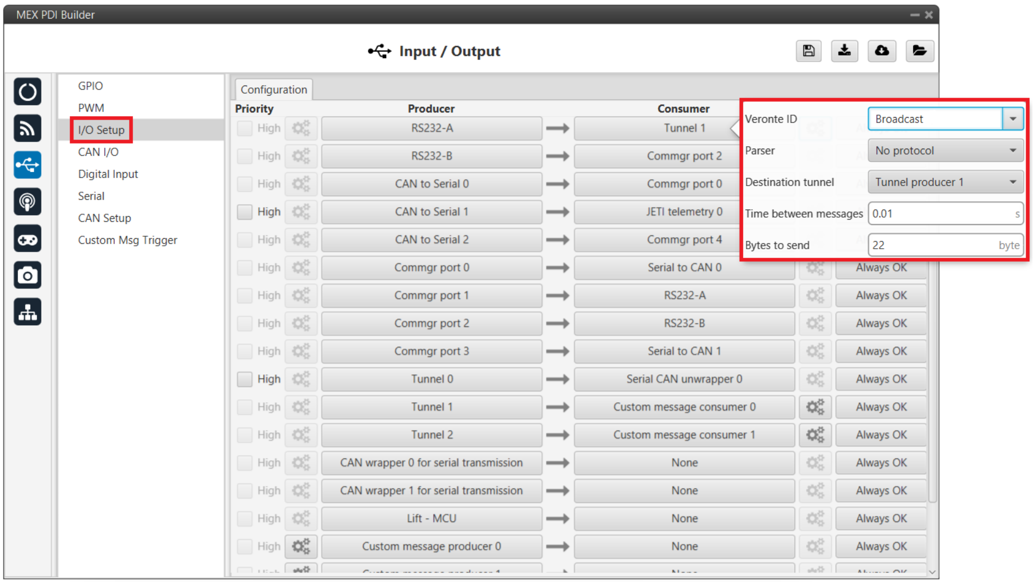
Field	Description
RS232-A	Serial Port 232 A
RS232-B	Serial Port 232 B
RS485-C	Serial Port 485
Commgr port	COM Manager ports send and receive VCP messages. This is the protocol used by Veronte products to communicate. For more information, read the VCP user manual
Tunnel	Creates a bidirectional bridge between two devices, see Tunnel
Custom message producer/ consumer	This allows user to send/ receive a serial custom message, see Serial Custom Messages
CAN to Serial / Serial to CAN	**Serial to CAN** sends serial streams over a CAN Bus / **CAN to Serial**

Field	Description
	undoes the transformation 'Serial to CAN'
CAN wrapper for serial transmission / Serial CAN unwrapper	**CAN wrapper for serial** transmission sends CAN streams over a serial Bus / **Serial CAN unwrapper** 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**
Tribunus ESC	Reads telemetry data from the Tribunus ESCs by connecting it to one of the serial ports
Lift - MCU	Created for communication with a Lift MCU
JETI box	Simulates a Jetibox to read telemetry from legacy Jeti devices, see JETI box
JETI telemetry	Reads telemetry from Jeti devices

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

Tunnel

A tunnel is a bidirectional bridge between units that communicate each other, sharing information about an external device connected to the serial or digital port. The following image shows an example of tunnel configuration:



Priority	Producer		Consumer	
☐ High	RS232-A	→	Tunnel 1	
☐ High	RS232-B	→	Commgr port 2	
☐ High	CAN to Serial 0	→	Commgr port 0	
☐ High	CAN to Serial 1	→	JETI telemetry 0	
☐ High	CAN to Serial 2	→	Commgr port 4	
☐ High	Commgr port 0	→	Serial to CAN 0	Always OK
☐ High	Commgr port 1	→	RS232-A	Always OK
☐ High	Commgr port 2	→	RS232-B	Always OK
☐ High	Commgr port 3	→	Serial to CAN 1	Always OK
☐ High	Tunnel 0	→	Serial CAN unwrapper 0	Always OK
☐ High	Tunnel 1	→	Custom message consumer 0	Always OK
☐ High	Tunnel 2	→	Custom message consumer 1	Always OK
☐ High	CAN wrapper 0 for serial transmission	→	None	Always OK
☐ High	CAN wrapper 1 for serial transmission	→	None	Always OK
☐ High	Lift - MCU	→	None	Always OK
☐ High	Custom message producer 0	→	None	Always OK
☐ High	Custom message producer 1	→	None	Always OK

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

Tunnel configuration

In the previous image there is a device connected to the **RS232-A (Producer)** and there is a **Tunnel (Consumer)** which sends that information to all units on network (Broadcast). On the other hand, another Veronte device has to be configured to receive the signal sent by another device. In this case, the **Producer** will be **Tunnel**, while **Consumer** will be the **port or destination tunnel where the device is connected**.

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

- **Veronte ID**: Enter the address that will receive the information. The following options are the most common:
    - **App 2**: Veronte applications address.
    - **Broadcast**: All units on the network. Select this option for a generic configuration.

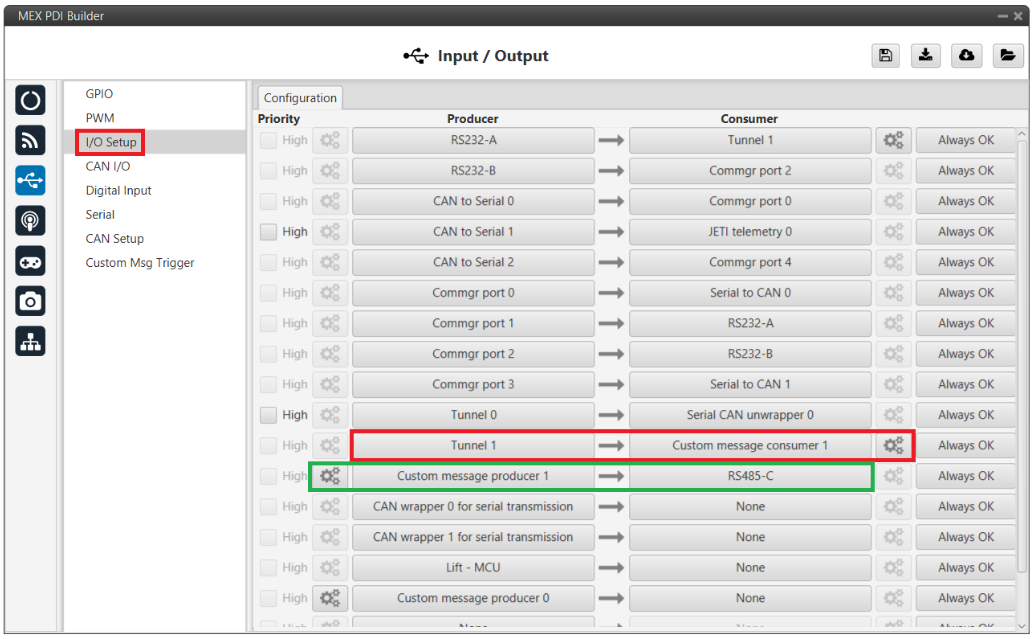
  - Address of a specific Veronte unit, it can be a 1x, a 4x, a CEX, a MEX, etc.
    For more information on the available addresses, see List of addresses section of the **MEX Software Manual**.
- **Parser**: The user can choose the protocol to parse message data. The options available are:
  - No protocol
  - RTCM3
  - CAN serial
- **Destination tunnel**: Number of port is used to avoid mistakes and identify each Tunnel when using more than one. Tunnel 0, 1 and 2 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**
>
> **MEX** has the following limitations for serial messages:
>
> - **10** maximum **vectors**.
> - **32** maximum **fields** (adding all serial messages).

It is possible to configure messages sent and received through the serial port and its conversion to system variables. To do that, select the option **Custom message producer/consumer** and configure the I/O port.



Priority	Producer		Consumer	
□ High	RS232-A	→	Tunnel 1	Always OK
□ High	RS232-B	→	Commgr port 2	Always OK
□ High	CAN to Serial 0	→	Commgr port 0	Always OK
□ High	CAN to Serial 1	→	JETI telemetry 0	Always OK
□ High	CAN to Serial 2	→	Commgr port 4	Always OK
□ High	Commgr port 0	→	Serial to CAN 0	Always OK
□ High	Commgr port 1	→	RS232-A	Always OK
□ High	Commgr port 2	→	RS232-B	Always OK
□ High	Commgr port 3	→	Serial to CAN 1	Always OK
□ High	Tunnel 0	→	Serial CAN unwrapper 0	Always OK
□ High	Tunnel 1	→	Custom message consumer 1	Always OK
□ High	Custom message producer 1	→	RS485-C	Always OK
□ High	CAN wrapper 0 for serial transmission	→	None	Always OK
□ High	CAN wrapper 1 for serial transmission	→	None	Always OK
□ High	Lift - MCU	→	None	Always OK
□ High	Custom message producer 0	→	None	Always OK
□ High	None	→	None	Always OK

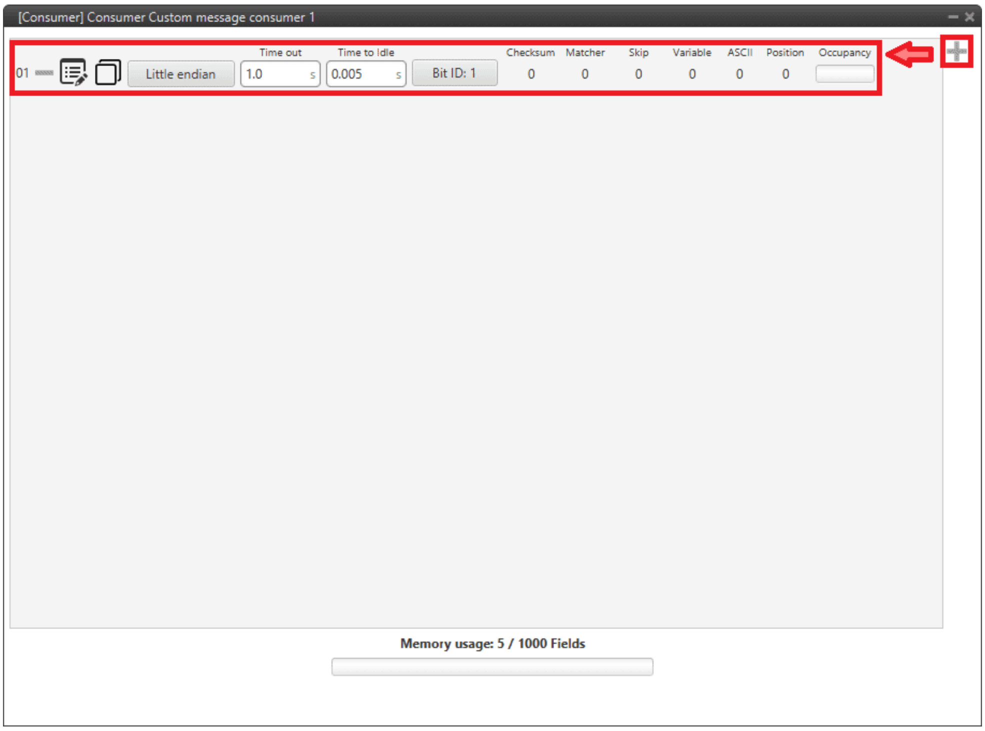
Serial Custom Messages

The previous image illustrates two possible configurations using a Custom Message.

The red one is configured to receive a given message from Tunnel 1 and the green is used to send a Custom Message through the RS485-C serial port. It is also possible to use the same Custom Message for both tasks if the bidirectional relationship is selected (the arrow indicates this, ↔).

To configure a 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.



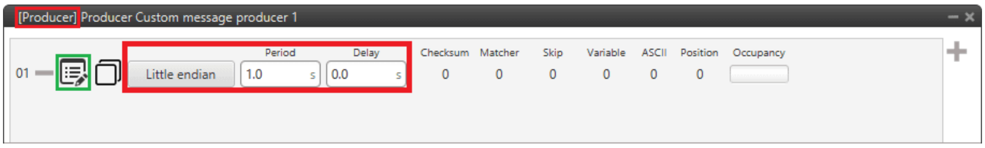
[Consumer] Consumer Custom message consumer 1

	Endianness	Time out	Time to Idle		Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Little endian	1.0 s	0.005 s	Bit ID: 1	0	0	0	0	0	0	

Memory usage: 5 / 1000 Fields

Serial Custom Message configuration

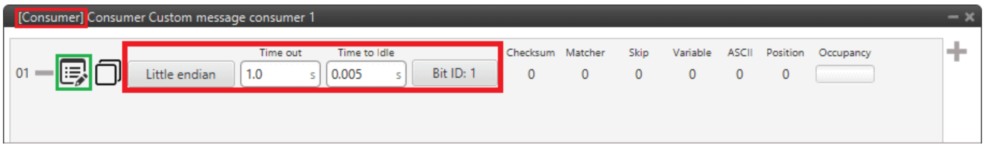
2. When it is already added, the following options are available to configure a custom message:



[Producer] Producer Custom message producer 1

	Endianness	Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Little endian	1.0 s	0.0 s	0	0	0	0	0	0	

Custom Message producer configuration



[Consumer] Consumer Custom message consumer 1

	Endianness	Time out	Time to Idle		Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Little endian	1.0 s	0.005 s	Bit ID: 1	0	0	0	0	0	0	

Custom Message consumer configuration

- **Endianness**: Depending on the order the message is made, it is possible to select:
  - **Big endian**: Set value from left to right.
  - **Little endian**: Set 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 whether it is used to transmit or receive data.
      - **Period - Producer**: Time between produced messages. It is the inverse of the send frequency
      - **Time out - Consumer**: Time threshold between receptions to consider that messages are not received correctly.
   - **Delay/Time to Idle**: This option has a dual role depending on whether it is used to transmit or receive data.
      - **Delay - Producer**: Delay applied before sending the message. This serves to send messages with the same period without overloading the Serial bus.
      - **Time to Idle - Consumer**: Time that MEX 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 📝 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** and **Parse ASCII**.

   The configuration of each structure is covered in the Custom Messages types - Input/Output section of the **1x PDI Builder** user manual.

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

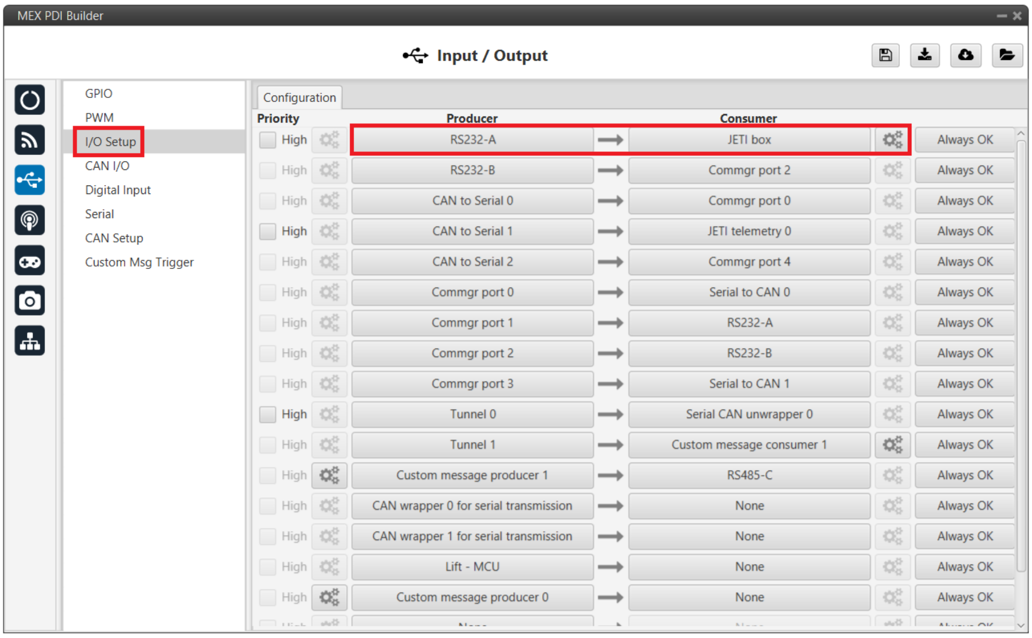
JETI box

JETIBOX is a universal communication terminal which can be used with any JETI products.

JETIBOX operates as a two-way terminal, showing all data stored in the JETIBOX Compatible product. Employing its **four buttons**, users browse its menu and set the selected values.

JETI box device

To simulate it, it is necessary to link the specific **JETI box consumer** to a serial port:

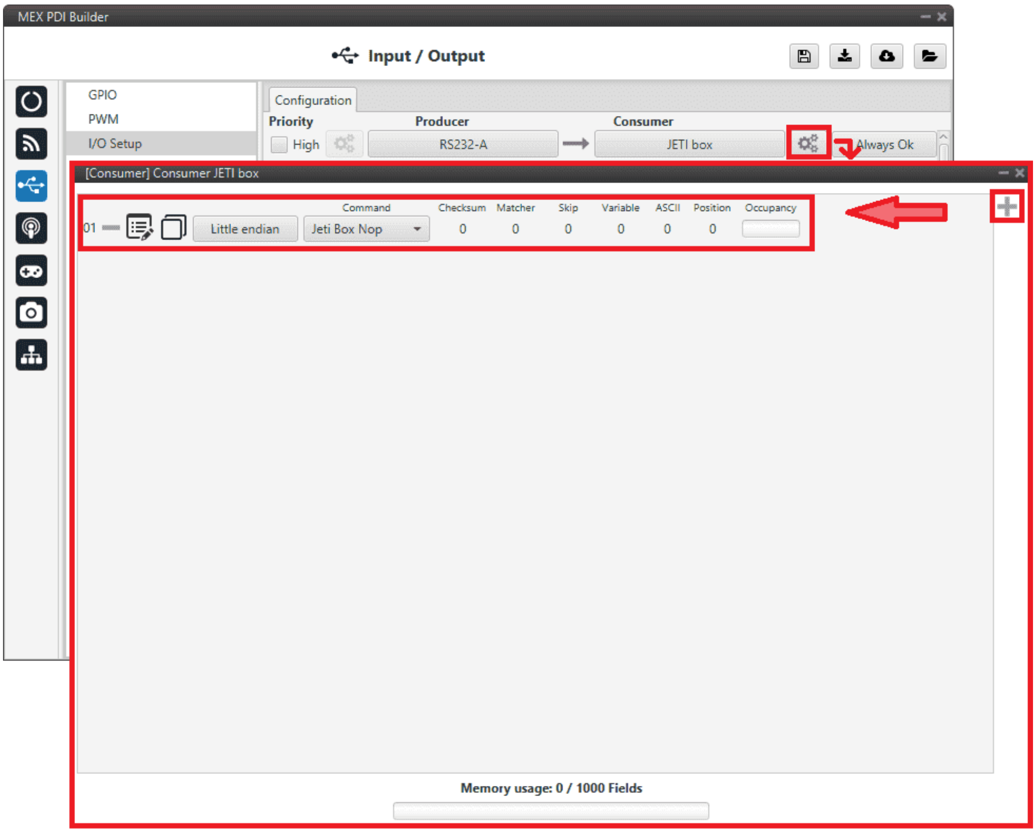


Priority	Producer		Consumer	
High	RS232-A	→	JETI box	Always OK
High	RS232-B	→	Commgr port 2	Always OK
High	CAN to Serial 0	→	Commgr port 0	Always OK
High	CAN to Serial 1	→	JETI telemetry 0	Always OK
High	CAN to Serial 2	→	Commgr port 4	Always OK
High	Commgr port 0	→	Serial to CAN 0	Always OK
High	Commgr port 1	→	RS232-A	Always OK
High	Commgr port 2	→	RS232-B	Always OK
High	Commgr port 3	→	Serial to CAN 1	Always OK
High	Tunnel 0	→	Serial CAN unwrapper 0	Always OK
High	Tunnel 1	→	Custom message consumer 1	Always OK
High	Custom message producer 1	→	RS485-C	Always OK
High	CAN wrapper 0 for serial transmission	→	None	Always OK
High	CAN wrapper 1 for serial transmission	→	None	Always OK
High	Lift - MCU	→	None	Always OK
High	Custom message producer 0	→	None	Always OK

JETI box Consumer

Then, the sequence to retrieve data shall be configured by clicking on ⚙:

To add a Custom Message, just click on the ➕ icon:



Priority	Producer		Consumer	
High	RS232-A	→	JETI box	Always Ok

		Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Little endian	Jeti Box Nop	0	0	0	0	0	0	

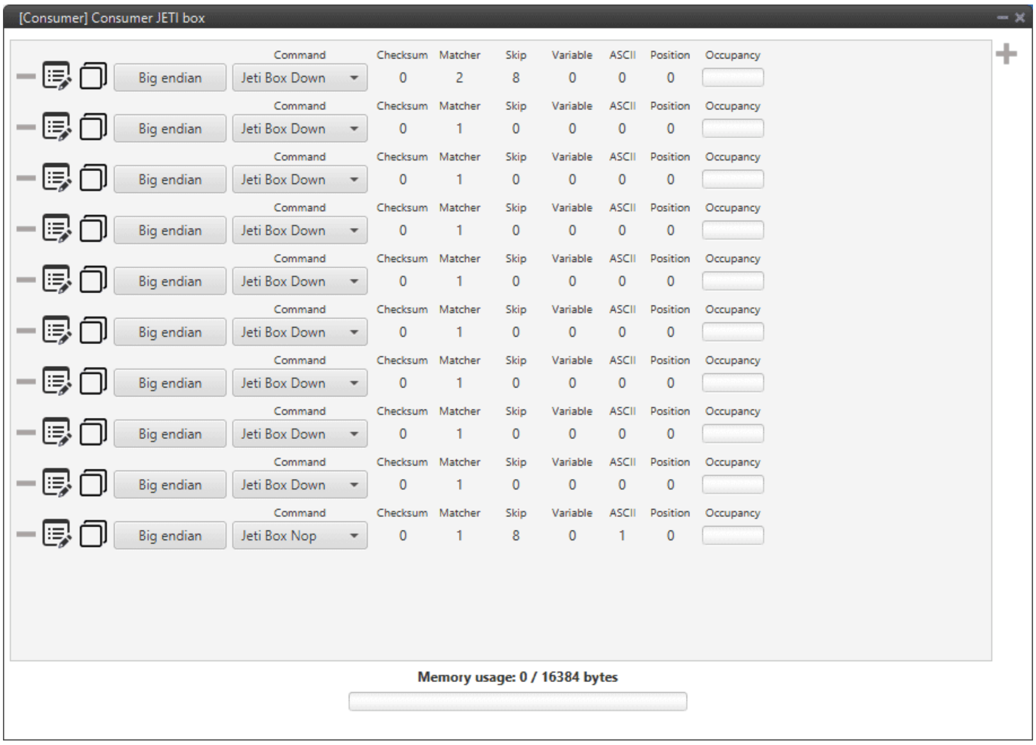
Memory usage: 0 / 1000 Fields

JETI box Configuration

The following parameters can be configured in the previous pop-up window:

- **Endianness**: Depending on the order in which the device issues the message, it is possible to select:
    - **Big endian**: Sets values from left to right.
    - **Little endian**: Sets values 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).
- **Command**: Here the user can select between **Jeti box Left**, **Jeti box Down**, **Jeti box Up**, **Jeti box Right** or **Jeti box Nop**. These correspond to the **four buttons** on the physical JETIBOX (see image above), except the **Jeti box Nop** that is only for simulating a "**wait**".

For example, to read the **Actual Voltage** of a **Jeti MasterSpin 220** the Consumer must be configured with a series of custom messages (use **Big endian** for all messages).



[Consumer] Consumer JETI box

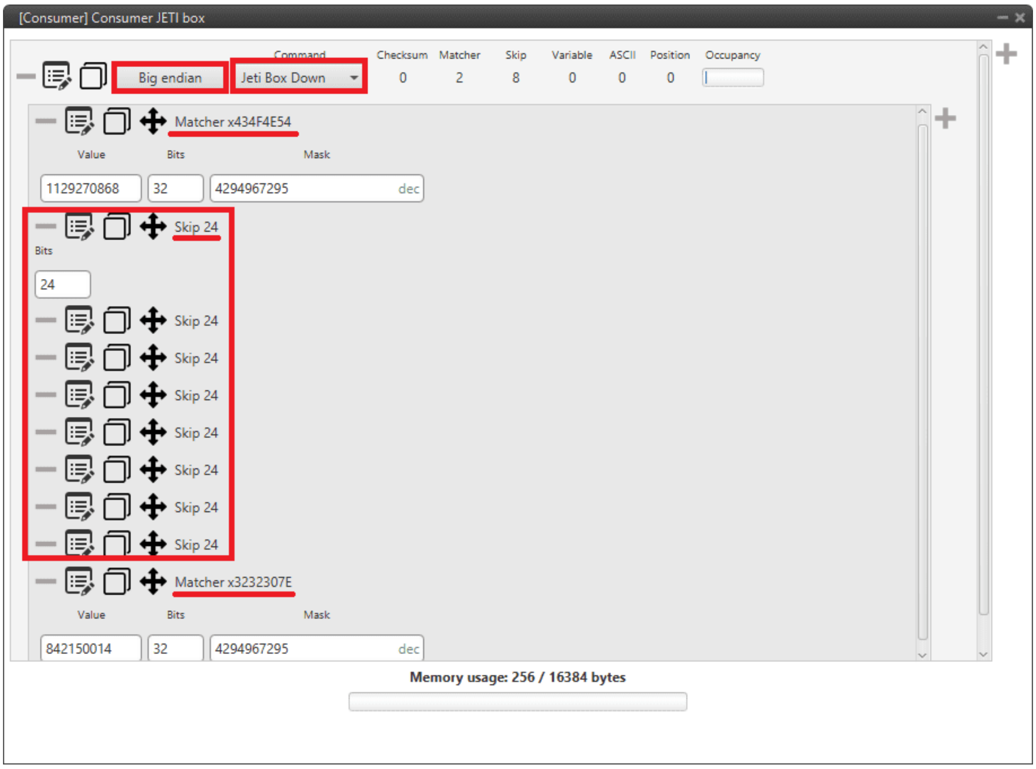
	Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
Big endian	Jeti Box Down	0	2	8	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Nop	0	1	8	0	1	0	

Memory usage: 0 / 16384 bytes

JETI box example

The following example shows the configuration of one of these messages, the full example can be found in Jetibox - Integration examples section of this manual.

- Expected text: "CONTROLLER TYPE MasterSpin 220~"
- Command: **Jeti box Down**
  - Matcher(32) "CONT" 0x434F4E54 (1129270868)
  - Skip(24*8) 192
  - Matcher(32) "220~" 0x3232307E (842150014)



[Consumer] Consumer JETI box

Big endian | Command: Jeti Box Down | Checksum: 0 | Matcher: 2 | Skip: 8 | Variable: 0 | ASCII: 0 | Position: 0 | Occupancy

Matcher x434F4E54 — Value: 1129270868, Bits: 32, Mask: 4294967295 dec

Skip 24 — Bits: 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Matcher x3232307E — Value: 842150014, Bits: 32, Mask: 4294967295 dec

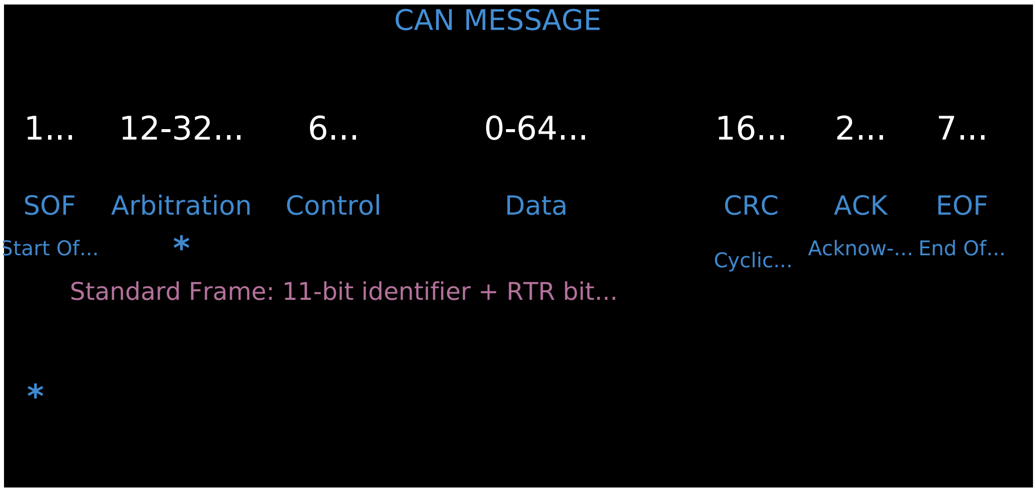
Memory usage: 256 / 16384 bytes

JETI box custom message example

CAN I/O

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

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



CAN message structure

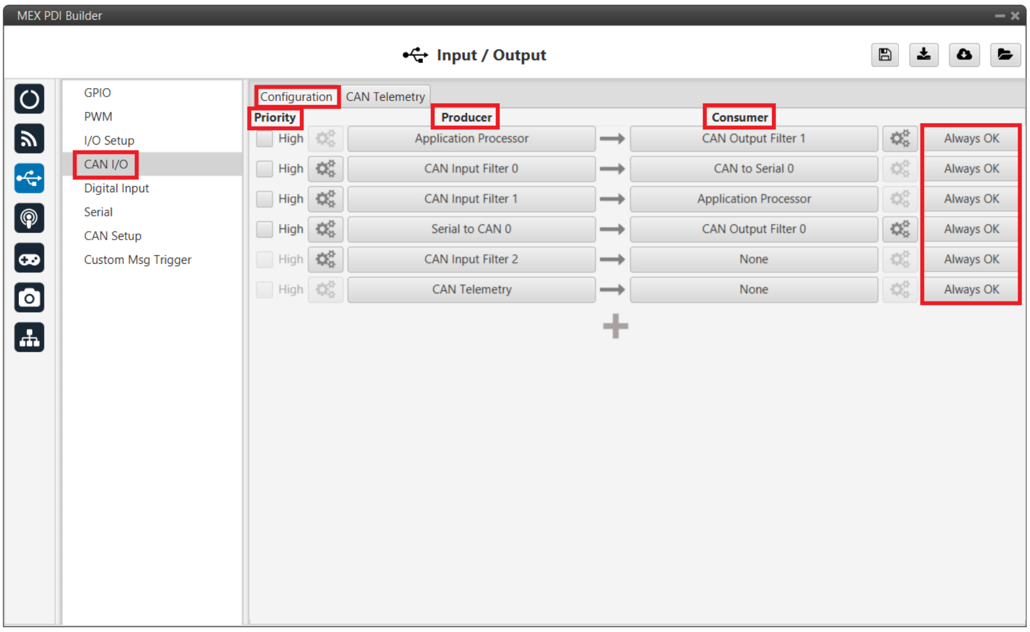
Only the ID is introduced in the system, the rest of the message layout is already coded. The data field is built by the user to send, and parsed when received.

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

The baudrate of both CAN buses can be configured in the CAN Setup panel.

Configuration

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



Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	Application Processor	Always OK
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	CAN Input Filter 2	→	None	Always OK
□ High	CAN Telemetry	→	None	Always OK

CAN I/O - Configuration panel

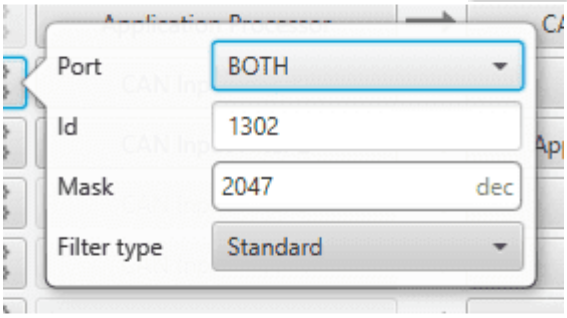
In this menu, the user can find the same 'columns' (**Priority**, **Producer**, **Consumer** and **Bit**) as in the I/O Setup panel. In addition, the process for configuring producers and consumers is also the same as described in the I/O Setup - Input/Output section.

> ⚠ **Warning**
>
> In CAN, while the specified period is not guaranteed in the Low state, in the 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 **MEX**.

On the one hand, **MEX** has the following list of **producers**:

- **Application Processor**: Sends a specific set of information, the "**status message**".
  This message is composed as: version (`major` . `minor` . `revision`), address (serial number), system bit error, system power up bit error, PDI bit error, memory allocation bit, fily system bit error, CAN A bit error, CAN B bit error, arbiter enabled and arbiter status.

- **CAN 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

  CAN 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.

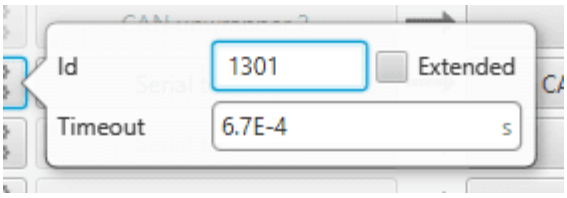
  - **Id**: CAN Id must be set and it is used to identify messages. The value set has to be **decimal format**.
  - **Mask**: A CAN Id mask can be set to filter messages. **The mask marks which bits of the message id (in binary) are matched**.

    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.

- **CAN unwrapper**: This undoes the 'CAN serial wrapper' action, it has to be connected to I/O Setup **consumer** (Serial CAN unwrapper).
- **Serial to CAN**: Serial messages through CAN output, it has to be connected to I/O Setup **consumer**. It can be configured in ⚙, a pop-up window will appear:



Id 1301 ☐ Extended

Timeout 6.7E-4 s

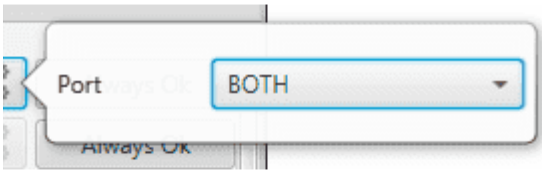
CAN Input Filter configuration

  - **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.

- **CAN Telemetry**: Telemetry messages sent via CAN (such as CAN custom messages on **Veronte Autopilot 1x**). They are configured in the next section: CAN Telemetry.

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

- **Application Processor**: Receives a specific set of information sent by **Veronte Autopilot 1x** or **Arbiter**.
- **CAN Output Filter**: CAN output filters. The user can choose between CAN A, CAN B or BOTH in ⚙.



Port BOTH

CAN Output Filter configuration

- **CAN serial wrapper**: CAN messages through serial output, it has to be connected to I/O Setup **producer** (CAN wrapper for serial transmission).
- **CAN GPIO consumer**: CAN messages from **Autopilot 1x** or **4x** for GPIO inputs.
  An example of how to implement it can be found in GPIO Command - Integration examples section of the present manual.
- **CAN to Serial**: This undoes the 'Serial to CAN' action, it has to be connected to I/O Setup **producer**.

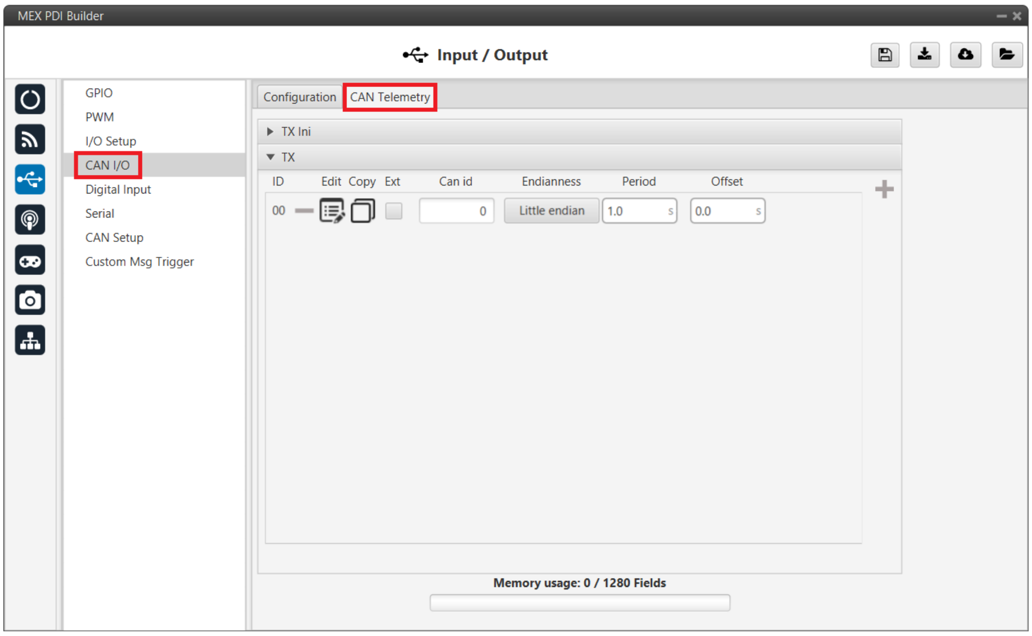
CAN Telemetry

In the CAN Telemetry tab, the user chooses the telemetry 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 MEX boots up). They can be used to initialize some devices.
- **TX**: It is employed to configure transmitted messages.

⚠ **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.
- MEX has the following **CAN limitations**:
- Maximum number of **vectors** (fieldsets): **8 → 1 for TX Ini** and **7 for TX**.
- Maximum number of **fields**: **32** (adding all CAN messages).



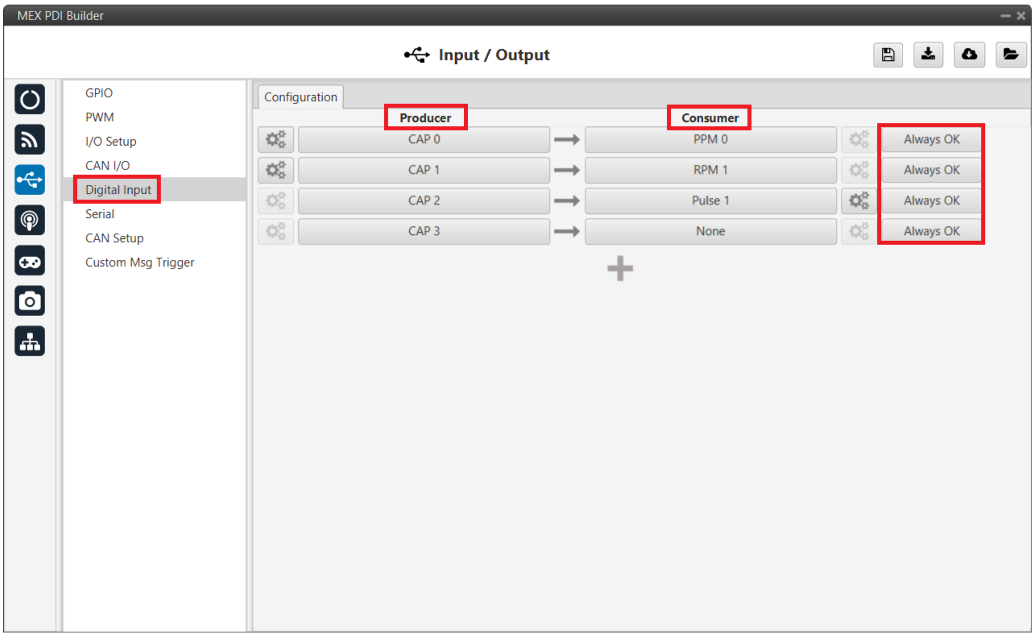
CAN I/O - CAN Telemetry panel

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

An example of the sending of **CAN telemetry messages** can be found in the CAN communication - Integration examples section of this manual.

Digital Input

Digital inputs can be used to measure pulse count, pulse widths and PPM signals from a RC radio. Each source shall be connected to the desired consumer to allow measurements.



Producer		Consumer	
CAP 0	→	PPM 0	Always OK
CAP 1	→	RPM 1	Always OK
CAP 2	→	Pulse 1	Always OK
CAP 3	→	None	Always OK

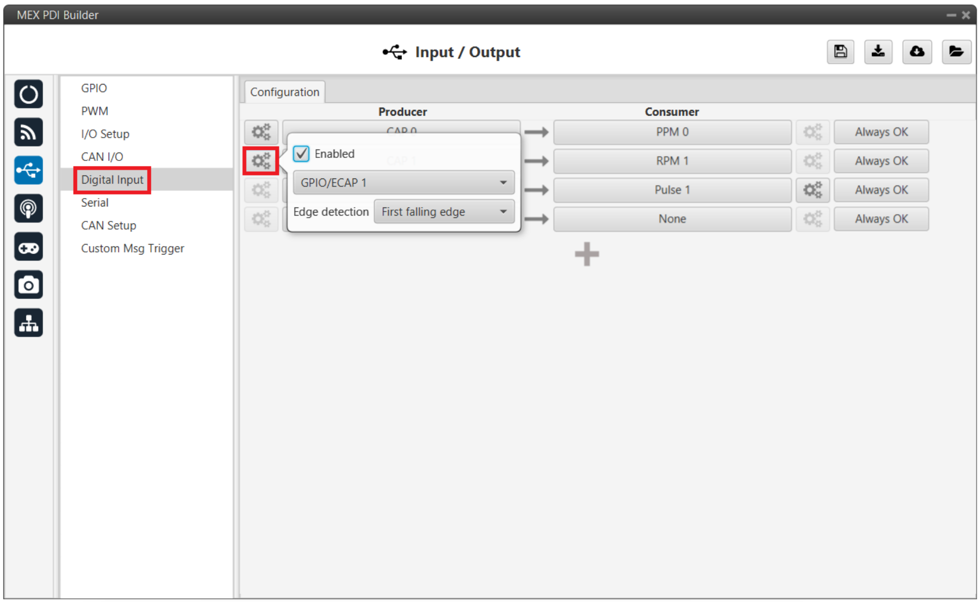
Digital Input panel

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

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

1. Select and configure a **Producer** to read the external signal. There are 2 possible producers: CAP 1 and 2.

   Press on the configuration button



☑ Enabled
GPIO/ECAP 1
Edge detection: First falling edge

Digital Input panel - Producer

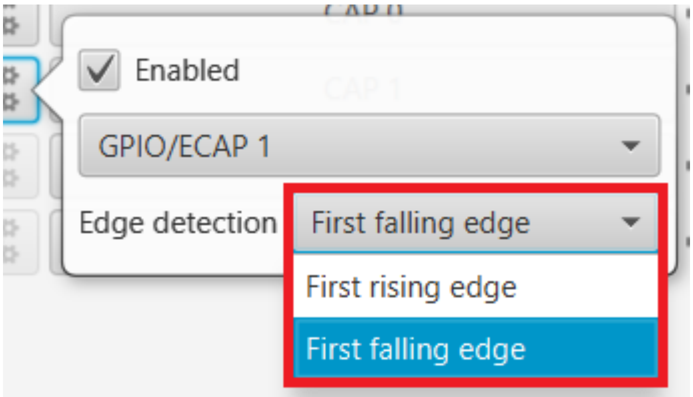
The pop-up window contains the following configurable elements:

- **Enable**: By ticking this checkbox, the corresponding producer is enabled.

  > **Note**
  >
  > When enabling it, in the GPIO menu the "Function" parameter corresponding to this pin shall automatically change to "Mux 1".
  >
  > > ⚠ **Warning**
  > >
  > > Please check that this change has been made.

- **CAP pin entry**: Selects which **pin** this CAP is associated to and, therefore, to which device is connected. They are associated in this way: **CAP 1** with **GPIO/ECAP 1** and **CAP 2** with **GPIO/ECAP 2**.
- **Edge detection**: How the pulses are read and transformed into a digital signal (how they are processed).

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

  ☑ Enabled

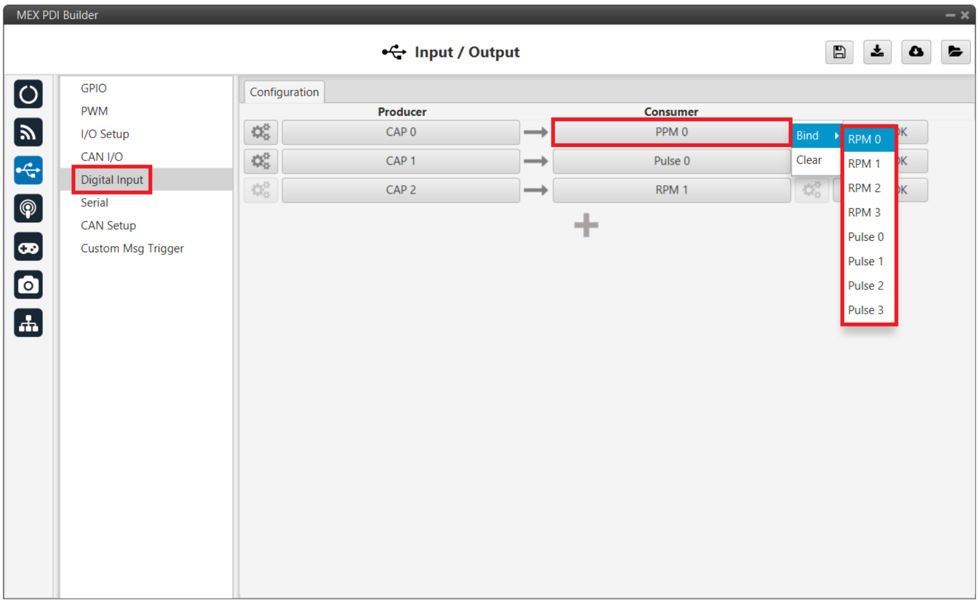
  GPIO/ECAP 1

  Edge detection: First falling edge

  Digital Input panel - Edge detection option

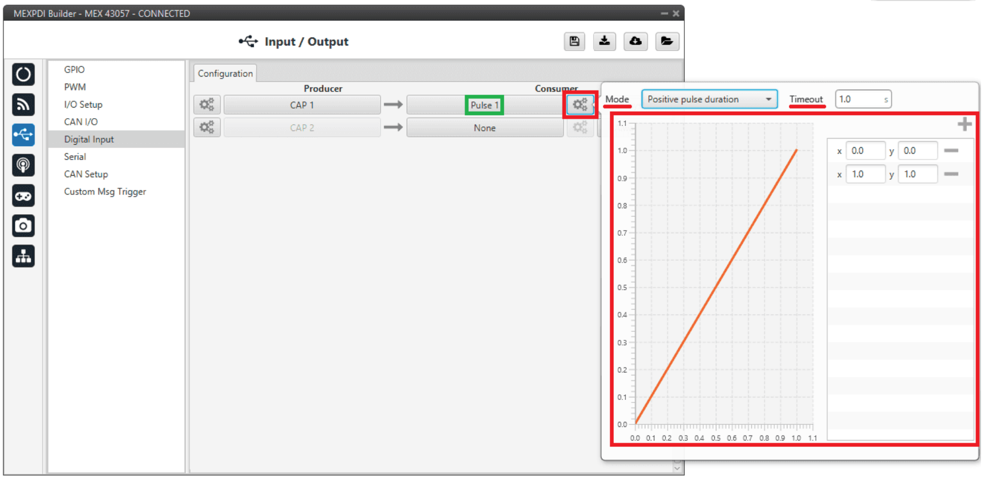
  - **First rising edge**: With this option, when the rise of the pulse is detected, the data will start to be stored. Recommended when consumer is **PPM** or **Pulse**.
  - **First falling edge**: With this option, when the **fall** of the pulse is detected, the data will start to be stored.

2. Click on the **Bind** button to select the type of **Consumer**, it is possible to choose PPM0 (Stick PPM), RPM 1-4 (RPM Sensor) or Pulse 1-4 (Pulse Sensor).



Digital Input panel - Consumer

- **PPM 0**: The variable where the information is stored is 'PPM channel 1 output'. Stick PPM is configured in the Stick panel.
- **RPM 0-3**: The variables where the information read here is stored are 'RPM 0-3'. For more information about RPM configuration, read the RPM section.
- **Pulse 0-3**: The variables where the information is stored are 'Captured pulse 0-3'. It is possible to configure it clicking on ⚙.



Mode: Positive pulse duration

Timeout: 1.0 s

x 0.0 y 0.0

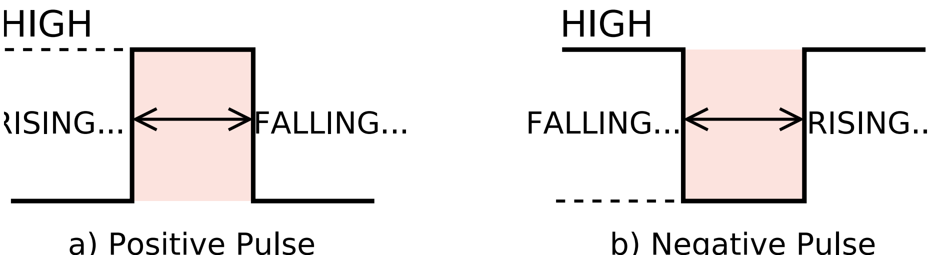
x 1.0 y 1.0

Digital Input panel - 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.



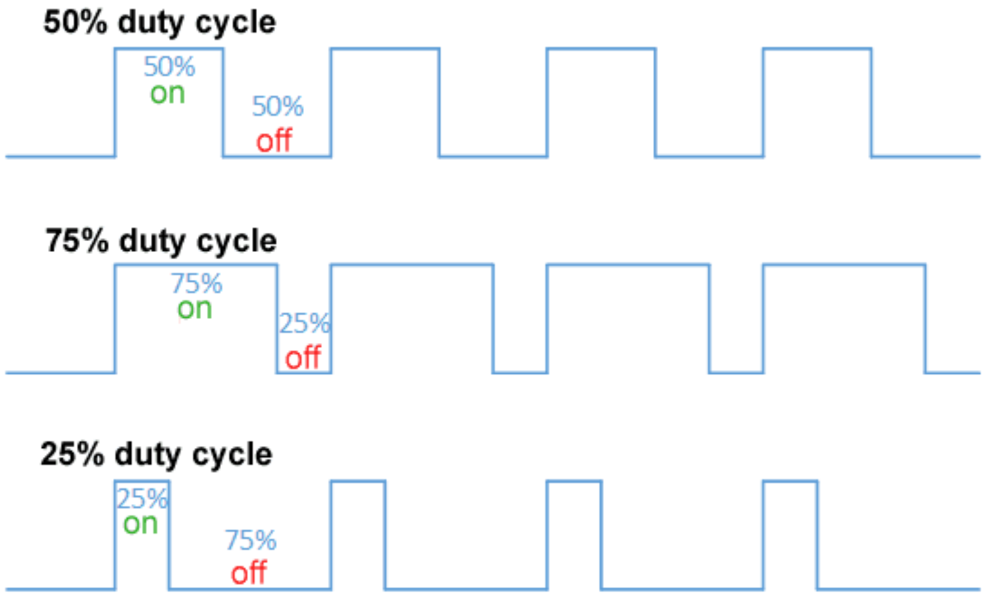
a) Positive Pulse

Text is not SVG - cannot display

b) Negative Pulse

Positive/Negative pulse duration

    - **Positive duty cycle**: The duty cycle of the pulse is obtained. It takes the time in 'High' state.
    - **Negative duty cycle**: The duty cycle of the pulse is obtained. It takes the time in 'Low' state.

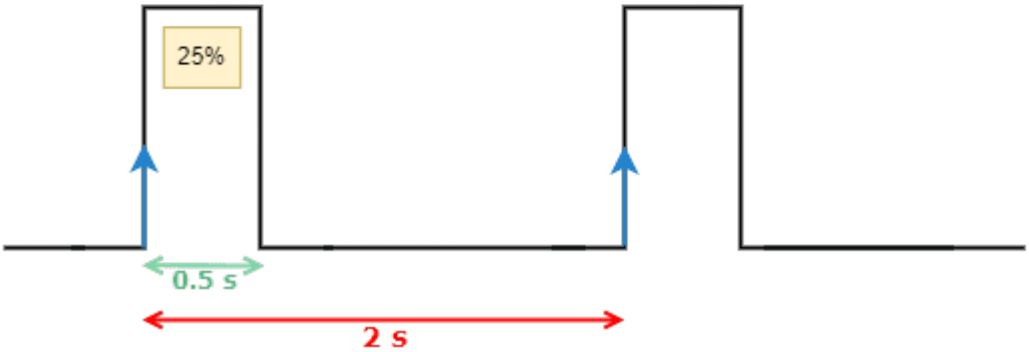


Positive/Negative duty cycle

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

Example

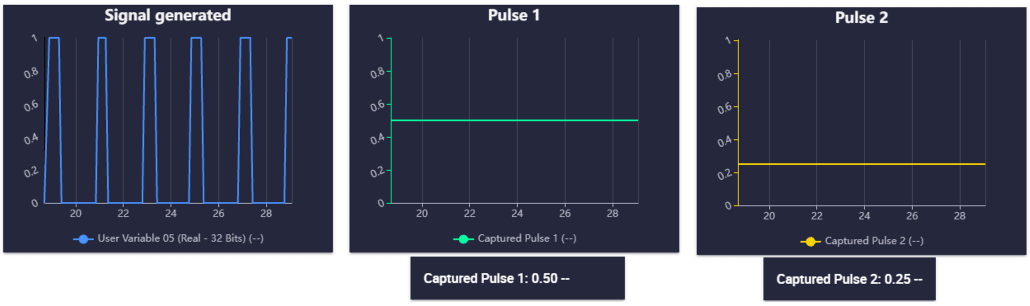
Let's imagine that **First rising edge** has been selected as the edge detection option in Producer and the pulse that **MEX** has to read is a square signal with a period of 2 seconds and a duty cycle of 25% (see the image below).



Signal generated

Therefore, if **Positive pulse duration** is selected as Consumer and it is configured as in the previous image (Digital Input - Pulse), it will get **0.50 s** as value in the variable **Captured pulse** (Captured pulse 1 in the following example), since it corresponds to the period of the "positive pulse" of that pulse.

Nonetheless, 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**, since it corresponds to the positive cycle of that pulse.



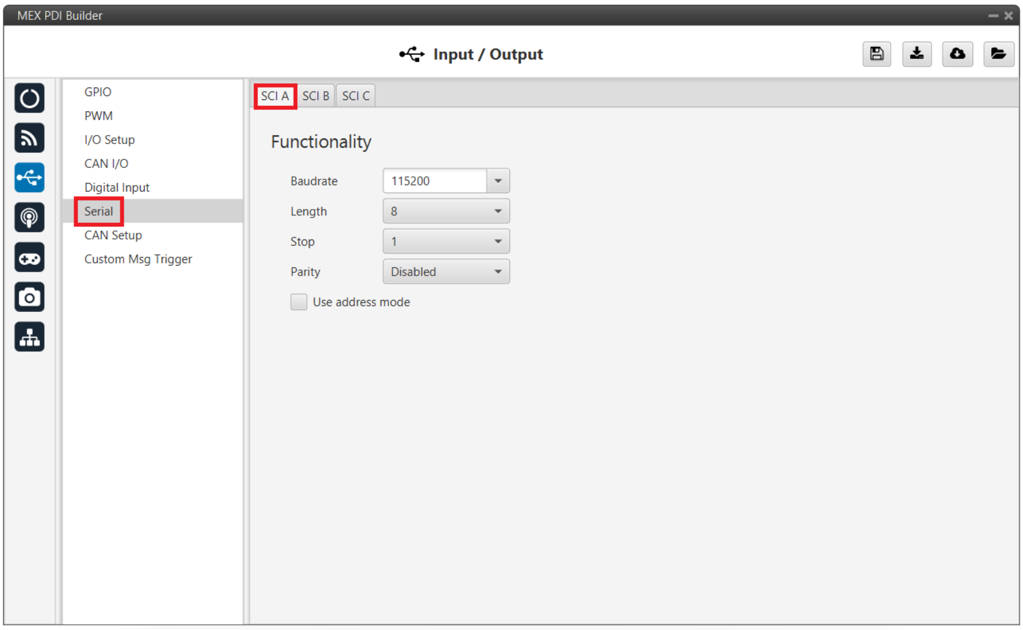
Captured Pulse 1: 0.50 --

Captured Pulse 2: 0.25 --

Digital Input example

Serial

MEX can use up to three serial peripherals (SCI A, SCI B and SCI C). Serial ports A, B and C parameters can be edited in this menu to fit the serial protocol requirements.



Functionality

Baudrate 115200

Length 8

Stop 1

Parity Disabled

□ Use address mode

SCI panel

- **Baudrate**: This specifies 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**: Method to detect errors during 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.

> **Note**
>
> SCI A corresponds to port RS232-A, SCI B to port RS232-B and SCI C to port RS485-C.

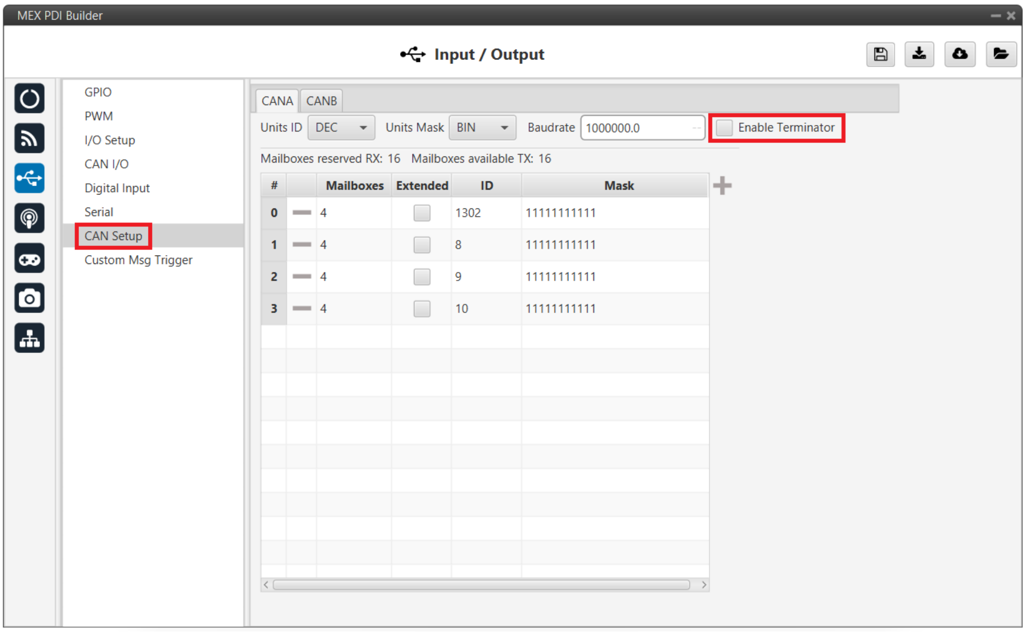
CAN Setup

In this screen users can configure the **baudrate** and the **reception mailboxes** of each **CAN Bus** (CAN A or CAN B), as well as enable an **internal CAN resistor**.

Since **MEX** is going to receive data on the CAN Bus, it is mandatory to configure a certain number of mailboxes to store that data until **MEX** 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 **MEX 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 □ Enable Terminator

Mailboxes reserved RX: 16 Mailboxes available TX: 16

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

CAN Setup panel

By activating '**Enable Terminator**', the internal **MEX** resistor of 120 Ω is activated. For more information on this resistor, refer to the Electrical diagram of CAN bus - Hardware Installation section of the **MEX Hardware Manual**.

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

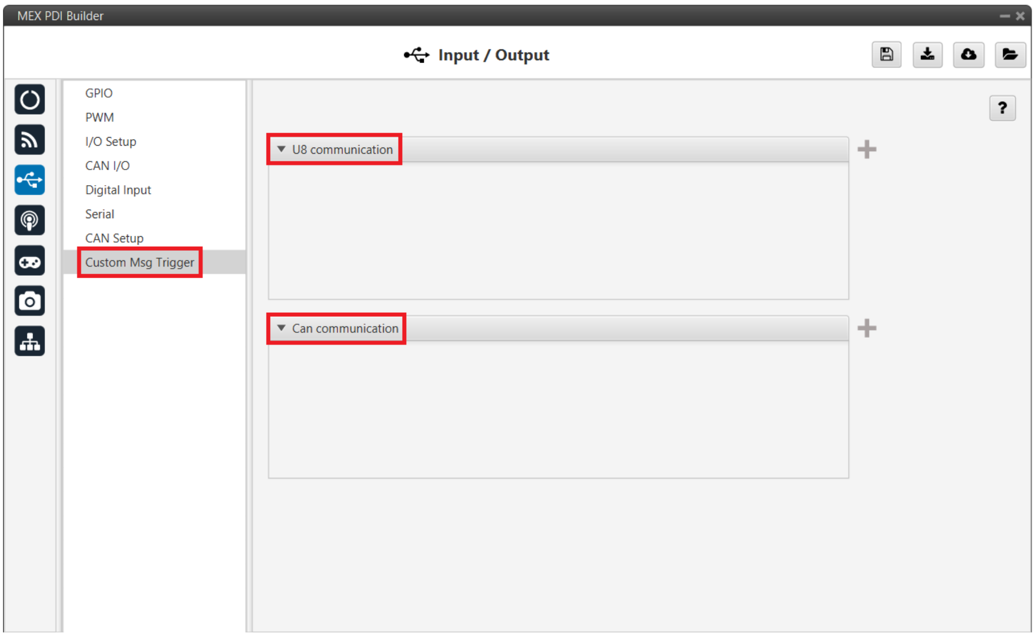
Custom Msg Trigger

This menu allows the user to **send custom messages** (which have been previously configured) every time there is a **new magnetometer reading/measurement**.

It works similar to the Custom Serial TX and Custom CAN TX automations of **1x PDI Builder** software.

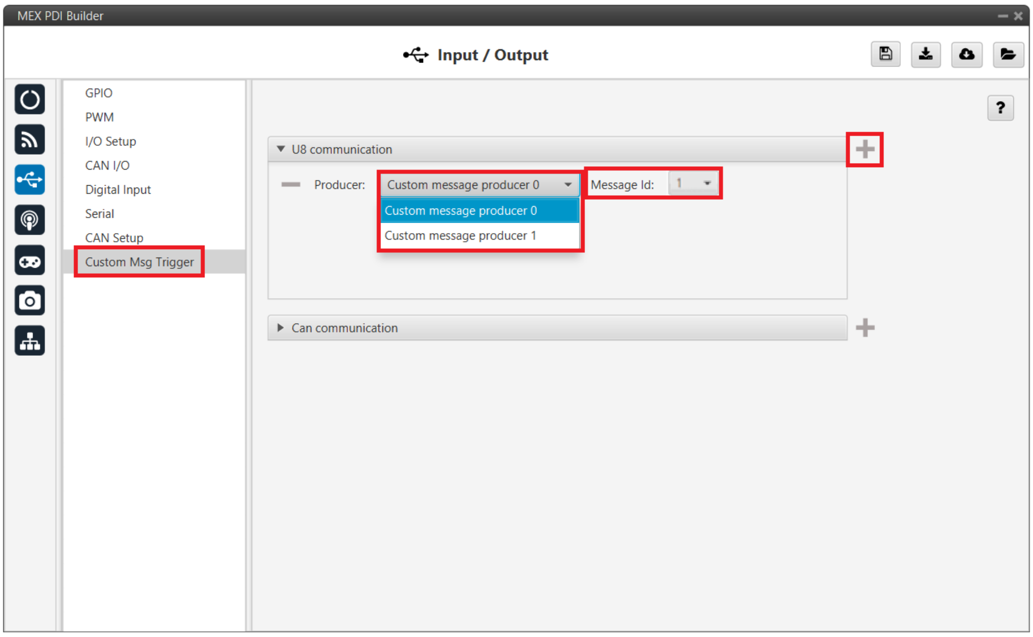
In this case, the "automation" will work like this:

- Events: Each new magnetometer reading/measurement.
- Action: Sending of the custom message already configured.



Custom Msg Trigger panel

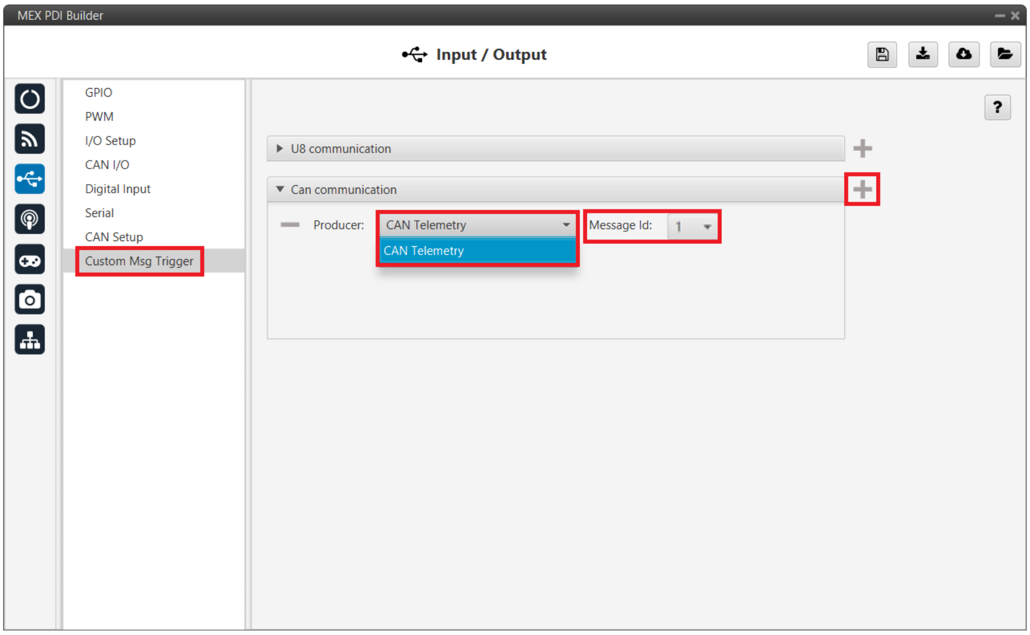
Users can configure custom messages to be sent via **serial** or **CAN**:



Producer: Custom message producer 0

Message Id: 1

Custom Msg Trigger panel - U8 communication



Producer: CAN Telemetry

Message Id: 1

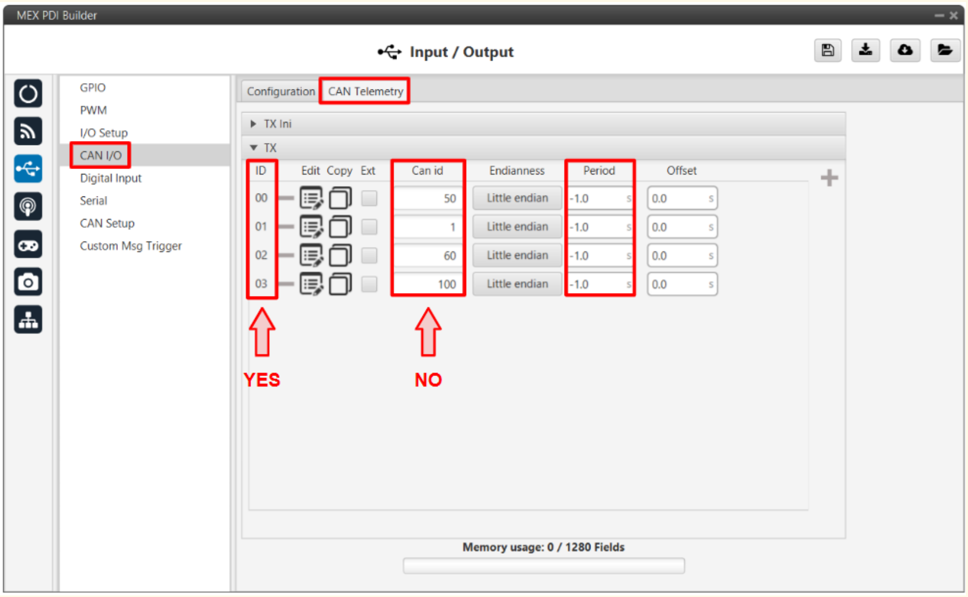
Custom Msg Trigger panel - CAN communication

The two parameters to configure in this menu are:

- **Producer**: The user has to specify where the custom message is located (mailbox):
    - For **serial**: Custom message producer 1 or 2.
    - For **CAN**: CAN Telemetry
- **Message ID**: The ID of the custom message (not the CAN ID) that will be sent.

⚠ **Warning**

- As it is used to send a single message on demand, in its configuration in **Custom Messages**, the user must set its **period to -1**. This way, this **message will only be sent when there is a new magnetometer reading/measurement**.
- For CAN communications, do not confuse it with the **CAN ID** of the custom message:



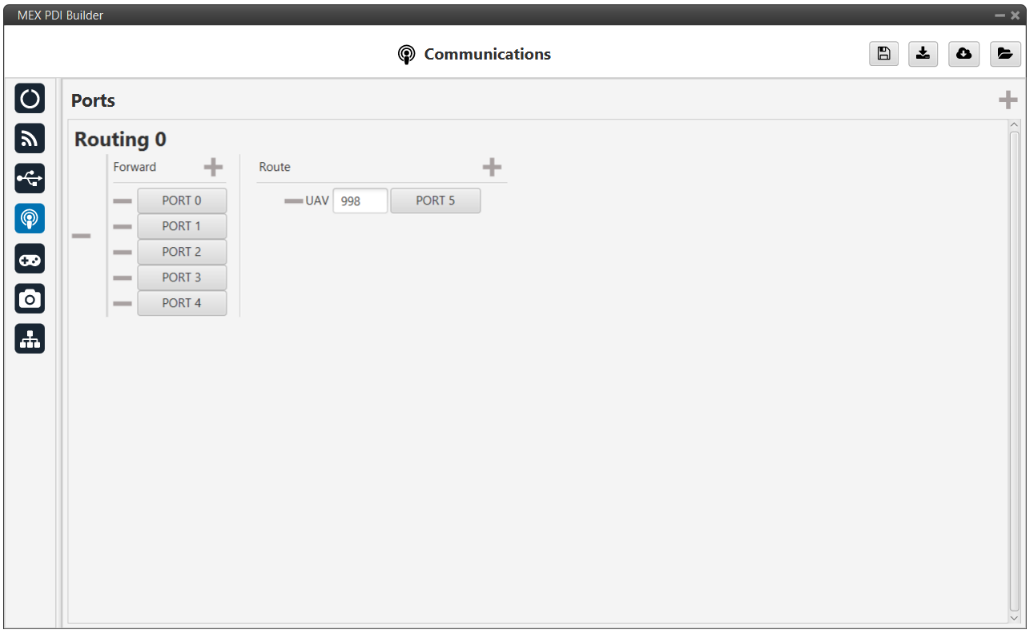
ID	Edit	Copy	Ext	Can id	Endianness	Period	Offset
00			□	50	Little endian	-1.0 s	0.0 s
01			□	1	Little endian	-1.0 s	0.0 s
02			□	60	Little endian	-1.0 s	0.0 s
03			□	100	Little endian	-1.0 s	0.0 s

Memory usage: 0 / 1280 Fields

Message ID in CAN communication

Communications

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



Communications menu

Each port 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 using the Ports action in the **Automations** menu of the **1x PDI Builder** software.

Routing 1 will always be selected by default when booting **MEX**.

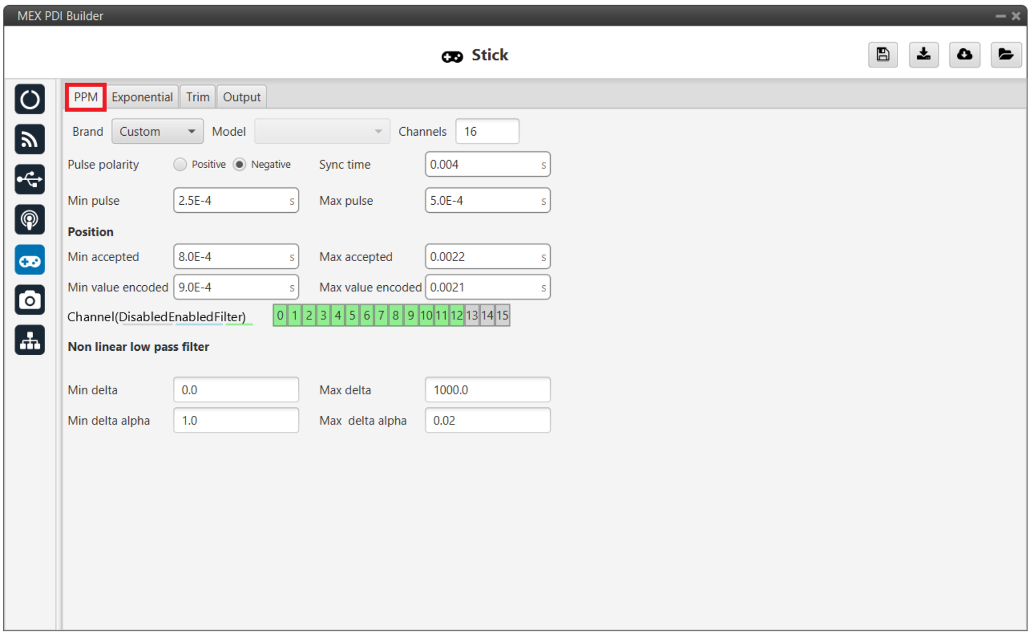
Stick

The wired transmitter is configured through the following tabs, which cover the following content:

- 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 autopilot.

PPM

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



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) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Non linear low pass filter

Min delta 0.0
Max delta 1000.0
Min delta alpha 1.0
Max delta alpha 0.02

PPM panel

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

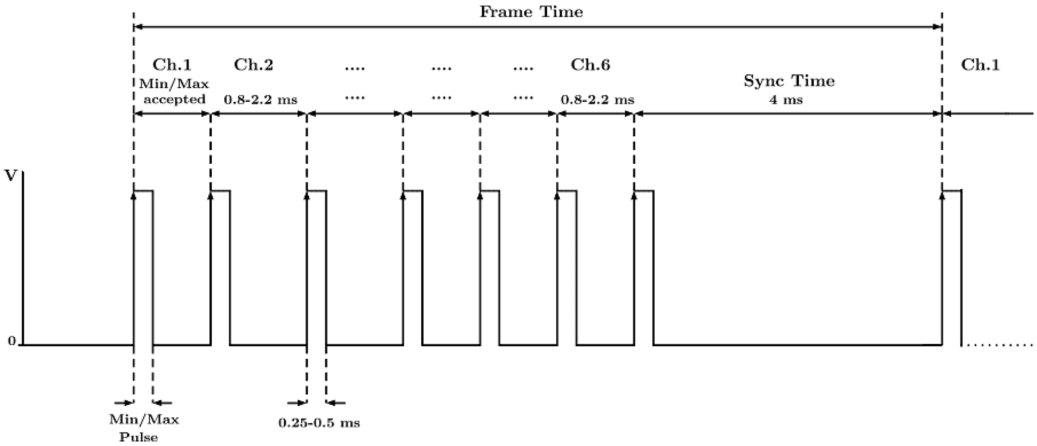
Brand	Models	Channels
Futaba	8J/10J/12K/14SG	8 (for 8J and 10J)

Brand	Models	Channels
		12 (for 12K and 14SG)
	T18SZ	8
Jeti	DC 16/DC 24	16
FrSky	Taranis X9D	8
	Horus X12S	8
TBS	Crossfire	8
Embetion	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, **MEX** 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**: Sets the number of channels accepted. 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 **MEX**:

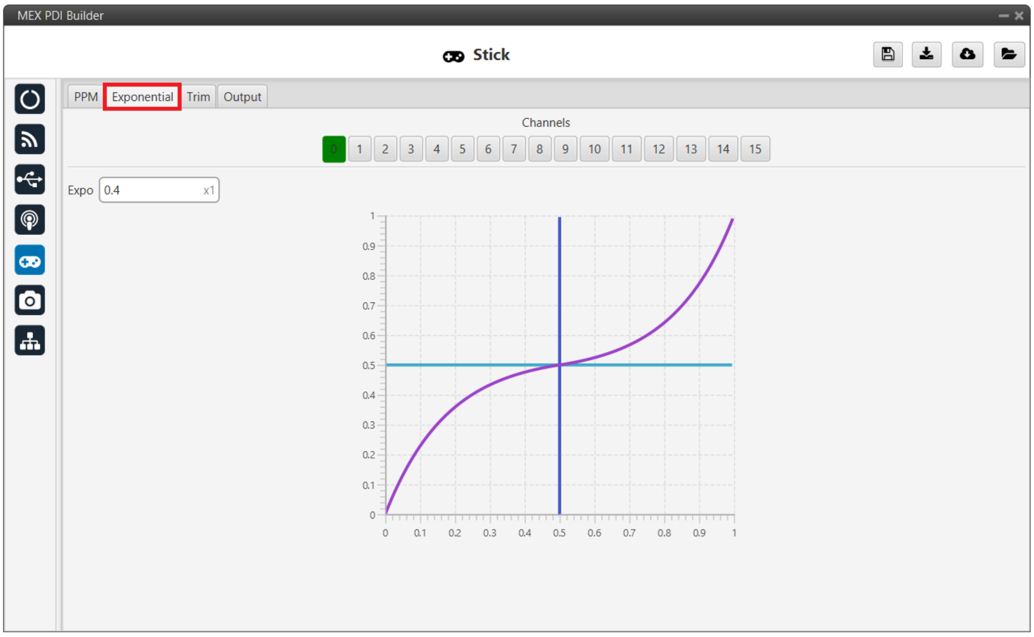


PPM signal

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.



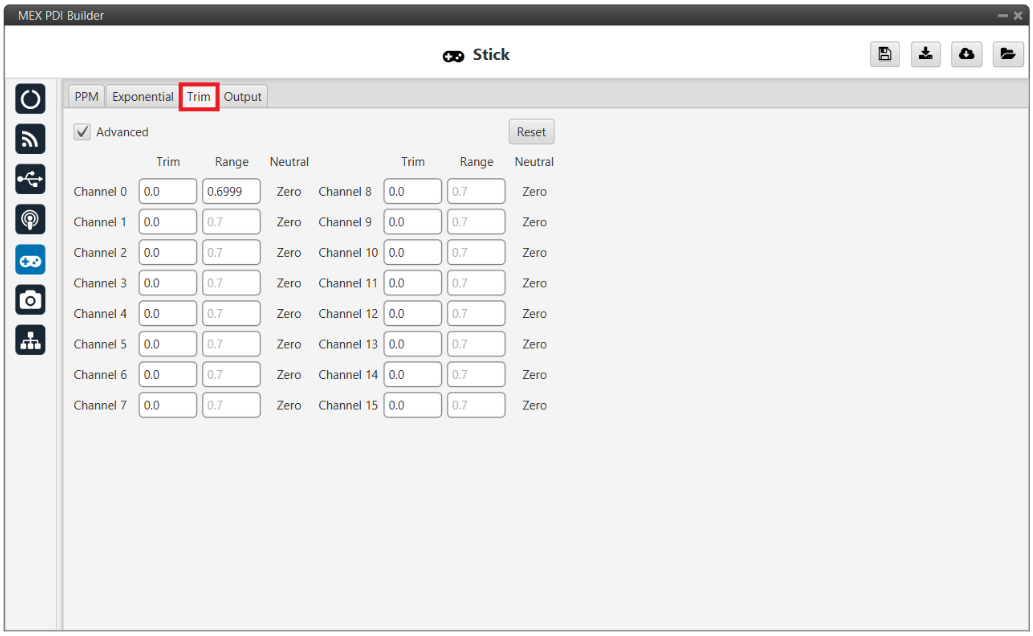
Expo 0.4 x1

Exponential panel

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.

Trim

The third tab available is the Trim option.



☑ Advanced

	Trim	Range	Neutral		Trim	Range	Neutral
Channel 0	0.0	0.6999	Zero	Channel 8	0.0	0.7	Zero
Channel 1	0.0	0.7	Zero	Channel 9	0.0	0.7	Zero
Channel 2	0.0	0.7	Zero	Channel 10	0.0	0.7	Zero
Channel 3	0.0	0.7	Zero	Channel 11	0.0	0.7	Zero
Channel 4	0.0	0.7	Zero	Channel 12	0.0	0.7	Zero
Channel 5	0.0	0.7	Zero	Channel 13	0.0	0.7	Zero
Channel 6	0.0	0.7	Zero	Channel 14	0.0	0.7	Zero
Channel 7	0.0	0.7	Zero	Channel 15	0.0	0.7	Zero

Trim panel

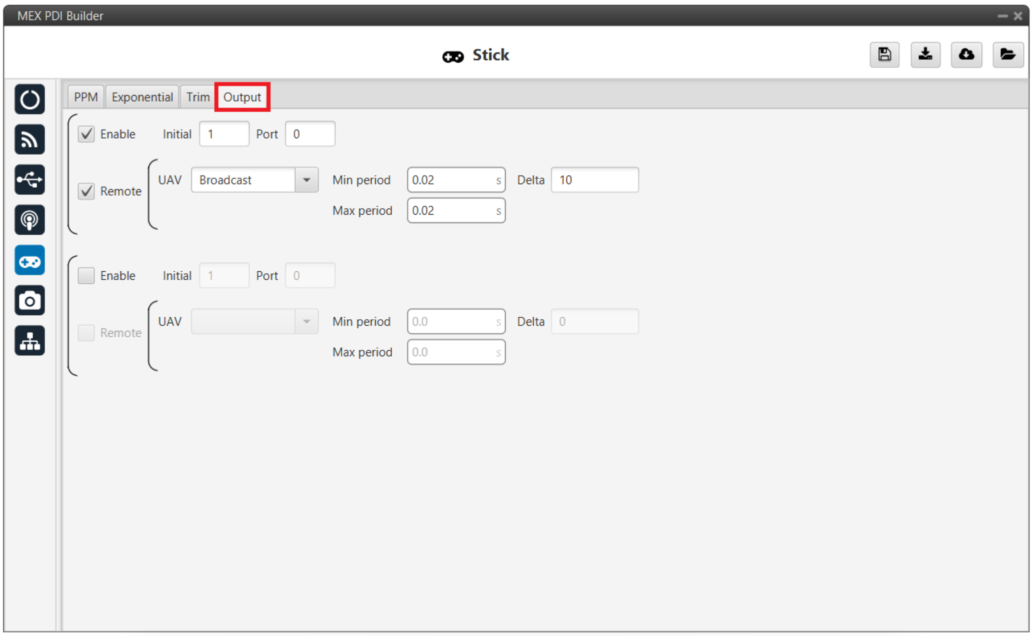
By enabling the **Avanced** option, the user can set the expected trim values manually. The user should have a deep knowledge on its transmitter if this option is selected.

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

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 ground autopilot have to be sent to the air unit.



☑ Enable Initial 1 Port 0
☑ Remote UAV Broadcast Min period 0.02 s Delta 10
Max period 0.02 s

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

Output panel

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 the most common:
        - App 2: Veronte applications address.
        - Broadcast: The commands are sent to all units on the network. **This option is recommended**.
        - 1x v4.X XXXX: The address of a specific air unit.

          For more information on the available addresses, see List of addresses section of the **MEX Software Manual**.
    - **Min period**: As the period is the inverse of the frequency, this defines the **maximum frequency**. Therefore, to give the pilot more control, this defines the frequency that is set when the **stick is commanding**. A Min period of **0.02s** is recommended.
    - **Max period**: As the period is the inverse of the frequency, this determines the **minimum frequency**. Thus, to free up bandwidth, this defines the frequency that is set when the **stick is idle**. A Max period of **0.2s** is recommended.
    - **Delta**: This parameter determines whether the frequency is set to the minimum or maximum period set above.

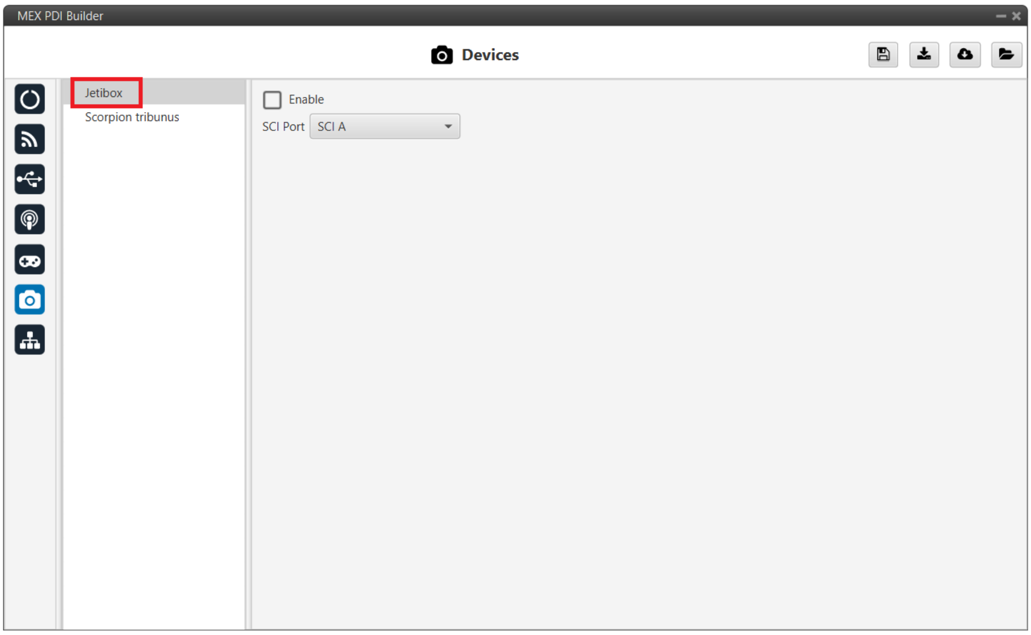
      If **MEX** detects a **change above** the **delta value**, the frequency goes to the **maximum frequency** (minimum period). While if the **changes are less than this value**, it switches to the **minimum frequency** (maximum period). **10 Hz** are recommended.

> ⓘ **Note**
>
> An example of how to configure a stick can be found in the Stick - Integration examples section of the **1x PDI Builder** user manual.

Devices

Jetibox

MEX can simulate a Jetibox to read telemetry from Legacy Jeti devices.



Jetibox panel

- **Enable**: It can be enabled or disabled by the user.
- **SCI Port**: A SCI port must be selected. The available options are: **SCI A**, **SCI B** or **SCI C**.

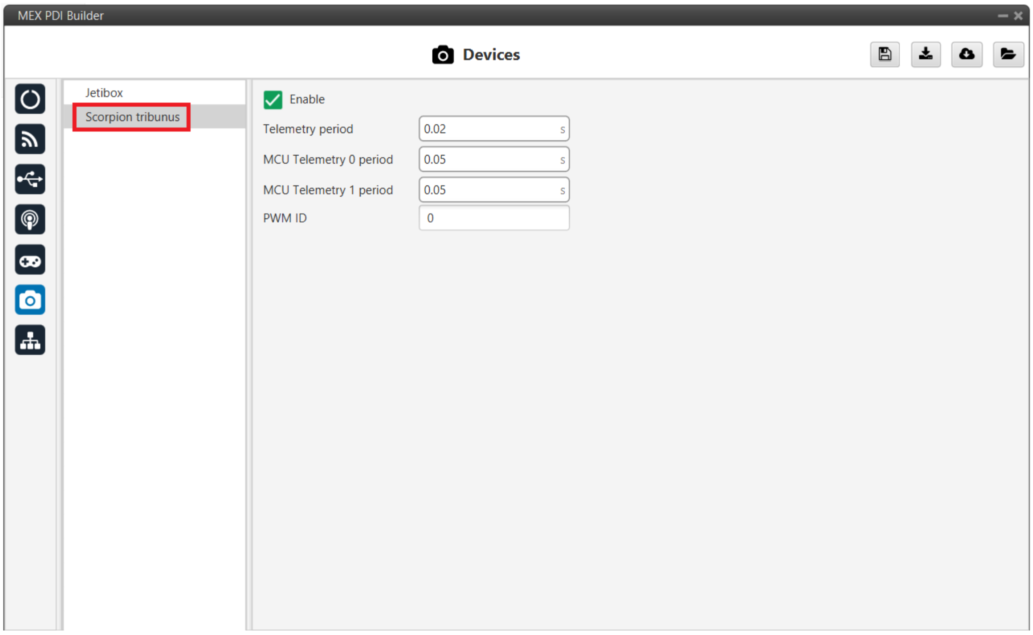
Besides, a configuration is needed in the Serial and I/O Setup panels.

An example of this can be seen in the Jetibox - Integration examples section of the present manual.

> ⓘ **Note**
>
> The serial port will be totally reserved for this, so it will not be usable to other things and the I/O Setup affecting it wil be ignored.

Scorpion tribunus

MEX can read telemetry from Tribunus ESC connecting it to a serial port.



☑ Enable

Telemetry period 0.02 s

MCU Telemetry 0 period 0.05 s

MCU Telemetry 1 period 0.05 s

PWM ID 0

Scorpion tribunus panel

The following parameters are configurable:

- **Enable**: It can be enabled or disabled by the user.
- **Telemetry period**.
- **MCU telemetry 0/1 period**: Telemetry period for MCU devices.
- **PWM ID**: PWM ID associated to the Scorpion Tribunus.

The serial port has to be configured in the I/O Setup panel.

An example is explained in the Scorpion tribunus - Integration examples section of this manual.

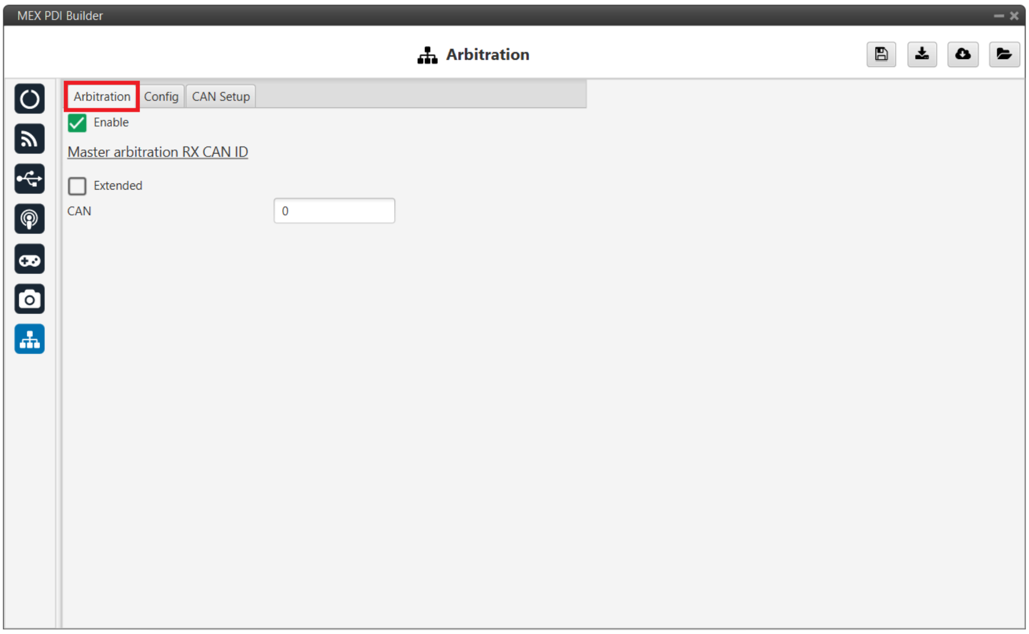
> **Note**
>
> The serial port will be totally reserved for this, so it will not be usable by other things and the I/O setup affecting it wil be ignored.

Arbitration

The arbitration algorithm is explained in detail in the Arbitration section of te **4x PDI Builder** user manual.

Arbitration

MEX is able to send PWMs using arbitration in the same way **Veronte Autopilot 4x** does. This functionality requires to be enabled in the **Enable** checkbox.



☑ Enable

Master arbitration RX CAN ID

□ Extended

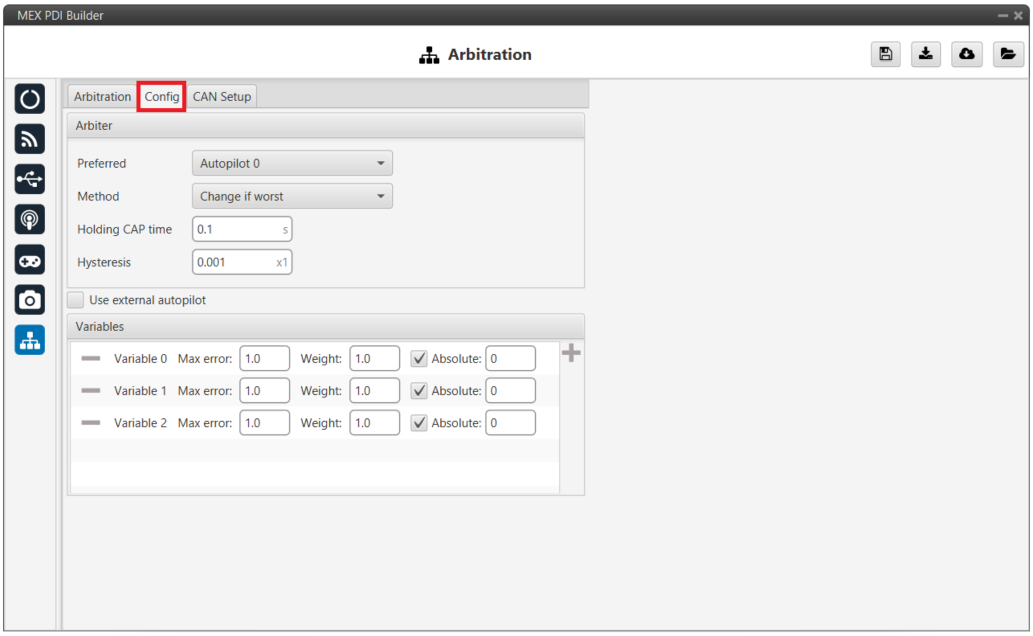
CAN 0

Arbitration panel

Master arbitration RX CAN Id is exclusive for **MEX** when it is used alongside a **Veronte Autopilot 4x**. If **enabled**, when an arbitration message is received from the Arbiter of the **Autopilot 4x** (**with the CAN Id configured here**), the selected autopilot will be updated according to the data received.

Config

In this tab the parameters of the arbiter algorithm are set.



MEX PDI Builder

Arbitration

Arbitration | Config | CAN Setup

Arbiter

Preferred: Autopilot 0

Method: Change if worst

Holding CAP time: 0.1 s

Hysteresis: 0.001 x1

☐ Use external autopilot

Variables

Variable 0	Max error: 1.0	Weight: 1.0	☑ Absolute: 0
Variable 1	Max error: 1.0	Weight: 1.0	☑ Absolute: 0
Variable 2	Max error: 1.0	Weight: 1.0	☑ Absolute: 0

Configuration panel

- **Preferred**: The preferred autopilot will be chosen in case of a score draw. **Fixed while ok** mode will always select this autopilot first.
- **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 0 is selected.
        - **Fixed 1**: Autopilot 1 is selected.
        - **Fixed 2**: Autopilot 2 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**.
- **Holding CAP time**: Amount of time needed from last switch in order to allow a new switch.

- **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.

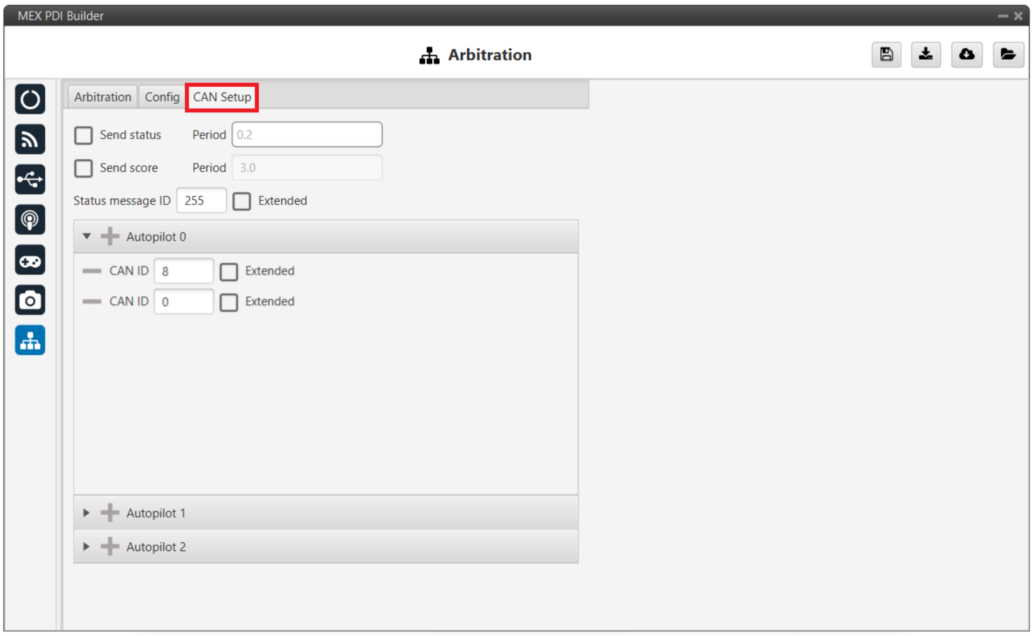
  i.e: If current selected autopilot is the number 1, arbitration mode is **Always best**, hysteresis is **0.5** and score for AP 1 is **0.3**, AP 2 will need a score lower than **0.15** in order to be selected.
- Enable arbitration of **external autopilot**.
- **Variable - Max error**: Arbitration maximum error for each variable.
- **Variable - Weight**: Arbitration weight for each variable.
- **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**.

CAN Setup

This menu allows to configure the receiving CAN Ids for each of the 3 possible **Veronte Autopilots 1x** that are sending data to **MEX**. Therefore, to send data from any of them to **MEX**, these Ids must be specified.

> ⓘ **Note**
>
> If arbitration is not enabled, only the configuration for Autopilot 0 will be used.



□ Send status Period 0.2

□ Send score Period 3.0

Status message ID 255 □ Extended

Autopilot 0

CAN ID 8 □ Extended

CAN ID 0 □ Extended

Autopilot 1

Autopilot 2

CAN Setup panel

- **Send status and Period**: The user can enable status message sending and set its period.

> **Note**
>
> The Send status Period must be set to **0.2 s**, otherwise the arbitration algorithm will force the selection of autopilot 0.

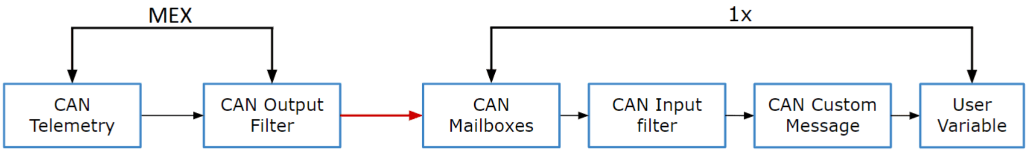
- **Send score and Period**: Users can enable the score message and define its period.
- **Status message ID**: CAN ID to which Status and Score messages are sent when there is **no External Arbitration** and therefore **MEX** works as **arbiter**.
    - **Extended**: If enabled, the frame format will be '**Extended**' (with a **29-bit identifier**). Otherwise, the frame format '**Standard**' (with a **11-bit identifier**) is set by default.
- **Autopilot 0-2**: CAN IDs used for the reception of autopilot 4x arbitration messages for each autopilot.
    - **Extended**: If enabled, the frame format will be '**Extended**' (with a **29-bit identifier**). Otherwise, the frame format '**Standard**' (with a **11-bit identifier**) is set by default.

Integration examples

CAN communication

MEX can send and receive messages via CAN.

CAN messages transmission

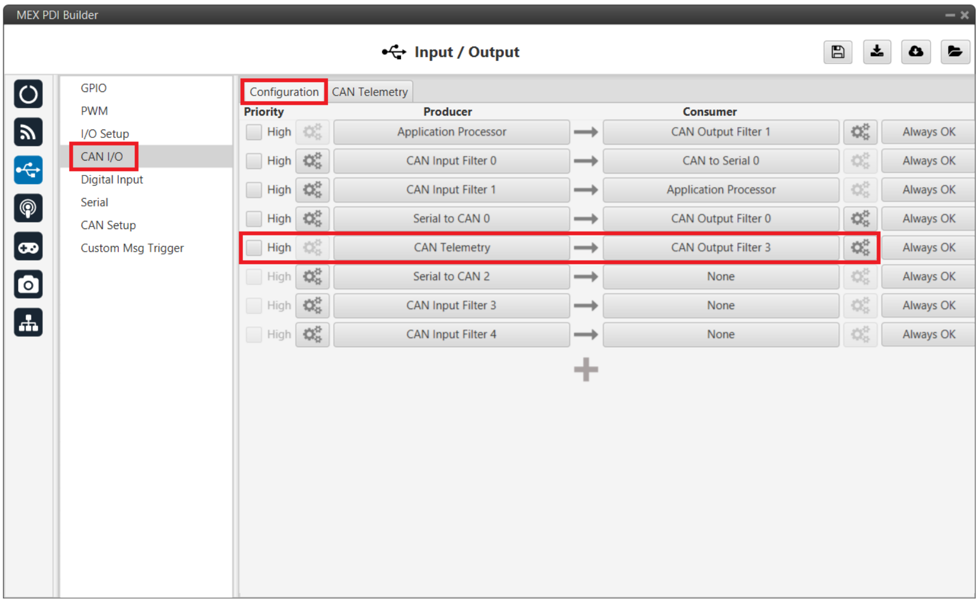


CAN messages transmission - Communication diagram MEX → 1x

Users must follow the steps below to send telemetry via CAN and configure an **Autopilot 1x** to read this telemetry.

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

   Connect **CAN Telemetry** to a **CAN Output Filter** as follows:

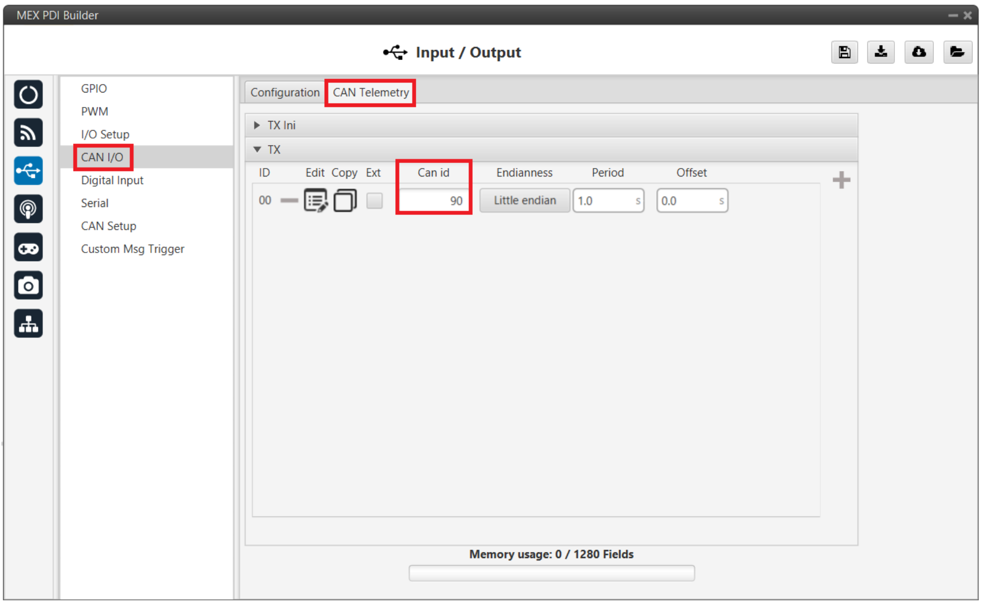


Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	Application Processor	Always OK
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	CAN Telemetry	→	CAN Output Filter 3	Always OK
□ High	Serial to CAN 2	→	None	Always OK
□ High	CAN Input Filter 3	→	None	Always OK
□ High	CAN Input Filter 4	→	None	Always OK

CAN messages transmission - CAN I/O configuration

2. Go to Input/Output menu → CAN I/O panel → **CAN Telemetry tab**.

   Select the fields to send in the **TX**. More information about the configuration of telemetry messages can be found in the CAN Telemetry - Input/Output section of this manual.

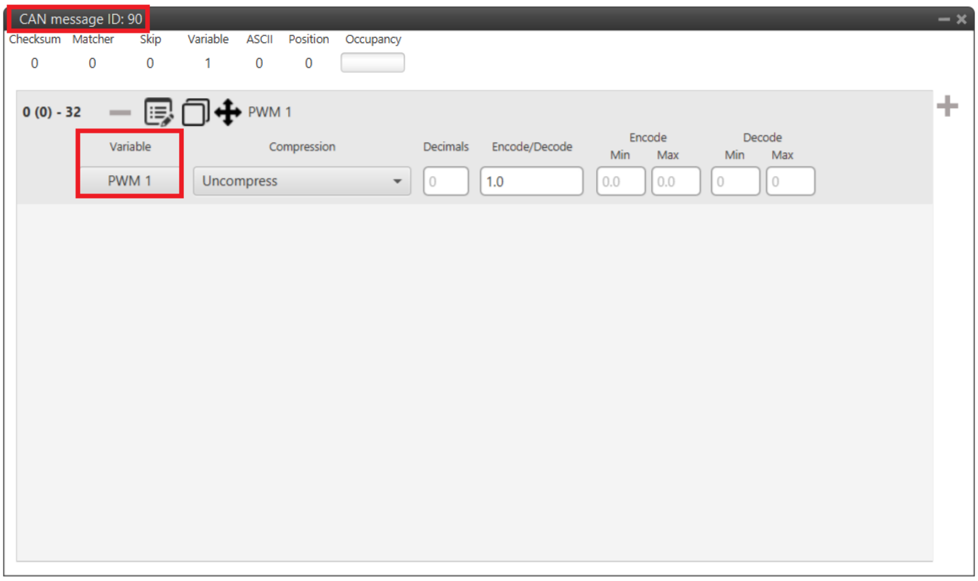


ID	Edit	Copy	Ext	Can id	Endianness	Period	Offset
00			□	90	Little endian	1.0 s	0.0 s

Memory usage: 0 / 1280 Fields

CAN messages transmission - CAN Telemetry

For example, a telemetry message with a variable set to ID 90:


CAN message ID: 90

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	0	0	1	0	0	

0 (0) - 32 PWM 1

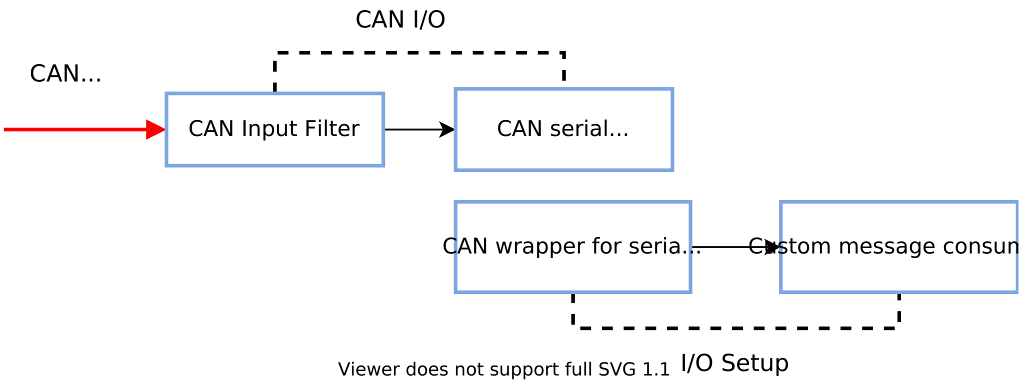
Variable	Compression	Decimals	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
PWM 1	Uncompress	0	1.0	0.0	0.0	0	0

CAN messages transmission - CAN Telemetry example

3. Configure the **Autopilot 1x** to read telemetry from **MEX**, following the CAN messages reception - Integration examples section of the **1x PDI Builder** user manual.

> ⓘ **Note**
>
> The configured CAN ID must match the one in **MEX**.

CAN messages reception

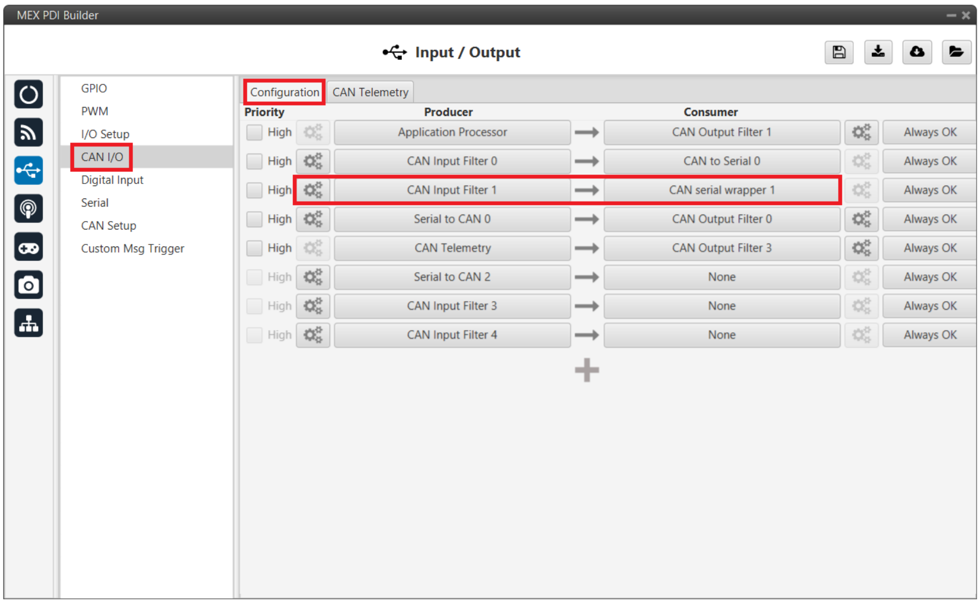


CAN messages reception - Communication diagram

Users must follow the steps below to read CAN messages.

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

   Connect a **CAN Input Filter** to a **CAN serial wrapper** as follows:



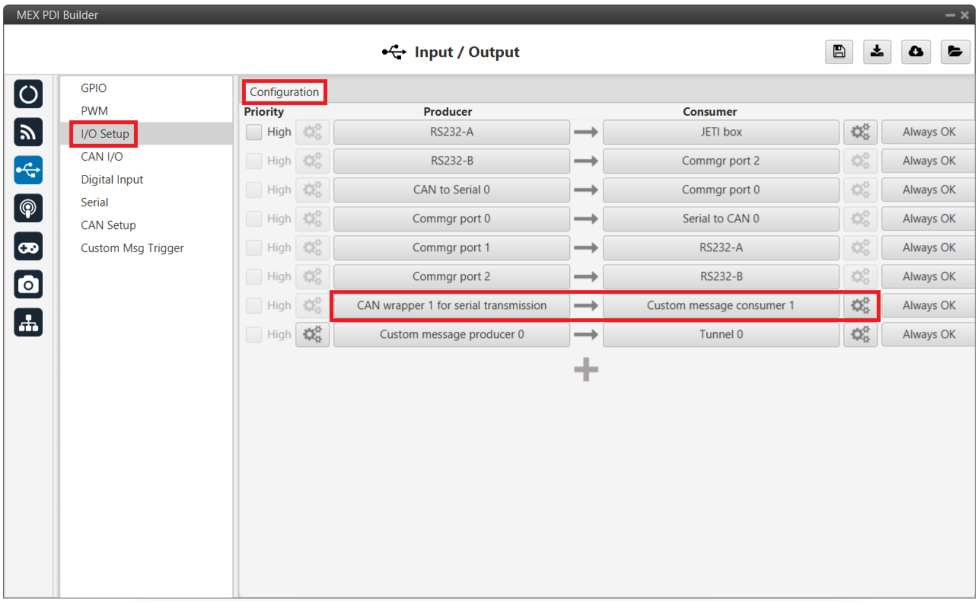
Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	CAN serial wrapper 1	Always OK
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	CAN Telemetry	→	CAN Output Filter 3	Always OK
□ High	Serial to CAN 2	→	None	Always OK
□ High	CAN Input Filter 3	→	None	Always OK
□ High	CAN Input Filter 4	→	None	Always OK

CAN messages reception - CAN I/O configuration

2. Click on the ⚙ button of the **CAN Input Filter** and configure the Port and CAN Id, according to the messages to receive.
3. Go to Input/Output menu → I/O Setup panel → **Configuration tab**.

   Connect the corresponding **CAN wrapper for serial transmission** to a **Custom message consumer**.

> **Note**
>
> The **CAN wrapper for serial transmission** index must match the index of the **CAN serial wrapper** configured in the previous step (in this example, CAN wrapper for serial transmission with **index 1** is used).



Priority	Producer		Consumer	
□ High	RS232-A	→	JETI box	Always OK
□ High	RS232-B	→	Commgr port 2	Always OK
□ High	CAN to Serial 0	→	Commgr port 0	Always OK
□ High	Commgr port 0	→	Serial to CAN 0	Always OK
□ High	Commgr port 1	→	RS232-A	Always OK
□ High	Commgr port 2	→	RS232-B	Always OK
□ High	CAN wrapper 1 for serial transmission	→	Custom message consumer 1	Always OK
□ High	Custom message producer 0	→	Tunnel 0	Always OK

CAN messages reception - I/O Setup configuration

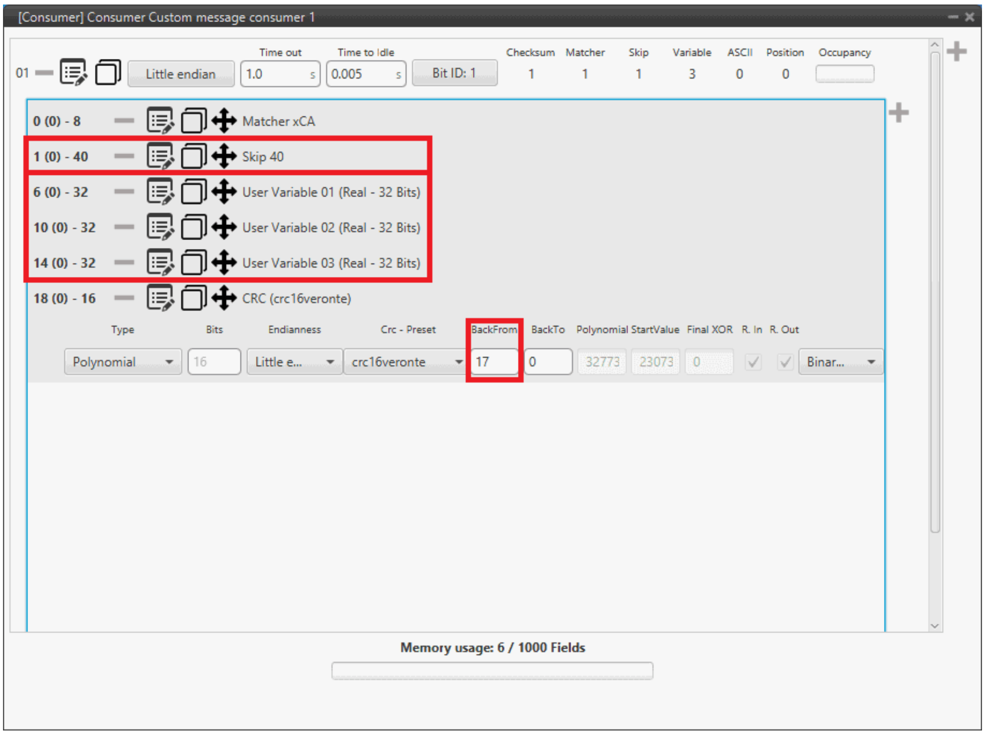
4. Click on the button of the **Custom message consumer** to access its configuration. To correctly read the CAN message, it must be setup as follows:
   - **Endianness**: Little endian
   - **Matcher**:
     - Value: 202
     - Bits: 8
   - **Skip 40**
   - **User variables to receive**.
   - **CRC**:
     - Type: Polynomial
     - Endianness: Little endian
     - Crc-Preset: crc16veronte
     - BackFrom: Number of bytes from which the CRC is computed, in this case, bytes comprehended between the Matcher and this CRC.
     - BackTo: 0

- Binary mode

If users wish more detailed information on this configuration, please refer to Custom message types - Input/Output section of the **1x PDI Builder** user manual.

The configuration of the **Custom message consumer** below is an example of **the reception of three Real variables**.

> **Note**
>
> The **CRC BackFrom entry** is 17 since there are 5 bytes in the **Skip 40** and 12 bytes for the User variables.



[Consumer] Consumer Custom message consumer 1

			Time out	Time to Idle		Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01		Little endian	1.0 s	0.005 s	Bit ID: 1	1	1	1	3	0	0	

- 0 (0) - 8 Matcher xCA
- 1 (0) - 40 Skip 40
- 6 (0) - 32 User Variable 01 (Real - 32 Bits)
- 10 (0) - 32 User Variable 02 (Real - 32 Bits)
- 14 (0) - 32 User Variable 03 (Real - 32 Bits)
- 18 (0) - 16 CRC (crc16veronte)

Type	Bits	Endianness	Crc - Preset	BackFrom	BackTo	Polynomial	StartValue	Final XOR	R. In	R. Out	
Polynomial	16	Little e...	crc16veronte	17	0	32773	23073	0	✓	✓	Binar...

Memory usage: 6 / 1000 Fields

CAN messages reception - Custom message configuration example

CAN Isolator

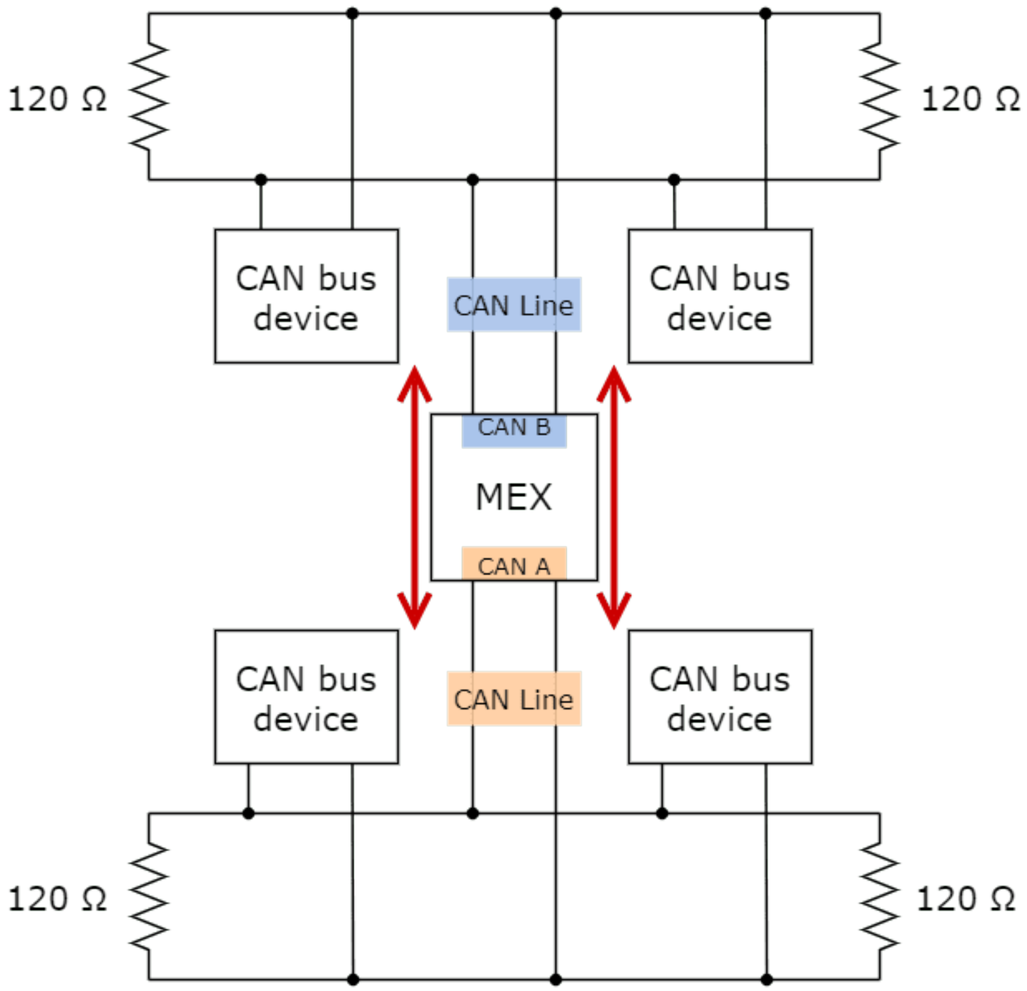
MEX can operate as a CAN bus isolator, since it manages both CAN buses as desired. The system has a built-in microcontroller; which manages CAN buses in real-time. Both buses are not electrically connected, in consequence, electrical signals that do not follow the CAN protocol will be isolated.

The functionalities of a CAN isolator are the following:

- CAN Input Filter

- Filtered CAN subnet
- CAN tunnel

The following diagram summarizes the behavior of MEX as a CAN bus isolator:

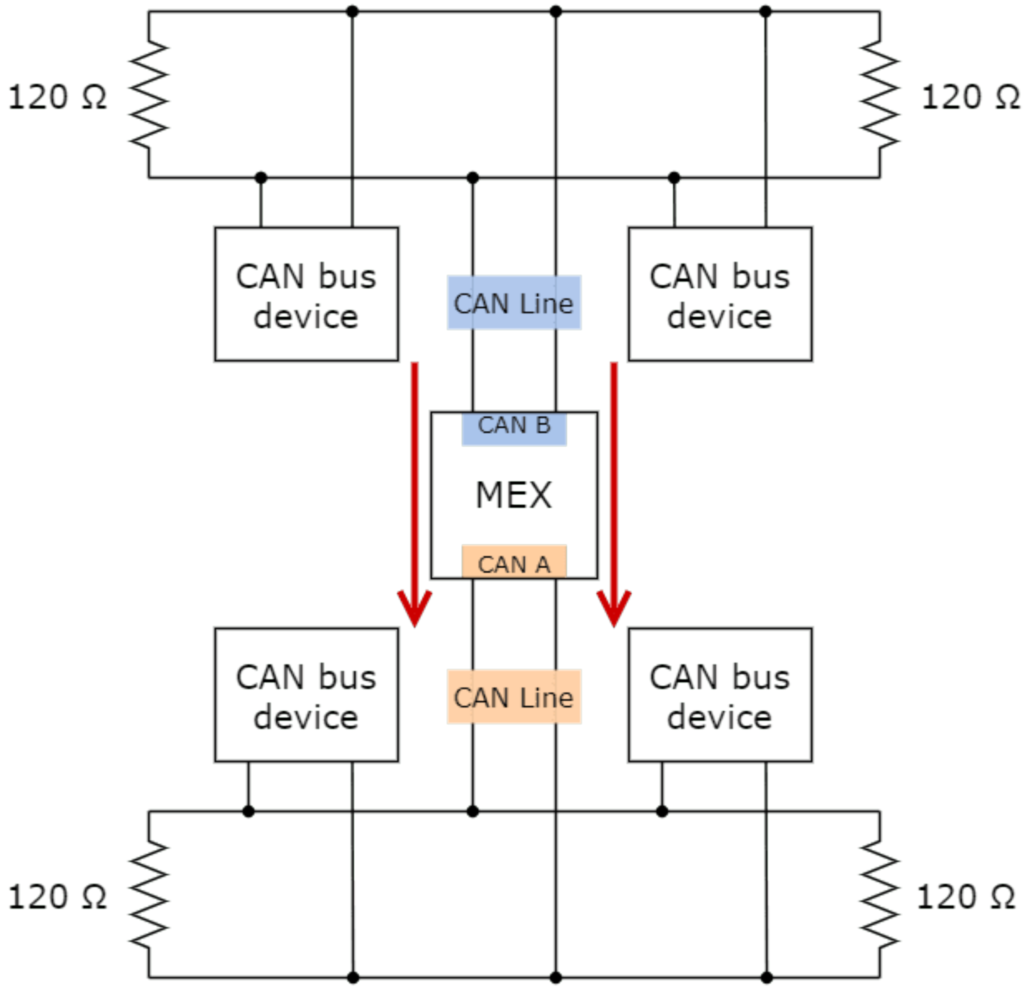


Subnets CAN diagram

Filtered CAN subnet

With **MEX** it is possible to isolate CAN nets, by filtering certain messages from one CAN line and only transmitting specific information through **MEX** to the other one.

In this example, only a certain range of CAN IDs will be allowed to cross from one CAN line to the other using the two CAN ports of **MEX**. Specifically, information will pass **from CAN B** of MEX **to CAN A** of MEX:

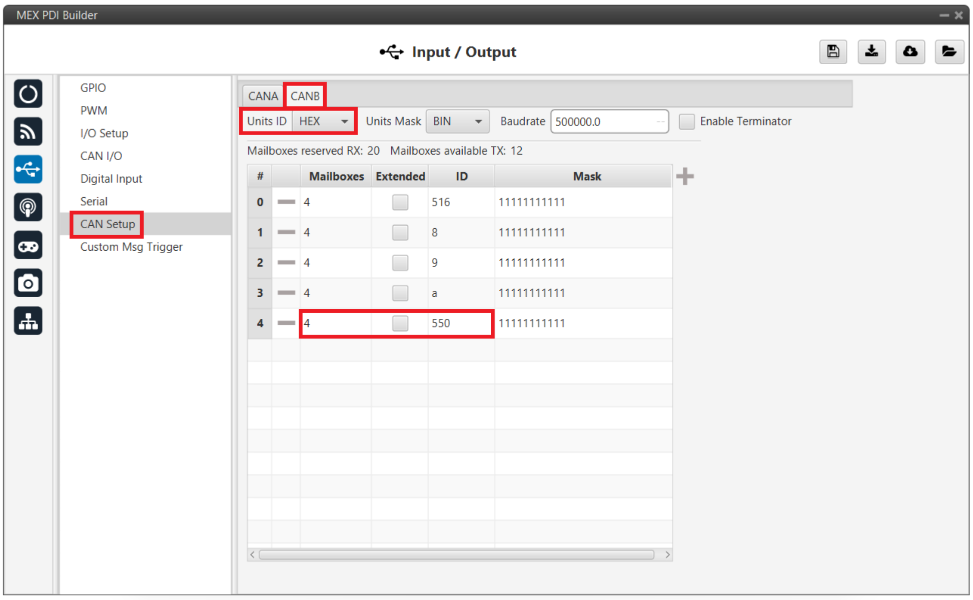


Subnets CAN example - From CAN B to CAN A

The allowed range will be from **0x550** to **0x55F**.

1. Create a new mailbox entry for **CAN B**.

   To do this, go to Input/Output menu → CAN Setup panel → **CAN B tab**. Assign some of the mailboxes to it and set the ID to **0x550**. This example uses an ID expressed in **hexadecimal** notation.



Units ID HEX | Units Mask BIN | Baudrate 500000.0 | □ Enable Terminator

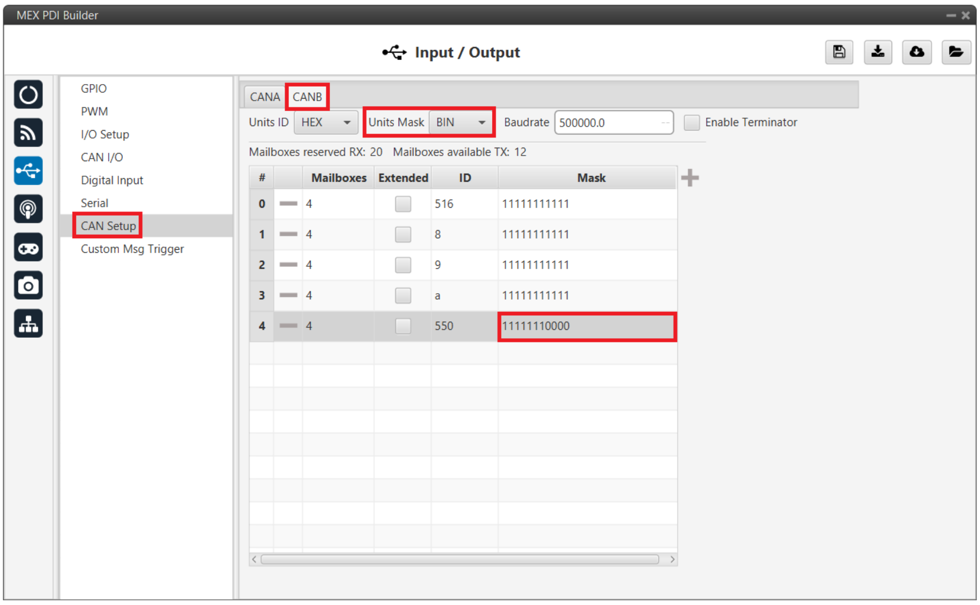
Mailboxes reserved RX: 20 Mailboxes available TX: 12

#		Mailboxes	Extended	ID	Mask
0	—	4	□	516	11111111111
1	—	4	□	8	11111111111
2	—	4	□	9	11111111111
3	—	4	□	a	11111111111
4	—	4	□	550	11111111111

Filtered CAN subnet - CAN Setup configuration

2. Set a Mask which will ignore the last 4 bits.

> **Note**
>
> Although the ID has been entered in hexadecimal notation, note that the Mask has been set in **binary** format.



Units ID HEX | Units Mask BIN | Baudrate 500000.0 | □ Enable Terminator

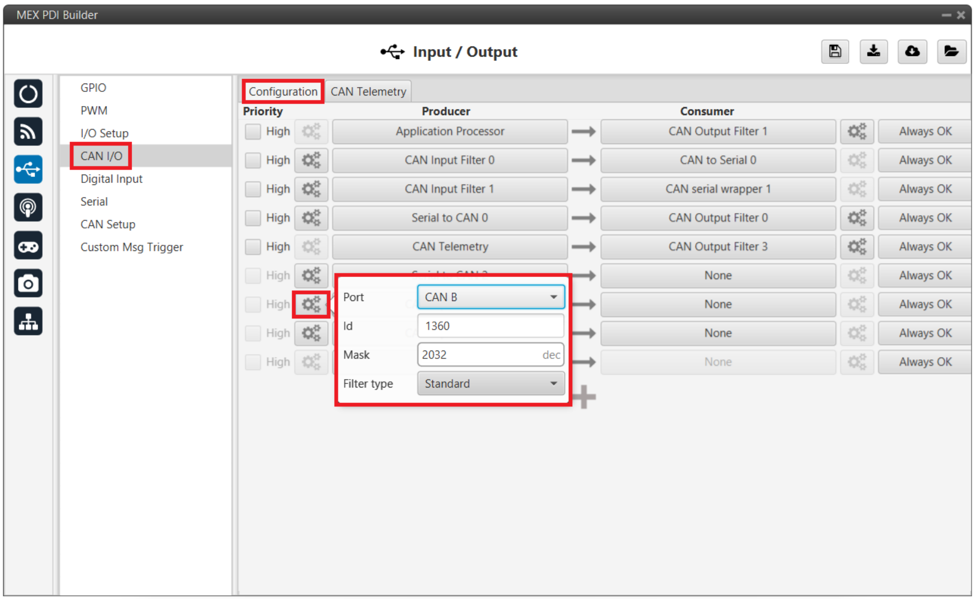
Mailboxes reserved RX: 20 Mailboxes available TX: 12

#		Mailboxes	Extended	ID	Mask
0	—	4	□	516	11111111111
1	—	4	□	8	11111111111
2	—	4	□	9	11111111111
3	—	4	□	a	11111111111
4	—	4	□	550	11111110000

Filtered CAN subnet - CAN Setup mask configuration

3. Go to Input/Output menu → CAN I/O panel → **Configuration tab**. Configure **CAN Input Filter 3** on **CAN B**, with the same settings as the **Mailbox**.

> **Important**
>
> This menu only allows setting the Id with **decimal numbers**. For this example the Id 0x550 is represented as **1360** in decimal format, and the Mask is also represented in decimal format as **2032**.

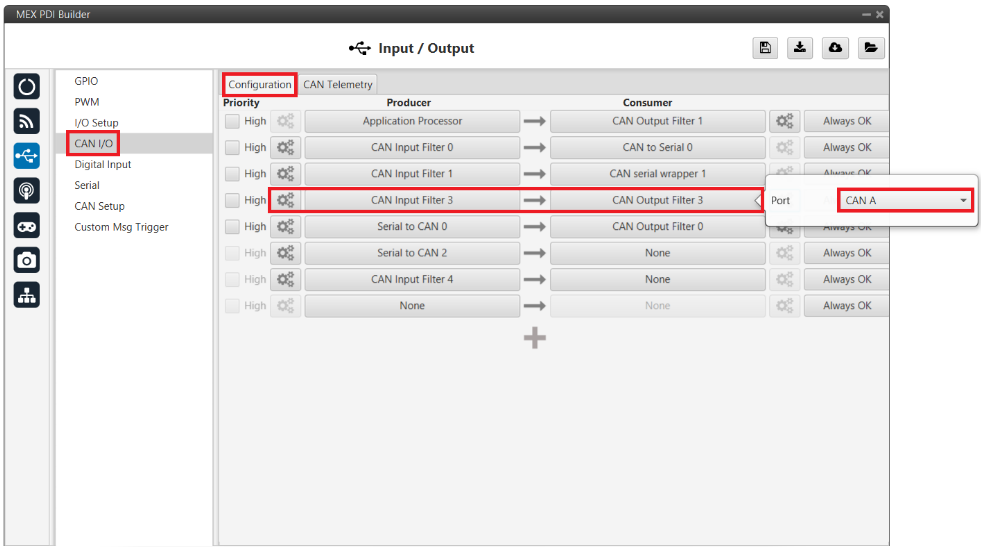


Priority	Producer		Consumer	
High	Application Processor	→	CAN Output Filter 1	Always OK
High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
High	CAN Input Filter 1	→	CAN serial wrapper 1	Always OK
High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
High	CAN Telemetry	→	CAN Output Filter 3	Always OK
High		→	None	Always OK
High		→	None	Always OK
High		→	None	Always OK
High		→	None	Always OK

Port: CAN B
Id: 1360
Mask: 2032 dec
Filter type: Standard

Filtered CAN subnet - CAN Input Filter configuration

4. Bind **CAN Output Filter 3** to **CAN Input Filter 3** and, configure the CAN Output Filter to **CAN A**.



Priority	Producer		Consumer	
High	Application Processor	→	CAN Output Filter 1	Always OK
High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
High	CAN Input Filter 1	→	CAN serial wrapper 1	Always OK
High	CAN Input Filter 3	→	CAN Output Filter 3	
High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
High	Serial to CAN 2	→	None	Always OK
High	CAN Input Filter 4	→	None	Always OK
High	None	→	None	Always OK

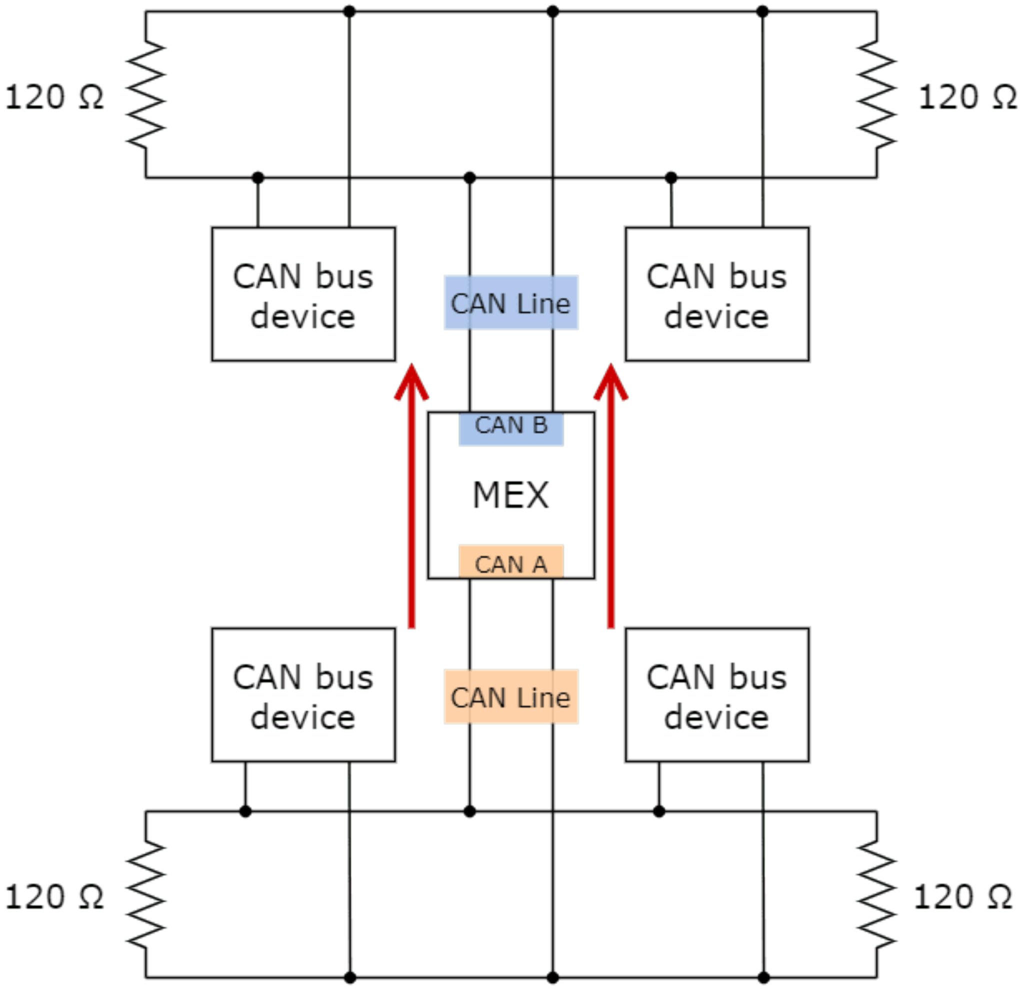
Port: CAN A

Filtered CAN subnet - CAN Output Filter configuration

CAN tunnel

A CAN tunnel is a specific case of a message tunnel; where messages are tunneled from one CAN port of **MEX** to the other.

In this example, a transparent tunnel will be created. So, any messages received on **Interface A** will be sent through **Interface B**:



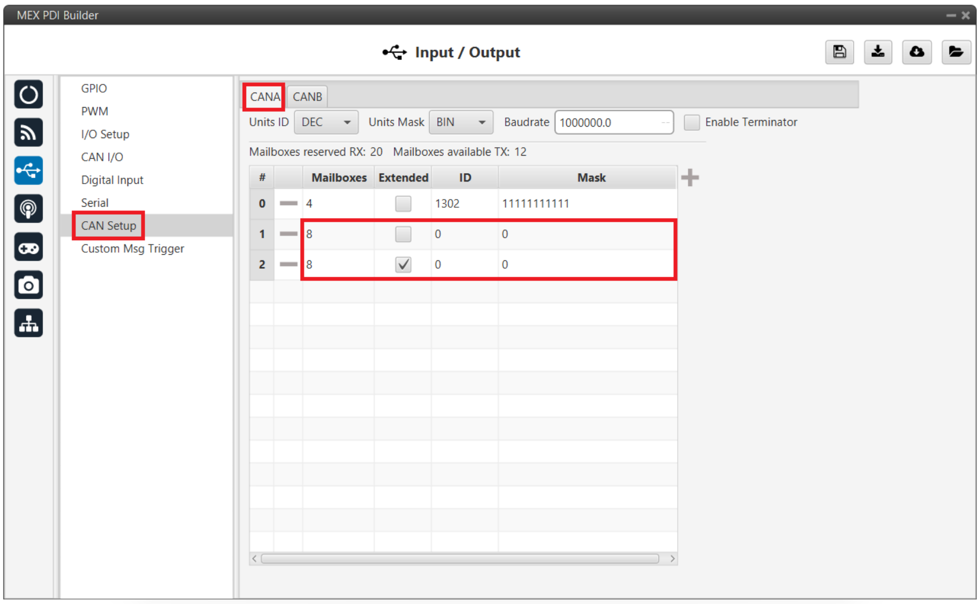
Subnets CAN example - From CAN A to CAN B

Optionally, the mailboxes can be equally distributed to support both standard and extended CAN IDs.

1. Create a new mailbox entry for **Interface A**.

   To do this, go to Input/Output menu → CAN Setup panel → **CAN A tab**.

   Assign half of the mailboxes to it and set a **Mask** of 0.



Units ID DEC Units Mask BIN Baudrate 1000000.0 ☐ Enable Terminator

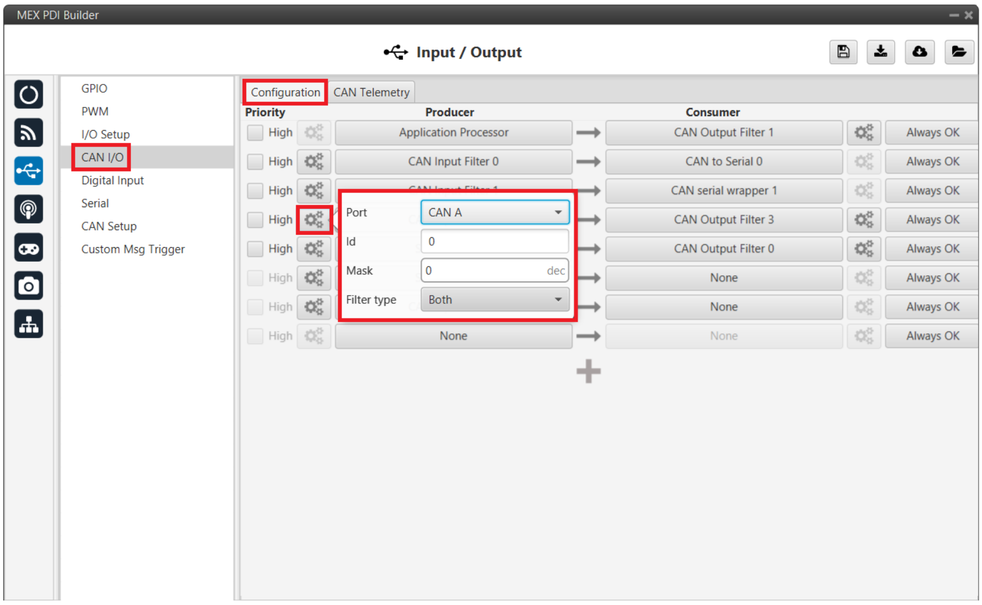
Mailboxes reserved RX: 20 Mailboxes available TX: 12

#	Mailboxes	Extended	ID	Mask
0	4	☐	1302	11111111111
1	8	☐	0	0
2	8	☑	0	0

CAN tunnel - CAN Setup configuration

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

   Configure **CAN Input Filter 3** on **CAN A**, with CAN **id 0**, a **Mask** of **0** and **Both** as frame format type.



Priority	Producer		Consumer	
☐ High	Application Processor	→	CAN Output Filter 1	Always OK
☐ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
☐ High	CAN Input Filter 1	→	CAN serial wrapper 1	Always OK
☐ High		→	CAN Output Filter 3	Always OK
☐ High		→	CAN Output Filter 0	Always OK
☐ High		→	None	Always OK
☐ High		→	None	Always OK
☐ High	None	→	None	Always OK

Port: CAN A

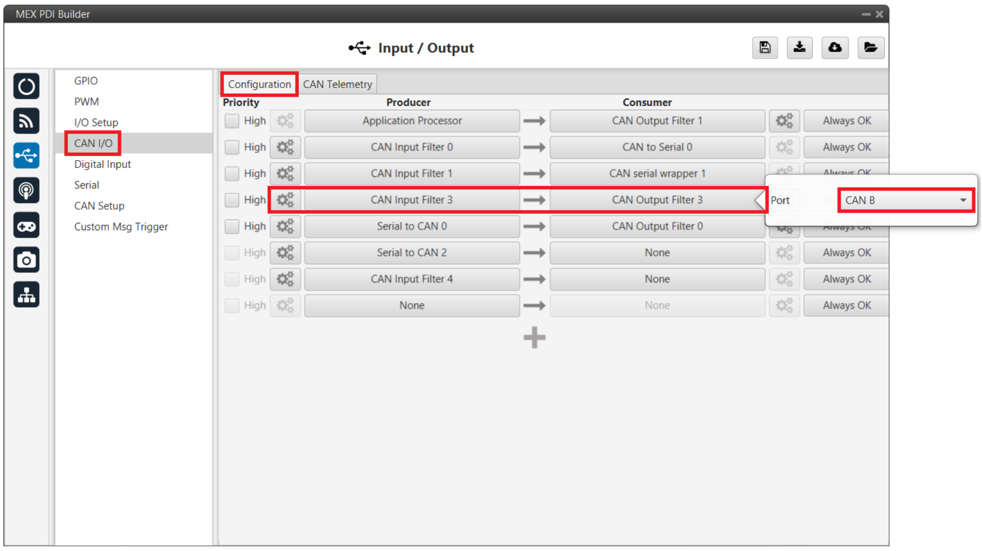
Id: 0

Mask: 0 dec

Filter type: Both

CAN tunnel - CAN Input Filter configuration

3. Bind **CAN Output Filter 3** to **CAN Input Filter 3** and, configure the CAN Output Filter to **CAN B**.



MEX PDI Builder

Input / Output

GPIO
PWM
I/O Setup
CAN I/O
Digital Input
Serial
CAN Setup
Custom Msg Trigger

Configuration | CAN Telemetry

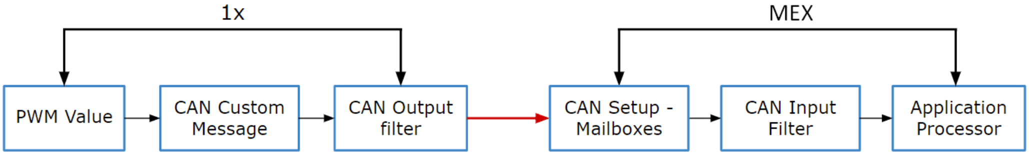
Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	CAN serial wrapper 1	Always OK
□ High	CAN Input Filter 3	→	CAN Output Filter 3	
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	Serial to CAN 2	→	None	Always OK
□ High	CAN Input Filter 4	→	None	Always OK
□ High	None	→	None	Always OK

Port: CAN B

CAN tunnel - CAN Output Filter configuration

Commanding/Reading PWMs

The appropriate PWM message format is described in Command PWMs - CAN Bus protocol section of the **MEX Software Manual**.

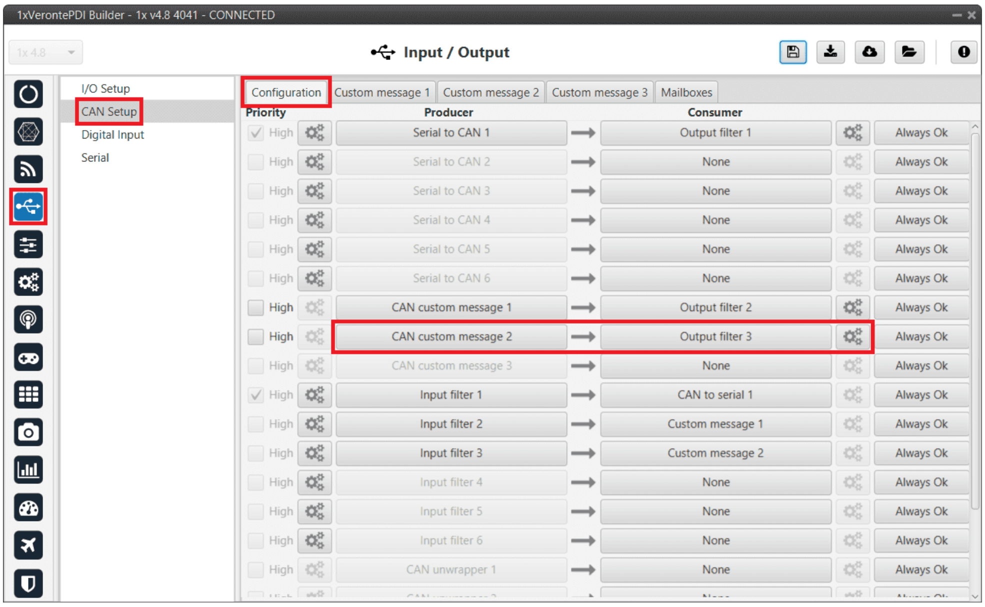


PWM Command - Communication diagram 1x → MEX

The following steps command PWM from **Autopilot 1x** to **MEX**:

1x PDI Builder side

1. Go to Input/Output menu → CAN Setup panel → **Configuration tab**. Connect a **CAN custom message** to an **Output filter** to send the message:

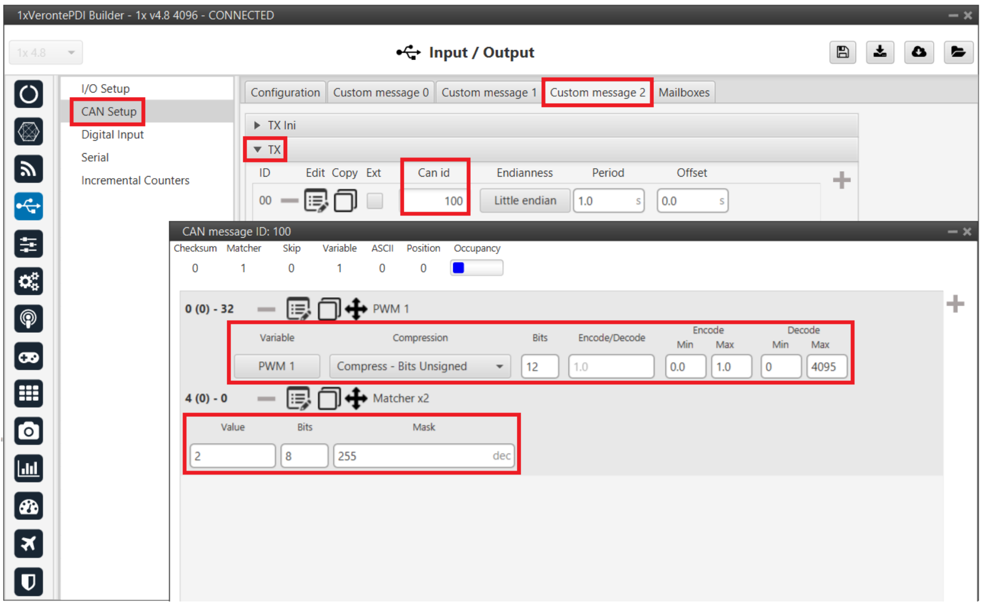


Priority	Producer		Consumer	
☑ High	Serial to CAN 1	→	Output filter 1	Always Ok
□ High	Serial to CAN 2	→	None	Always Ok
□ High	Serial to CAN 3	→	None	Always Ok
□ High	Serial to CAN 4	→	None	Always Ok
□ High	Serial to CAN 5	→	None	Always Ok
□ High	Serial to CAN 6	→	None	Always Ok
□ High	CAN custom message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

1x PDI Builder - CAN Setup configuration

2. Go to Input/Output menu → CAN Setup panel → **Custom Message 2 tab** (because Custom message **2** has been selected as producer).

   Build a CAN Custom message using the PWM variable and the appropriate message format. Select the fields to send in the **TX** as it is a Producer. CAN ID is arbitrary, for this example ID 100 will be used:



ID	Edit	Copy	Ext	Can id	Endianness	Period	Offset
00			□	100	Little endian	1.0 s	0.0 s

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	1	0	1	0	0	

0 (0) - 32 PWM 1

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
PWM 1	Compress - Bits Unsigned	12	1.0	0.0	1.0	0	4095

4 (0) - 0 Matcher x2

Value	Bits	Mask
2	8	255 dec

1x PDI Builder - CAN Custom message configuration

Message format:

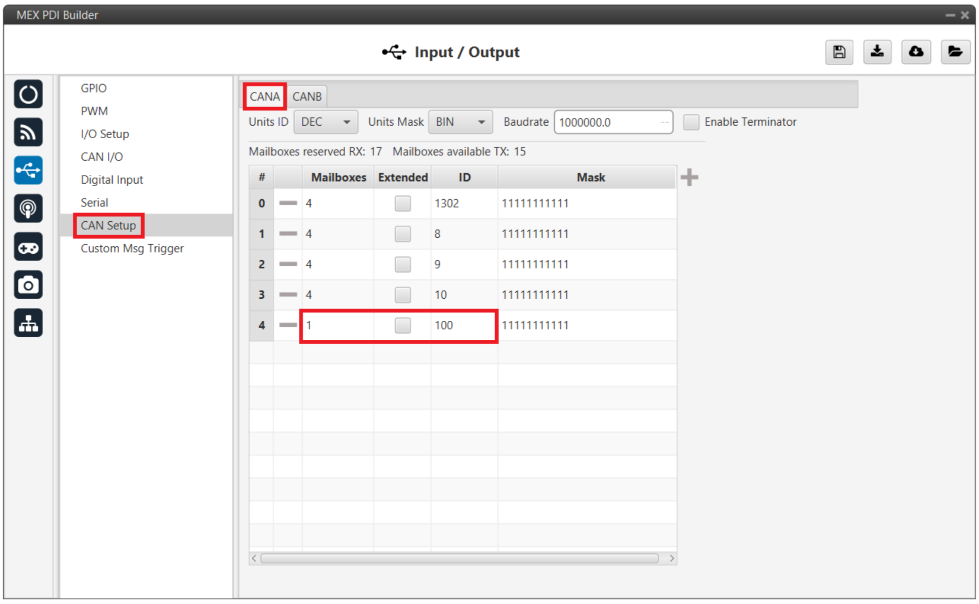
- Matcher (2) [8 bits]
  - PWM 0 [12 bits]
  - PWM 1 [12 bits]
  - PWM 2 [12 bits]

    - PWM 3 [12 bits]
  - Matcher (3) [8 bits]
    - PWM 4 [12 bits]
    - PWM 5 [12 bits]
    - PWM 6 [12 bits]
    - PWM 7 [12 bits]
  - Each PWM variable has to be set using the following format:
    - Variable: PWM X
    - **Compression**: Compress - Bits Unsigned
    - **Bits**: 12
    - **Encode**: Min=0.0 / Max=1.0
    - **Decode**: Min=0 / Max=4095

MEX PDI Builder side

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

   Create the mailbox to receive the new message (ID 100):



Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

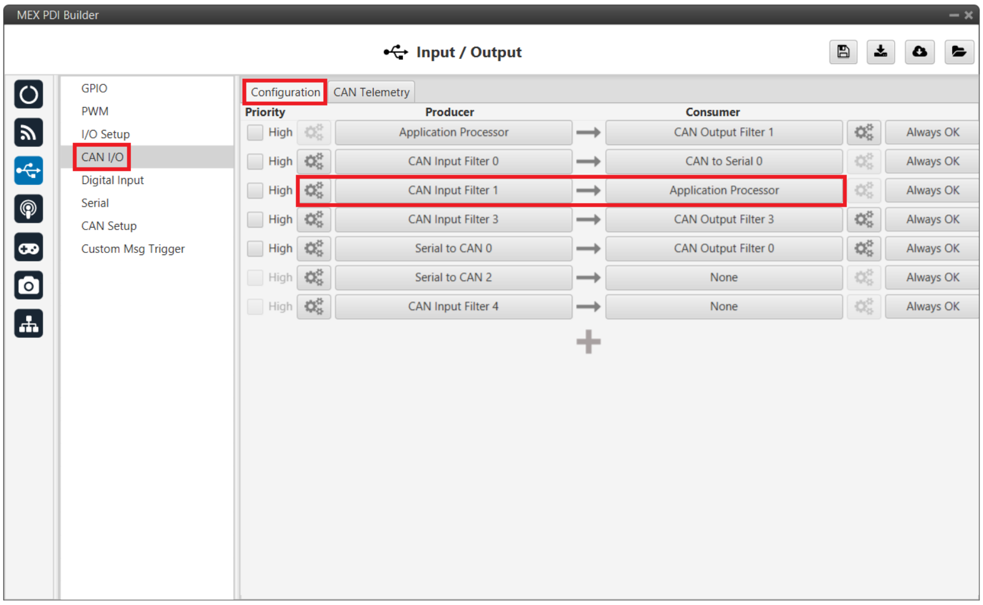
Mailboxes reserved RX: 17 Mailboxes available TX: 15

#		Mailboxes	Extended	ID	Mask
0	—	4	□	1302	11111111111
1	—	4	□	8	11111111111
2	—	4	□	9	11111111111
3	—	4	□	10	11111111111
4	—	1	□	100	11111111111

MEX PDI Builder - CAN Setup configuration

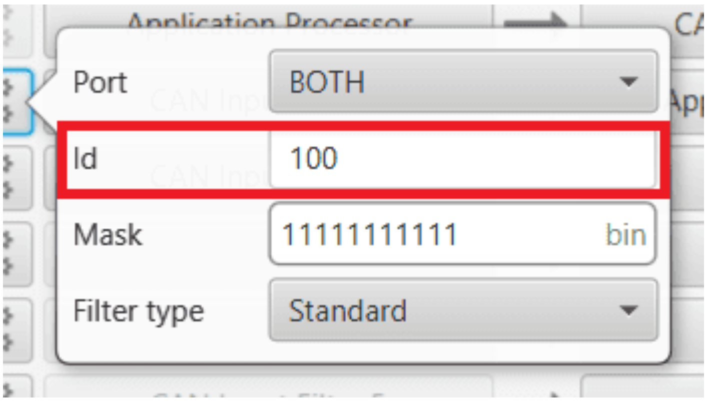
2. Go to Input/Output menu → CAN I/O panel → **Configuration tab**.

   Connect a **CAN Input Filter** with the **right CAN ID** to the **Application Processor consumer**:



Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	Application Processor	Always OK
□ High	CAN Input Filter 3	→	CAN Output Filter 3	Always OK
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	Serial to CAN 2	→	None	Always OK
□ High	CAN Input Filter 4	→	None	Always OK

MEX PDI Builder - CAN I/O configuration



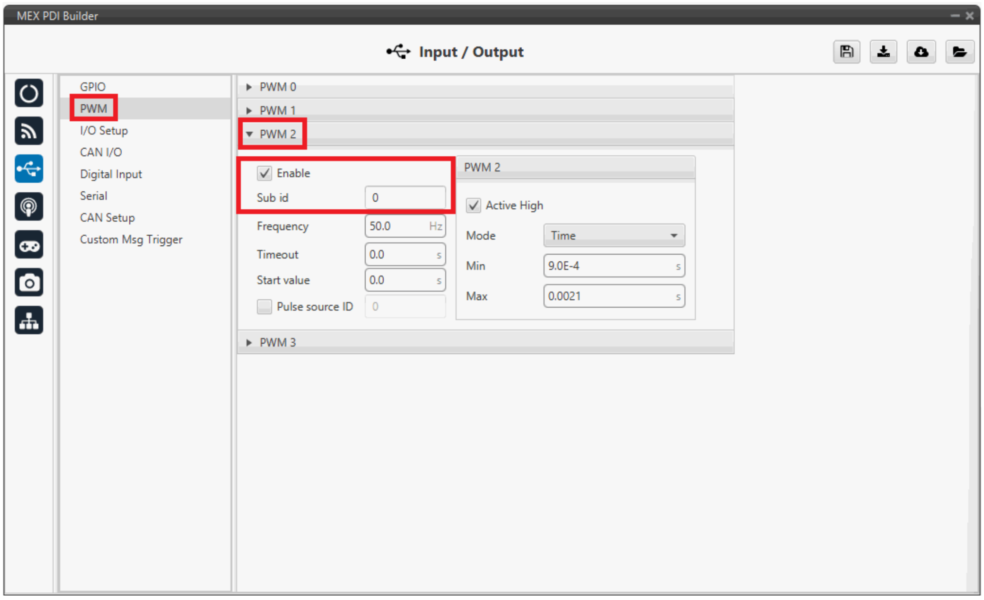
Port	BOTH
Id	100
Mask	11111111111 bin
Filter type	Standard

MEX PDI Builder - CAN Input Filter configuration

3. Go to Input/Output menu → PWM panel → **PWM 2 tab** (as PWM **1** has been selected in **1x PDI Builder**).

> **Note**
>
> Check the order of the PWM signal, the first PWM signal of Veronte Autopilot 1x must match the first PWM signal of MEX. **Numeration may differ** between Veronte applications depending on the used version.

Configure parameters for PWM 2:



PWM 2

☑ Enable

Sub id: 0

Frequency: 50.0 Hz

Timeout: 0.0 s

Start value: 0.0 s

□ Pulse source ID: 0

PWM 2

☑ Active High

Mode: Time

Min: 9.0E-4 s

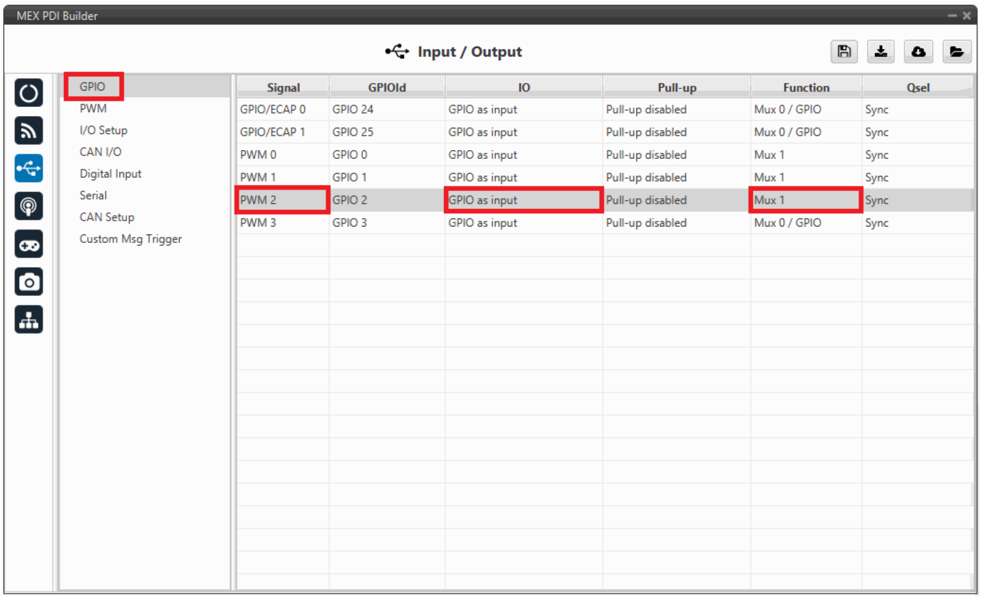
Max: 0.0021 s

MEX PDI Builder - PWM configuration

4. Go to Input/Output menu → **GPIO panel**.

   Check that this PWM pin (**PWM 2** in this case) has correctly switched from GPIO to PWM.

   It should look like this:



Signal	GPIOId	IO	Pull-up	Function	Qsel
GPIO/ECAP 0	GPIO 24	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
GPIO/ECAP 1	GPIO 25	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 0	GPIO 0	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 1	GPIO 1	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 2	GPIO 2	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 3	GPIO 3	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync

MEX PDI Builder - GPIO configuration

   - **IO**: GPIO as **output**.
   - **Function**: Mux **1**.

5. Finally, save changes.

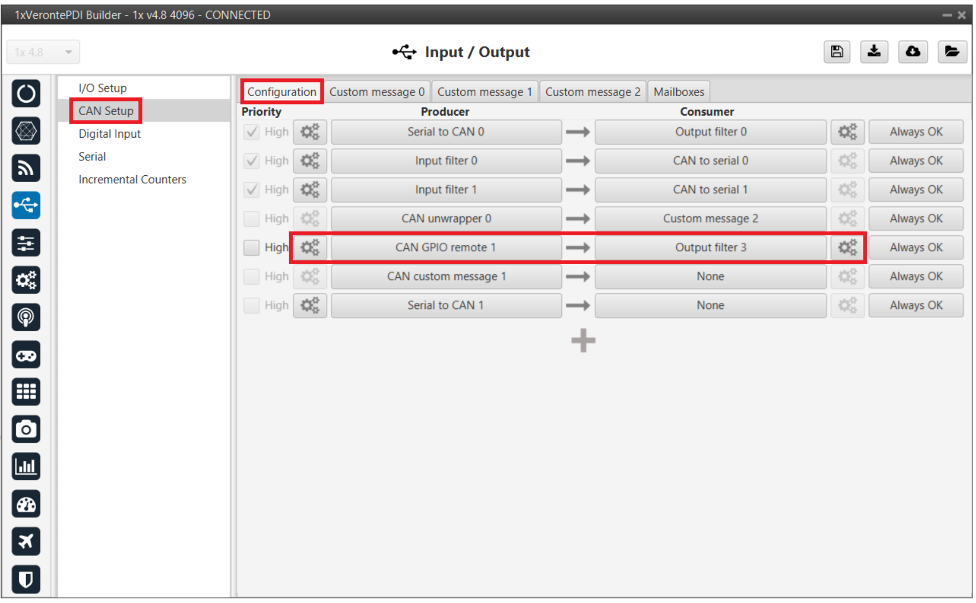
GPIO Command

The following are the steps to send a GPIO command from the **Veronte Autopilot 1x**, receive it at **MEX** and process it, so that **MEX** carries out the command.

GPIO Command is very similar to PWM command with a few differences.

1x PDI Builder side

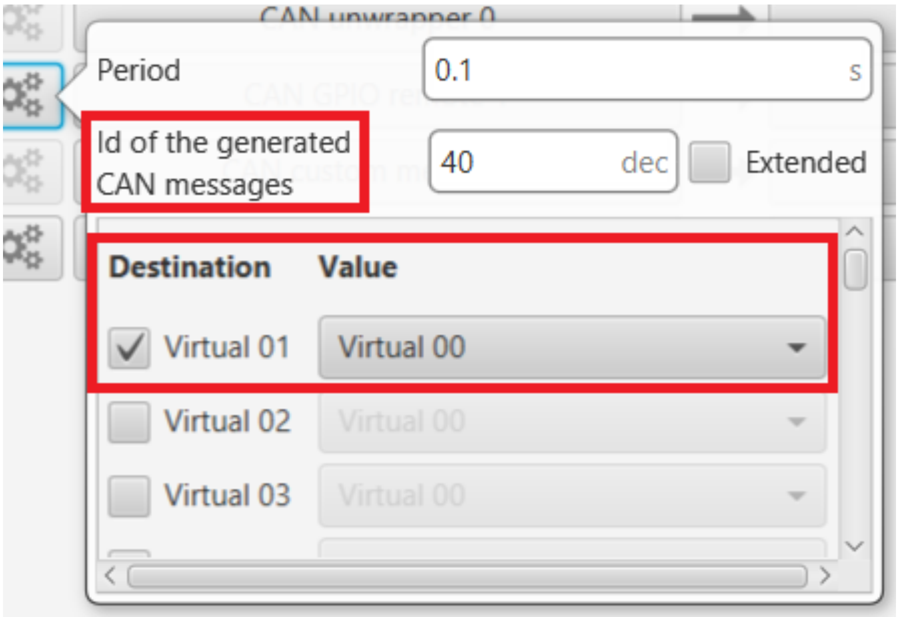
1. Go to Input/Output menu → CAN Setup panel → **Configuration tab**. Connect, as Producer, a '**CAN GPIO remote**' to an **Output filter**:



Priority	Producer		Consumer	
☑ High	Serial to CAN 0	→	Output filter 0	Always OK
☑ High	Input filter 0	→	CAN to serial 0	Always OK
☑ High	Input filter 1	→	CAN to serial 1	Always OK
□ High	CAN unwrapper 0	→	Custom message 2	Always OK
□ High	CAN GPIO remote 1	→	Output filter 3	Always OK
□ High	CAN custom message 1	→	None	Always OK
□ High	Serial to CAN 1	→	None	Always OK

1x PDI Builder - CAN Setup configuration

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



Period 0.1 s

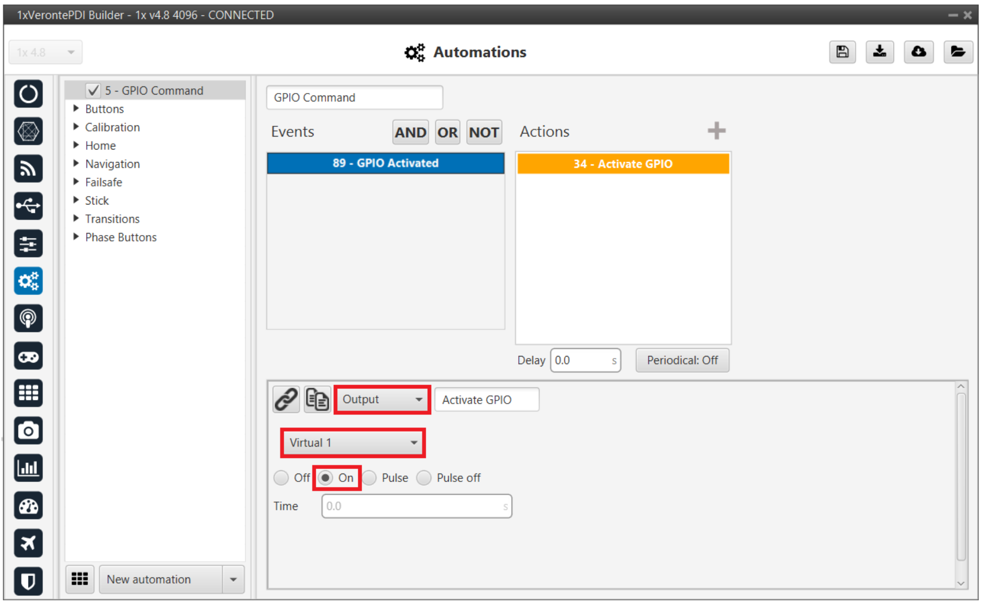
Id of the generated CAN messages 40 dec □ Extended

Destination	Value
☑ Virtual 01	Virtual 00
□ Virtual 02	Virtual 00
□ Virtual 03	Virtual 00

1x PDI Builder - CAN GPIO remote configuration

2. Go to **Automations menu**.

   GPIO must be activated using an '**Output' action**:



GPIO Command

89 - GPIO Activated

34 - Activate GPIO

Delay 0.0 s Periodical: Off

Output Activate GPIO

Virtual 1

○ Off ● On ○ Pulse ○ Pulse off

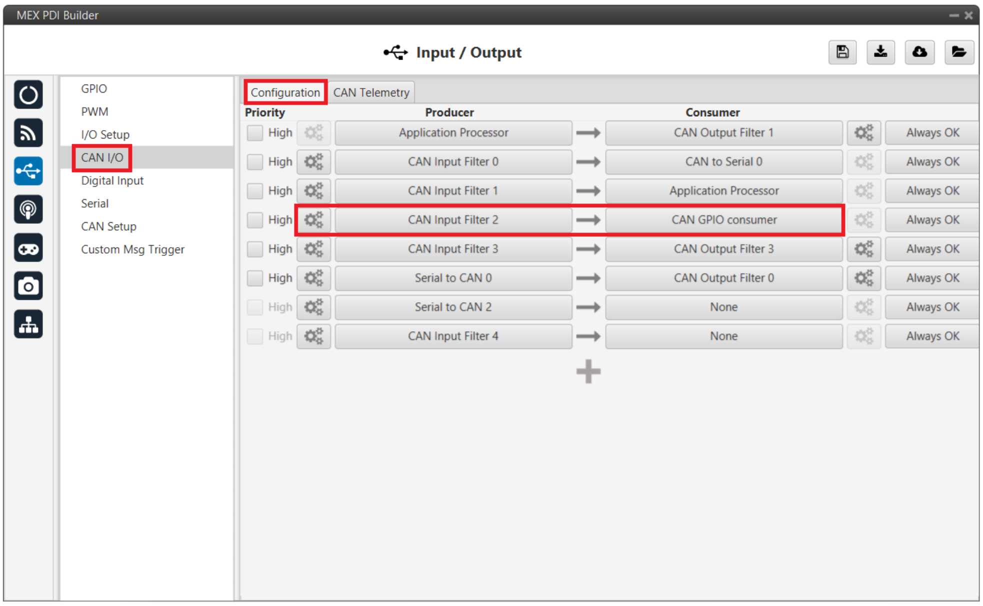
Time 0.0 s

1x PDI Builder - Automation configuration

MEX PDI Builder side

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

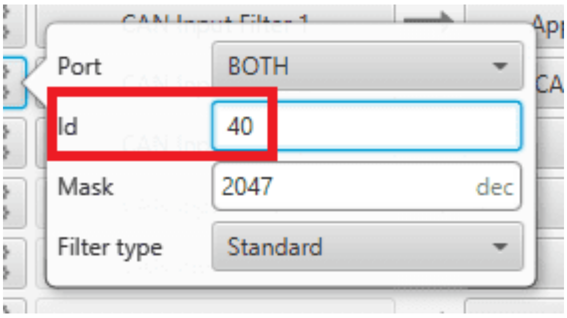
   Finally, connect a **CAN Input Filter** to the **CAN GPIO consumer**:



Priority	Producer		Consumer	
□ High	Application Processor	→	CAN Output Filter 1	Always OK
□ High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	→	Application Processor	Always OK
□ High	CAN Input Filter 2	→	CAN GPIO consumer	Always OK
□ High	CAN Input Filter 3	→	CAN Output Filter 3	Always OK
□ High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□ High	Serial to CAN 2	→	None	Always OK
□ High	CAN Input Filter 4	→	None	Always OK

MEX PDI Builder - CAN I/O configuration

The Id must match the one configured in **1x PDI Builder** as **Output filter**:



Port: BOTH
Id: 40
Mask: 2047 dec
Filter type: Standard

MEX PDI Builder - CAN Input Filter configuration

MEX as external magnetometer for Autopilot 1x

MEX has to be configured with **MEX PDI Builder** to send the magnetometer measurements by messages, in the same way as **external magnetometer** does. **Autopilot 1x** also requires a configuration, which is made with **1x PDI Builder**.

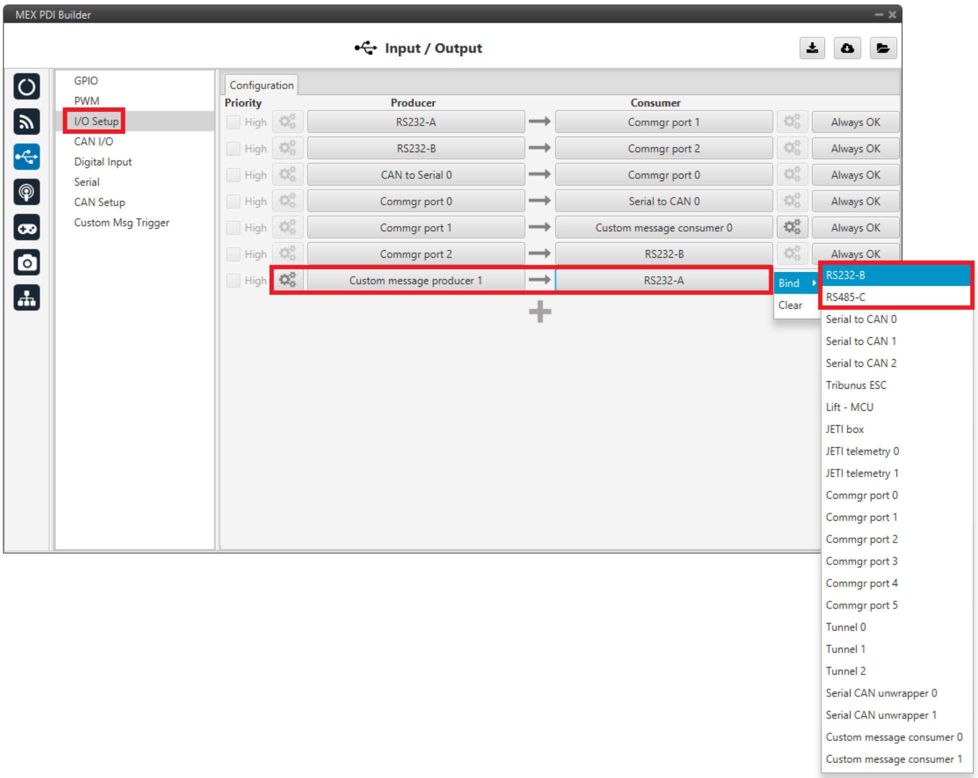
This configuration can be done both via **Serial** or **CAN** interfaces.

Serial

MEX PDI Builder side

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

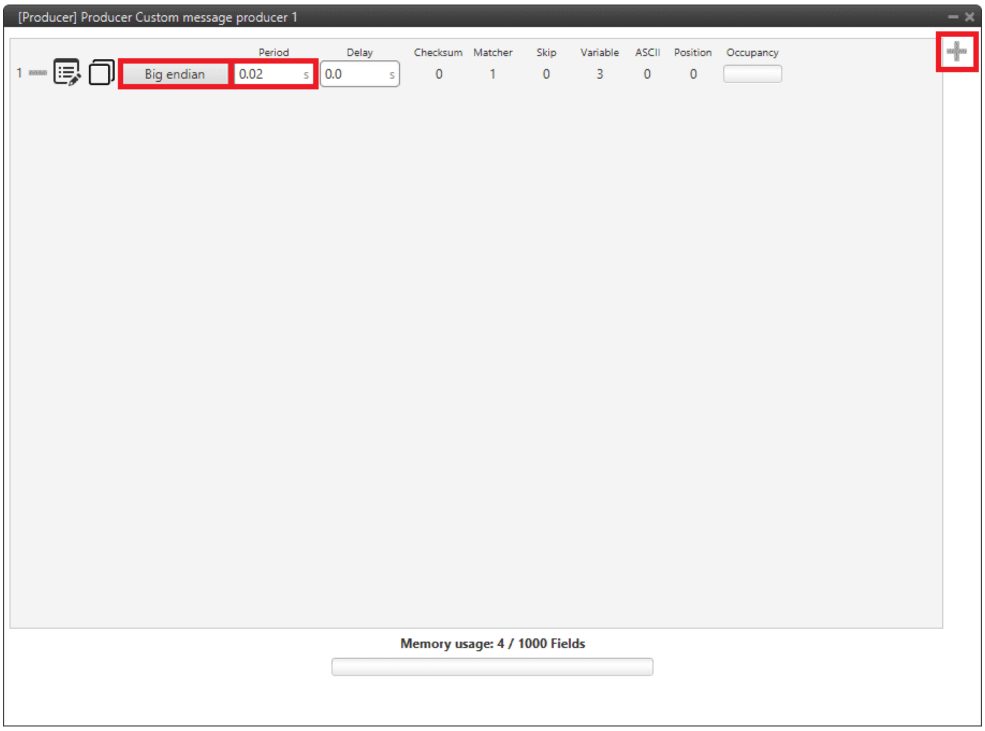
   Connect the **Custom message producer 1** or **2** to the **RS232-A**, **RS232-B** or **RS485-C** port.



Priority	Producer		Consumer	
□ High	RS232-A	→	Commgr port 1	Always OK
□ High	RS232-B	→	Commgr port 2	Always OK
□ High	CAN to Serial 0	→	Commgr port 0	Always OK
□ High	Commgr port 0	→	Serial to CAN 0	Always OK
□ High	Commgr port 1	→	Custom message consumer 0	Always OK
□ High	Commgr port 2	→	RS232-B	Always OK
□ High	Custom message producer 1	→	RS232-A	

MEX PDI Builder - I/O Setup configuration

2. Click on the button of the chosen **Custom message producer** and add a new message (clicking on), then configure it as **Big endian** and set a **Period of 0.02 s**.



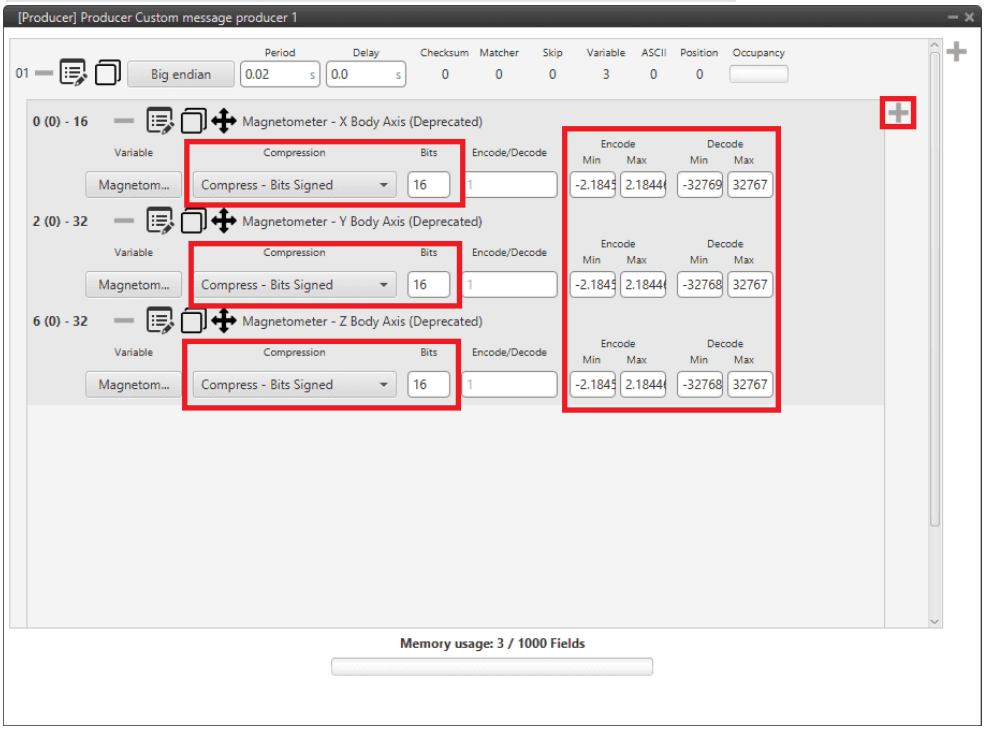
		Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
1	Big endian	0.02 s	0.0 s	0	1	0	3	0	0	

Memory usage: 4 / 1000 Fields

MEX PDI Builder - Custom message configuration

3. Click on ➕ to add the 3 following **Real variables**: **313**, **314** and **315**. Then configure them according to the following table:

Parameter	Configuration
Compression	Compress - Bits Signed
Bits	16
Encode	Min: -2.1845334E-4 / Max: 2.1844667E-4
Decode	Min: -32768 / Max: 32767



[Producer] Producer Custom message producer 1

	Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01 Big endian	0.02 s	0.0 s	0	0	0	3	0	0	

0 (0) - 16 Magnetometer - X Body Axis (Deprecated)

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32769	32767

2 (0) - 32 Magnetometer - Y Body Axis (Deprecated)

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32768	32767

6 (0) - 32 Magnetometer - Z Body Axis (Deprecated)

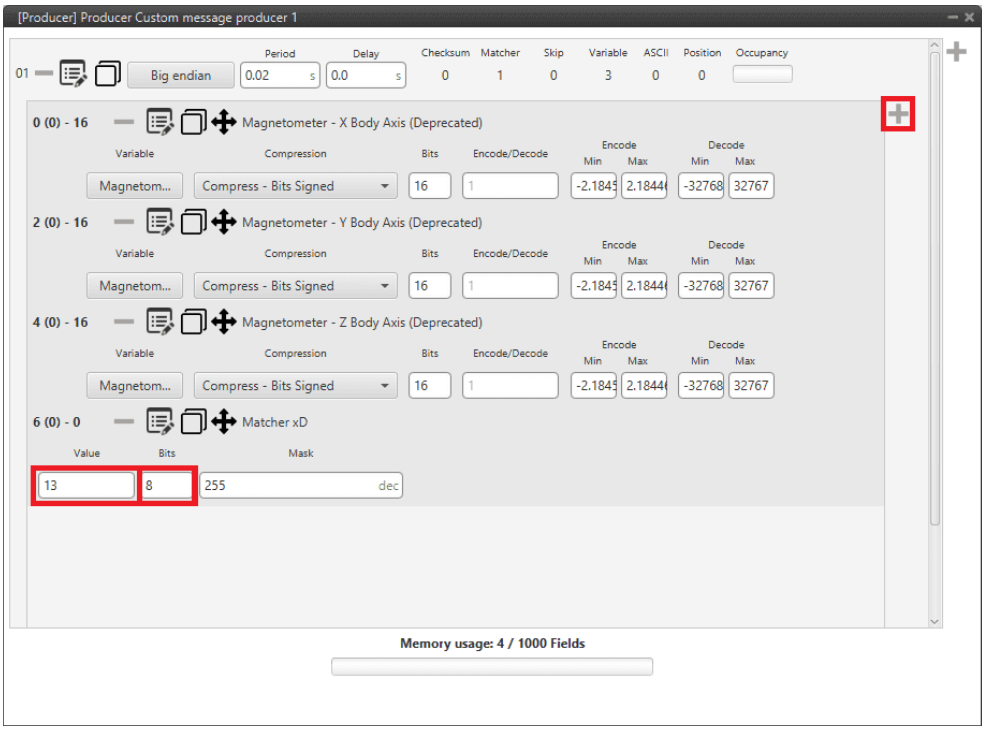
Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32768	32767

Memory usage: 3 / 1000 Fields

MEX PDI Builder - Custom message Variables configuration

4. Click on ➕→ **Matcher**, to add a matcher to the message and configure it as follows:

Parameter	Configuration
Value	13
Bits	8



[Producer] Producer Custom message producer 1

		Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Big endian	0.02 s	0.0 s	0	1	0	3	0	0	

0 (0) - 16 Magnetometer - X Body Axis (Deprecated)

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32768	32767

2 (0) - 16 Magnetometer - Y Body Axis (Deprecated)

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32768	32767

4 (0) - 16 Magnetometer - Z Body Axis (Deprecated)

Variable	Compression	Bits	Encode/Decode	Encode Min	Encode Max	Decode Min	Decode Max
Magnetom...	Compress - Bits Signed	16	1	-2.1845	2.1844	-32768	32767

6 (0) - 0 Matcher xD

Value	Bits	Mask
13	8	255 dec

Memory usage: 4 / 1000 Fields

MEX PDI Builder - Custom message Matcher configuration

> **Matcher Explanation**
>
> The matcher is a binary code employed to discard messages with errors. The length and the number represented can be edited clicking on ☰ of the matcher. If the receiver does not receive the exact same matcher sequence, the entire message will be discarded, since it will be considered corrupted.

> ⚠ **Warning**
>
> The order of variables and matcher must be the same.

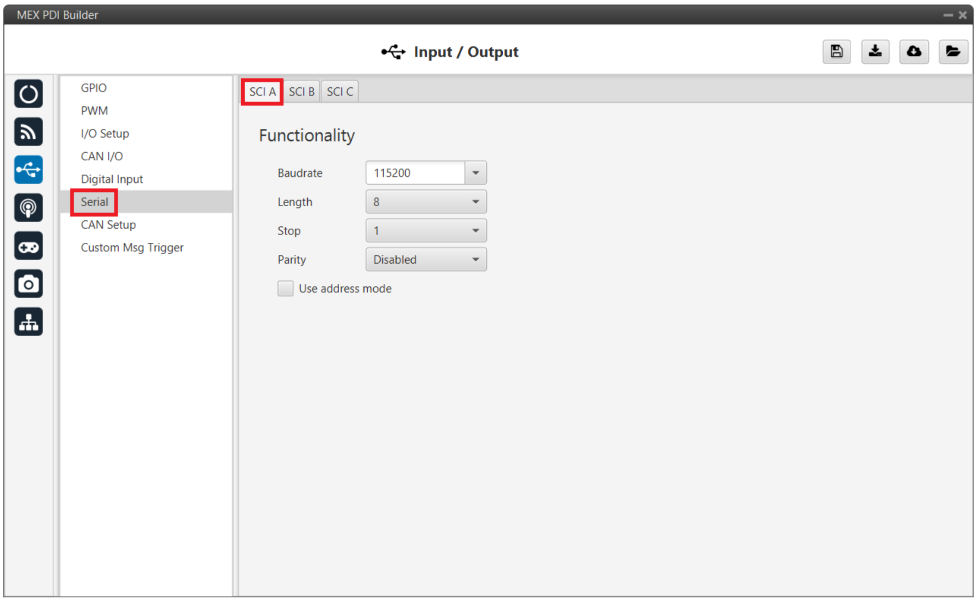
For more information about custom messages and matchers, read Custom Messages types - Input/Output section of the **1x PDI Builder** user manual.

5. Go to Input/Output menu → **Serial panel**.

   Configure the **SCI A, B or C** tab accordingly to the consumer selected in step 1 (**RS232-A**, **RS232-B** or **RS485-C**). The configuration is the following:

> ⓘ **Note**
>
> In this example, **RS232-A** was selected, so **SCI A** is configured.

Parameter	Configuration
Baudrate	115200
Length	8
Stop	1
Parity	Disabled
Use address mode	Disabled



Functionality

Baudrate 115200

Length 8

Stop 1

Parity Disabled

□ Use address mode

MEX PDI Builder - Serial configuration

6. Once finished the configuration, click on 💾 to write the configuration in **MEX**.

1x PDI Builder side

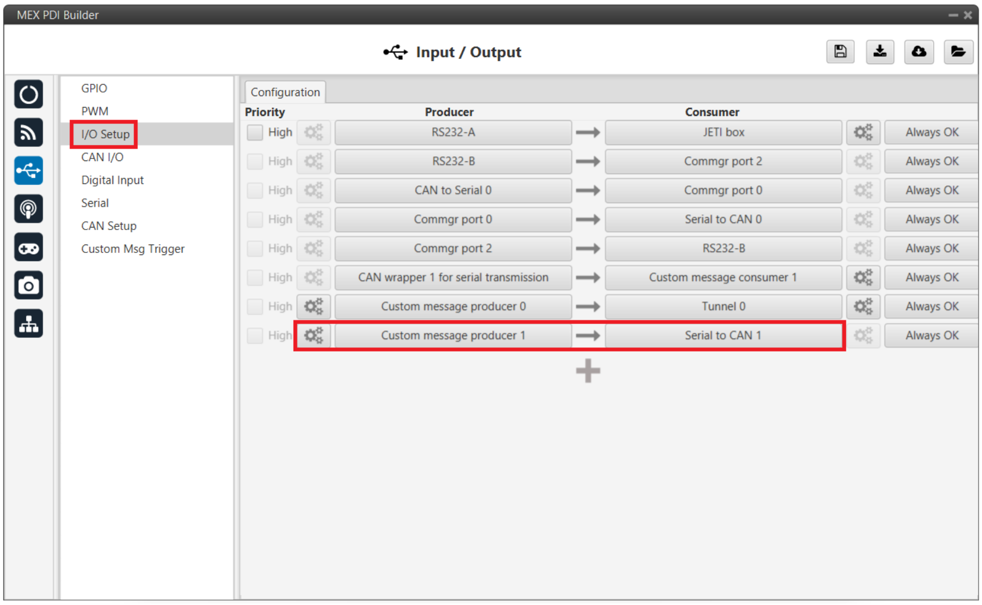
The configuration for the **Autopilot 1x** is explained in MEX as External Magnetometer - Integration examples section of the **1x PDI Builder** user manual.

CAN

MEX PDI Builder side

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

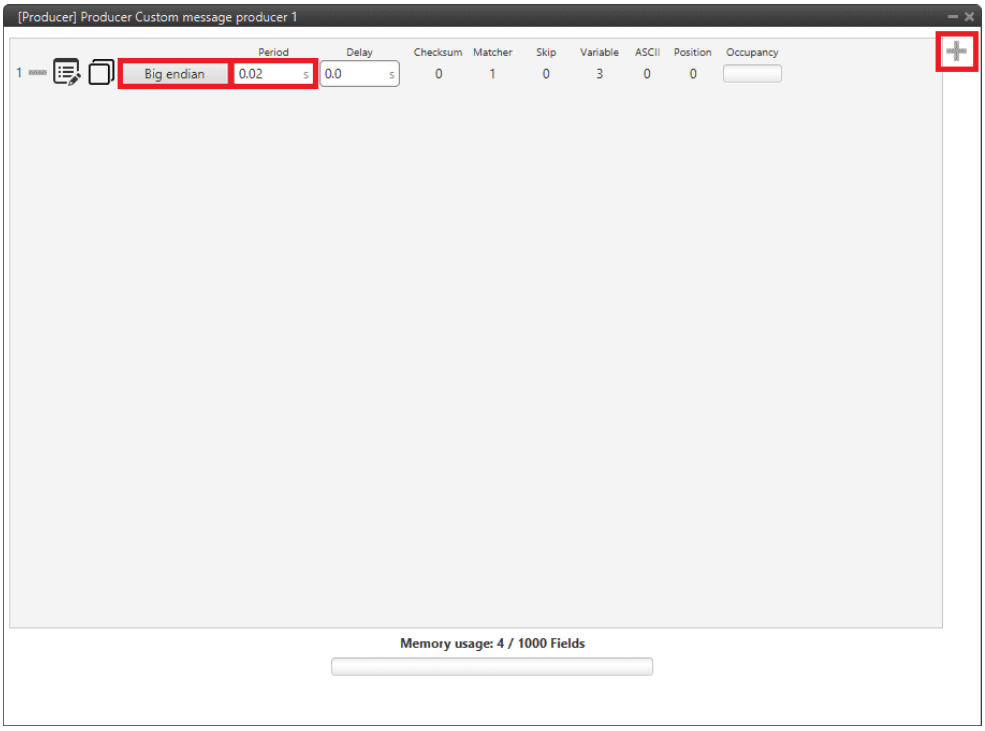
Connect the **Custom message producer 1** or **2** to the **Serial to CAN 1**, **Serial to CAN 2** or **Serial to CAN 3** port.



Priority	Producer		Consumer	
High	RS232-A	→	JETI box	Always OK
High	RS232-B	→	Commgr port 2	Always OK
High	CAN to Serial 0	→	Commgr port 0	Always OK
High	Commgr port 0	→	Serial to CAN 0	Always OK
High	Commgr port 2	→	RS232-B	Always OK
High	CAN wrapper 1 for serial transmission	→	Custom message consumer 1	Always OK
High	Custom message producer 0	→	Tunnel 0	Always OK
High	Custom message producer 1	→	Serial to CAN 1	Always OK

MEX PDI Builder - I/O Setup configuration

2. Click on the ⚙ button of the chosen **Custom message producer** and add a new message ＋, then configure it as **Big endian** and set a **Period of 0.02 s**.

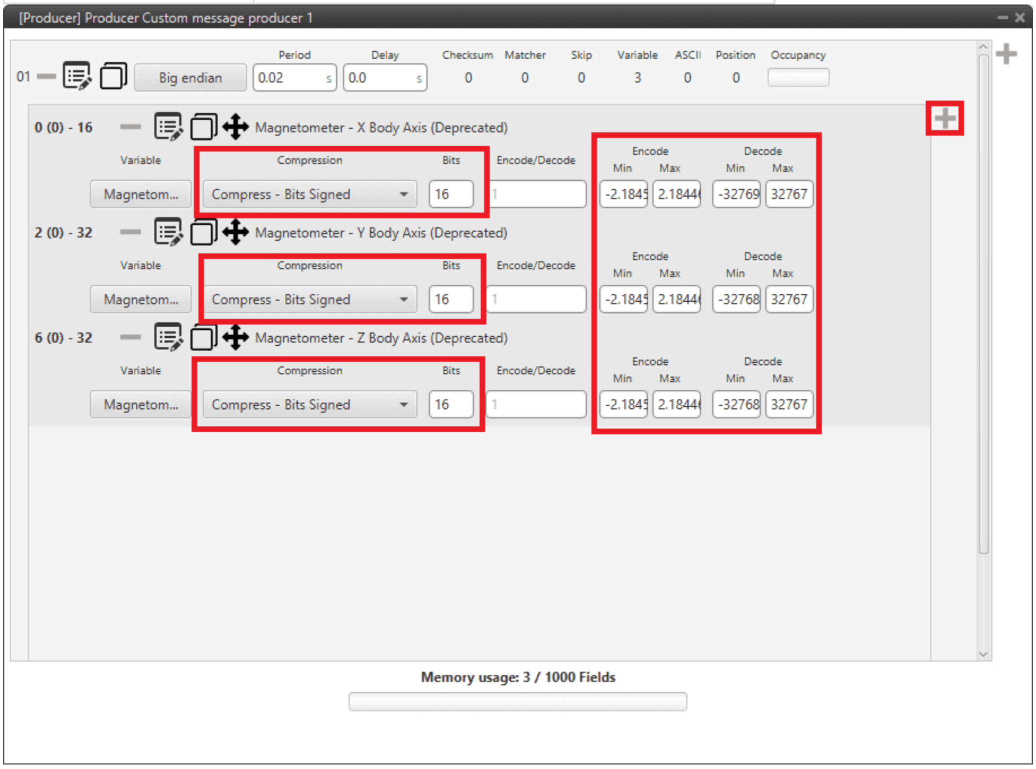

[Producer] Producer Custom message producer 1
Memory usage: 4 / 1000 Fields

		Period	Delay	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
1	Big endian	0.02 s	0.0 s	0	1	0	3	0	0	

MEX PDI Builder - Custom message configuration

3. Click on ＋ to add the 3 following **Real variables**: **313**, **314** and **315**. Then configure them according to the following table:

Parameter	Configuration
Compression	Compress - Bits Signed
Bits	16
Encode	Min: -2.1845334E-4 / Max: 2.1844667E-4
Decode	Min: -32768 / Max: 32767


[Producer] Producer Custom message producer 1

01 Big endian Period: 0.02 s Delay: 0.0 s Checksum: 0 Matcher: 0 Skip: 0 Variable: 3 ASCII: 0 Position: 0 Occupancy

0 (0) - 16 Magnetometer - X Body Axis (Deprecated)
Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32769 Max: 32767

2 (0) - 32 Magnetometer - Y Body Axis (Deprecated)
Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32768 Max: 32767

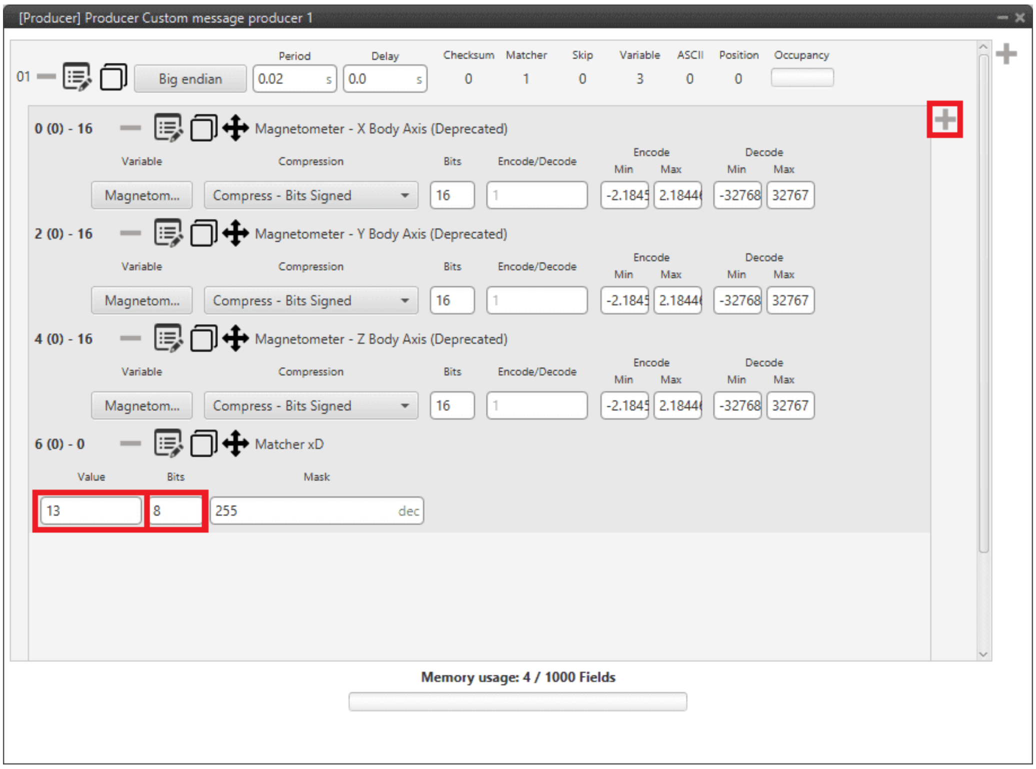
6 (0) - 32 Magnetometer - Z Body Axis (Deprecated)
Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32768 Max: 32767

Memory usage: 3 / 1000 Fields

MEX PDI Builder - Custom message Variables configuration

1. Click on ➕ → **Matcher**, to add a matcher to the message and configure it as follows:

Parameter	Configuration
Value	13
Bits	8



[Producer] Producer Custom message producer 1

01 Big endian Period 0.02 s Delay 0.0 s Checksum 0 Matcher 1 Skip 0 Variable 3 ASCII 0 Position 0 Occupancy

0 (0) - 16 Magnetometer - X Body Axis (Deprecated)

Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32768 Max: 32767

2 (0) - 16 Magnetometer - Y Body Axis (Deprecated)

Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32768 Max: 32767

4 (0) - 16 Magnetometer - Z Body Axis (Deprecated)

Variable: Magnetom... Compression: Compress - Bits Signed Bits: 16 Encode/Decode: 1 Encode Min: -2.1845 Max: 2.1844 Decode Min: -32768 Max: 32767

6 (0) - 0 Matcher xD

Value: 13 Bits: 8 Mask: 255 dec

Memory usage: 4 / 1000 Fields

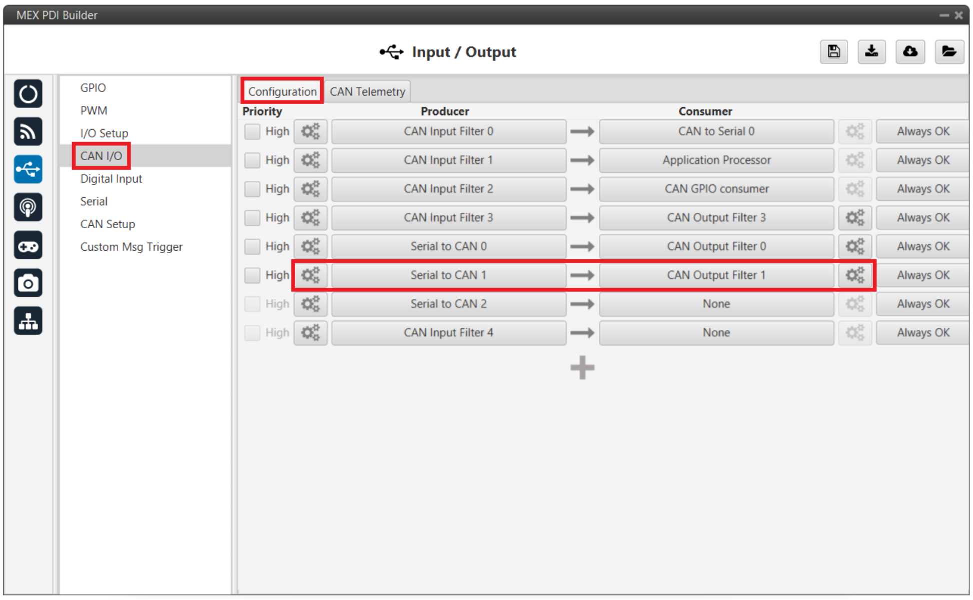
MEX PDI Builder - Custom message Matcher configuration

⚠ Warning

The order of variables and matcher must be the same as shown above.

1. Go to Input/Output menu → CAN I/O panel → **Configuration tab**. Connect the **Serial to CAN 1**, **2** or **3** to a **CAN Output Filter**.

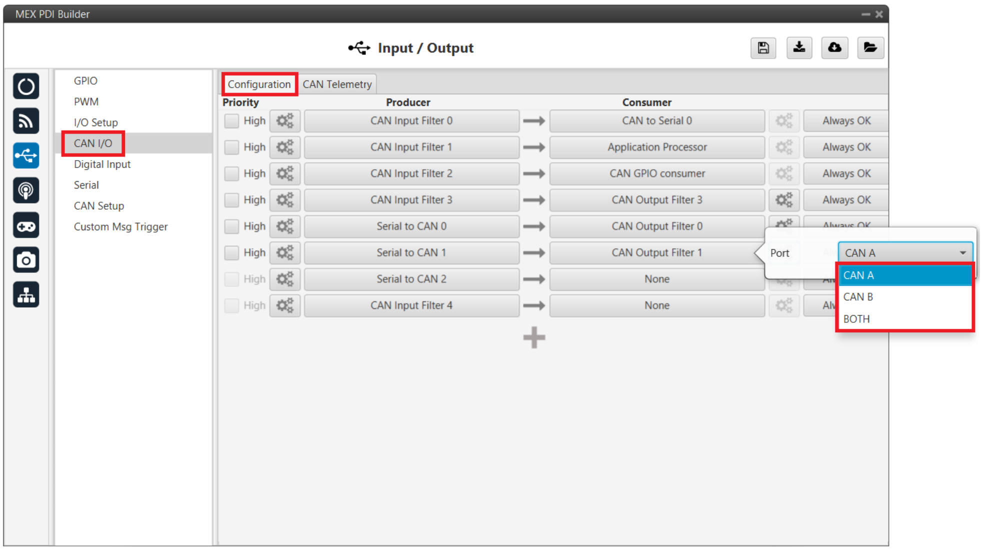
   > ### ⓘ Note
   >
   > The **Serial to CAN** index must match the one previously selected in the **I/O Setup panel** (step 1).



Priority	Producer	Consumer	
□ High	CAN Input Filter 0	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	Application Processor	Always OK
□ High	CAN Input Filter 2	CAN GPIO consumer	Always OK
□ High	CAN Input Filter 3	CAN Output Filter 3	Always OK
□ High	Serial to CAN 0	CAN Output Filter 0	Always OK
□ High	Serial to CAN 1	CAN Output Filter 1	Always OK
□ High	Serial to CAN 2	None	Always OK
□ High	CAN Input Filter 4	None	Always OK

MEX PDI Builder - CAN I/O configuration

2. Select the **CAN port** by clicking on the ⚙ button of the chosen **CAN Output Filter**.



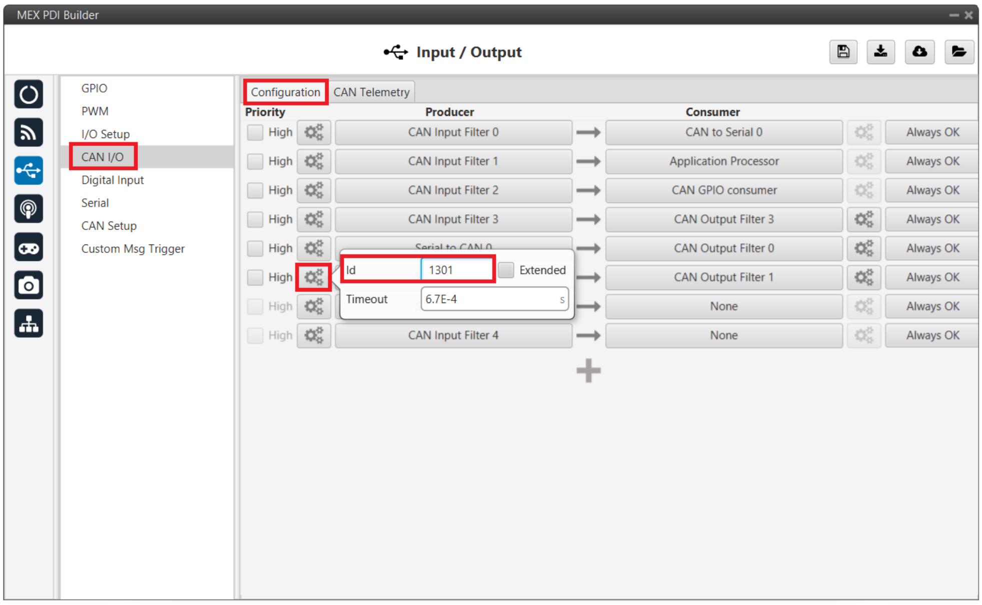
Priority	Producer	Consumer	
□ High	CAN Input Filter 0	CAN to Serial 0	Always OK
□ High	CAN Input Filter 1	Application Processor	Always OK
□ High	CAN Input Filter 2	CAN GPIO consumer	Always OK
□ High	CAN Input Filter 3	CAN Output Filter 3	Always OK
□ High	Serial to CAN 0	CAN Output Filter 0	Always OK
□ High	Serial to CAN 1	CAN Output Filter 1	
□ High	Serial to CAN 2	None	
□ High	CAN Input Filter 4	None	Al

Port: CAN A

MEX PDI Builder - CAN Output Filter configuration

3. Asign a CAN ID to the **Serial to CAN 1**.

> ⓘ **Note**
>
> This ID must match the one configured for reception in **Autopilot 1x** with the **1x PDI Builder** software.



Priority		Producer		Consumer	
□	High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
□	High	CAN Input Filter 1	→	Application Processor	Always OK
□	High	CAN Input Filter 2	→	CAN GPIO consumer	Always OK
□	High	CAN Input Filter 3	→	CAN Output Filter 3	Always OK
□	High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
□	High		→	CAN Output Filter 1	Always OK
□	High		→	None	Always OK
□	High	CAN Input Filter 4	→	None	Always OK

Id 1301 □ Extended

Timeout 6.7E-4 s

MEX PDI Builder - Serial to CAN configuration

1x PDI Builder side

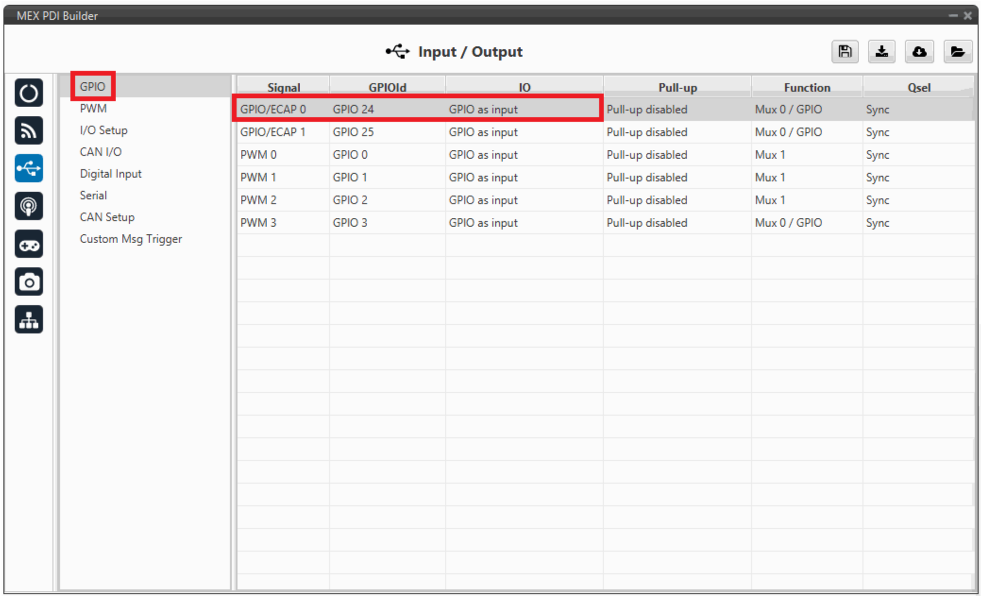
The configuration for the **Autopilot 1x** is explained in MEX as External Magnetometer - Integration examples section of the **1x PDI Builder** user manual.

Reading/Sending RPMs

This section explains the steps to read RPMs.

MEX PDI Builder side

1. Go to Input/Output menu → **GPIO panel**.
   RPM can be read on the available digital inputs GPIO/ECAP 0 and 1. The chosen pin needs to be configured as "**GPIO as input**". In the example shown here, GPIO/ECAP 0 is chosen.



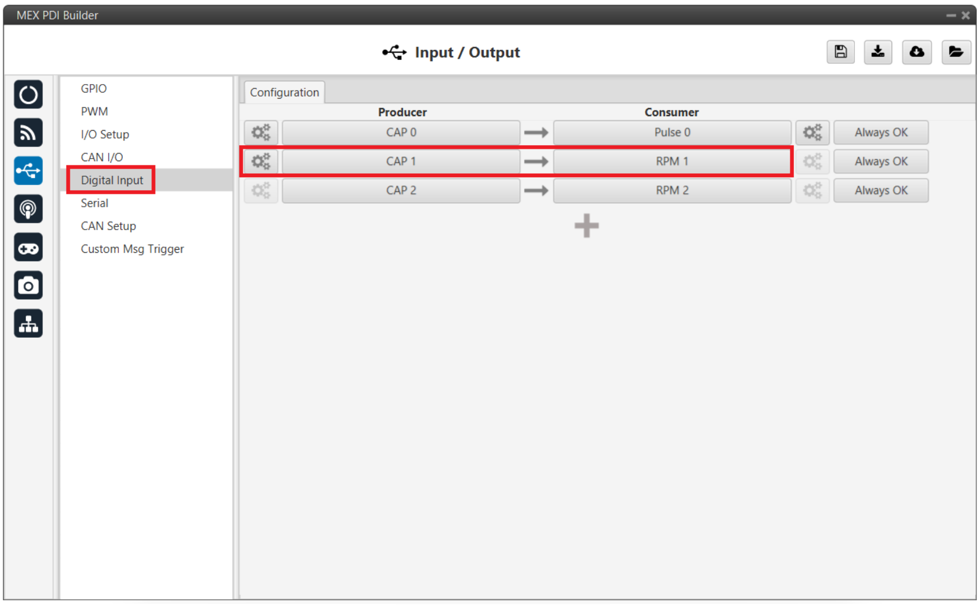
Signal	GPIOId	IO	Pull-up	Function	Qsel
GPIO/ECAP 0	GPIO 24	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
GPIO/ECAP 1	GPIO 25	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync
PWM 0	GPIO 0	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 1	GPIO 1	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 2	GPIO 2	GPIO as input	Pull-up disabled	Mux 1	Sync
PWM 3	GPIO 3	GPIO as input	Pull-up disabled	Mux 0 / GPIO	Sync

MEX PDI Builder - GPIO configuration

2. Go to Input/Output menu → **Digital Input panel**.

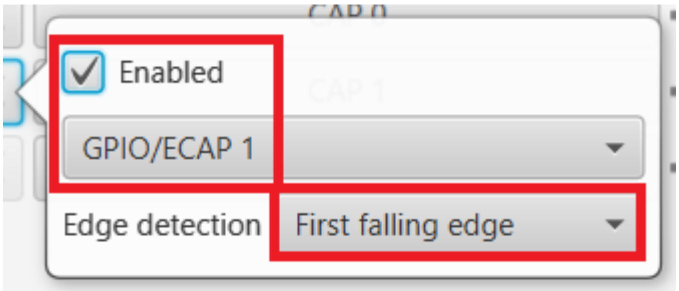
   There are 2 possible producers: CAP 1 and 2. One needs to be chosen and linked to one of the RPMs consumers (RPM 1-4). For more information about Digital input configuration, read the Digital Input - Input/Output section of this manual.

   Then, select an **edge detection option**: "First rising edge" or "First falling edge".



Producer		Consumer	
CAP 0	→	Pulse 0	Always OK
CAP 1	→	RPM 1	Always OK
CAP 2	→	RPM 2	Always OK

MEX PDI Builder - Digital Input configuration



☑ Enabled

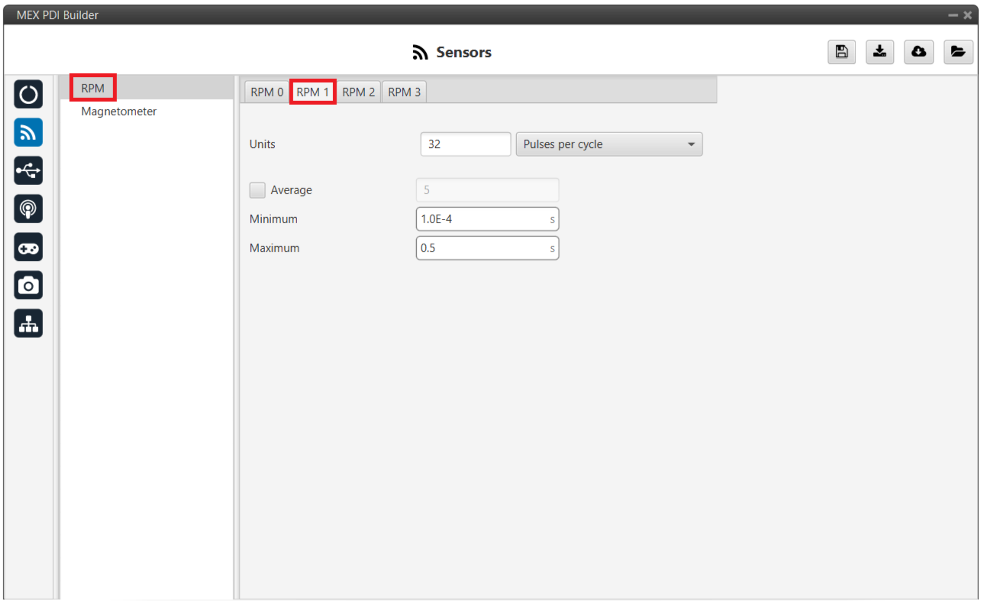
GPIO/ECAP 1

Edge detection: First falling edge

MEX PDI Builder - Edge detection configuration

3. Go to Sensors menu → RPM panel → **RPM 1 tab** (since RPM **1** has been selected in the Digital Input panel).

   Here the expected pulse needs to be defined. For more information on RPM configuration, see RPM - Sensors section of this manual.



Units: 32 Pulses per cycle

□ Average: 5

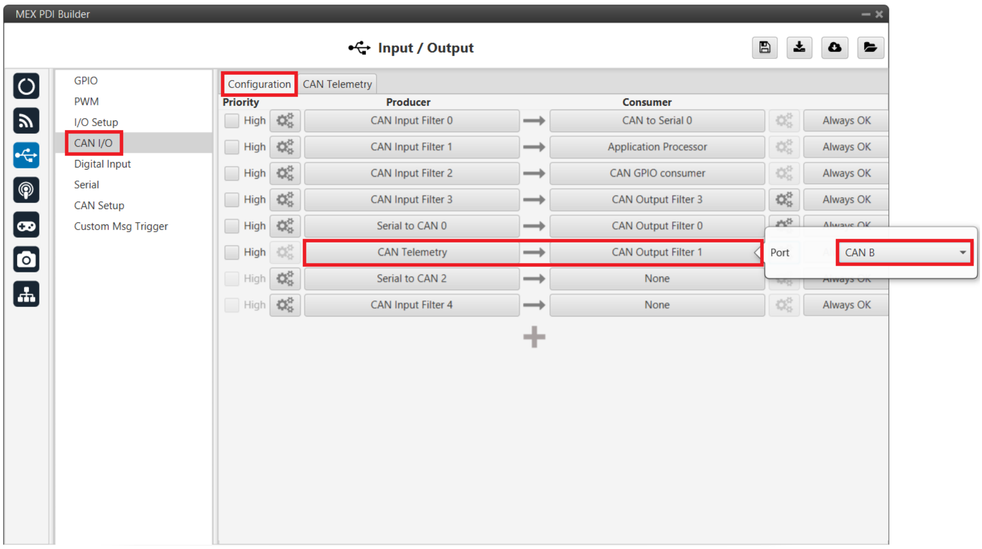
Minimum: 1.0E-4 s

Maximum: 0.5 s

MEX PDI Builder - RPM configuration

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

   Connect **CAN Telemetry** to a **CAN Output Filter** as follows (in this example, the **RPM1** variable is sent over **CAN B** bus of the **MEX**):



Priority	Producer		Consumer	
High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
High	CAN Input Filter 1	→	Application Processor	Always OK
High	CAN Input Filter 2	→	CAN GPIO consumer	Always OK
High	CAN Input Filter 3	→	CAN Output Filter 3	Always OK
High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
High	CAN Telemetry	→	CAN Output Filter 1	
High	Serial to CAN 2	→	None	Always OK
High	CAN Input Filter 4	→	None	Always OK

Port: CAN B

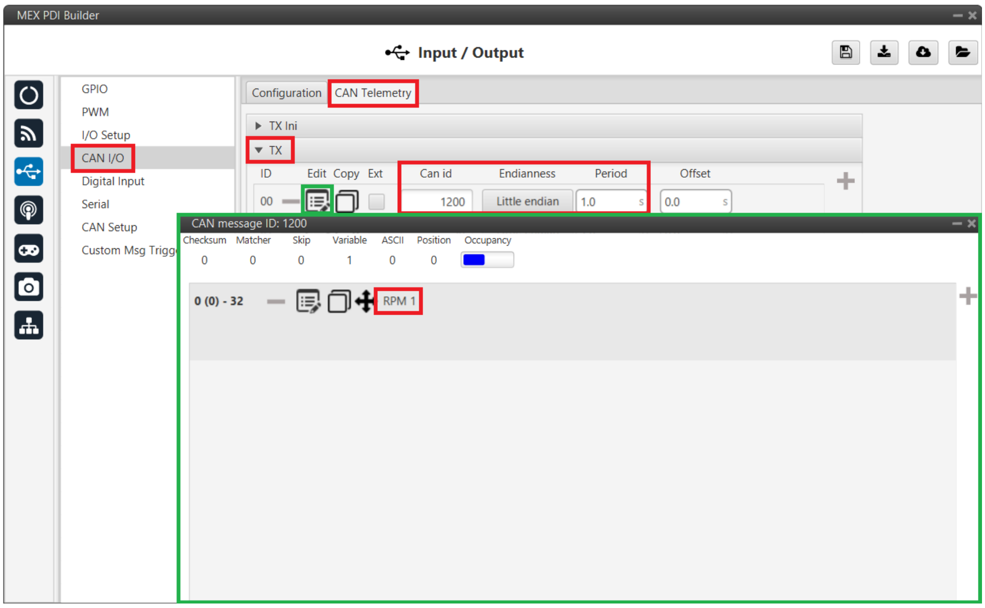
MEX PDI Builder - CAN I/O configuration

5. Go to Input/Output menu → CAN I/O panel → **CAN Telemetry tab**.

   A new telemetry message needs to be created with its correspondent ID, endianness and period.

   - **Can ID**: 1200.
   - **Endianness**: Little endian.
   - **Period**: 0.01 s.

   In the telemetry message one of MEX variables needs to be selected. As **RPM1** has been chosen as consumer in the **Digital Input**, the variable required to send is **RPM1**.



ID	Edit	Copy	Ext	Can id	Endianness	Period	Offset
00			□	1200	Little endian	1.0 s	0.0 s

CAN message ID: 1200

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	0	0	1	0	0	

0 (0) - 32 RPM 1

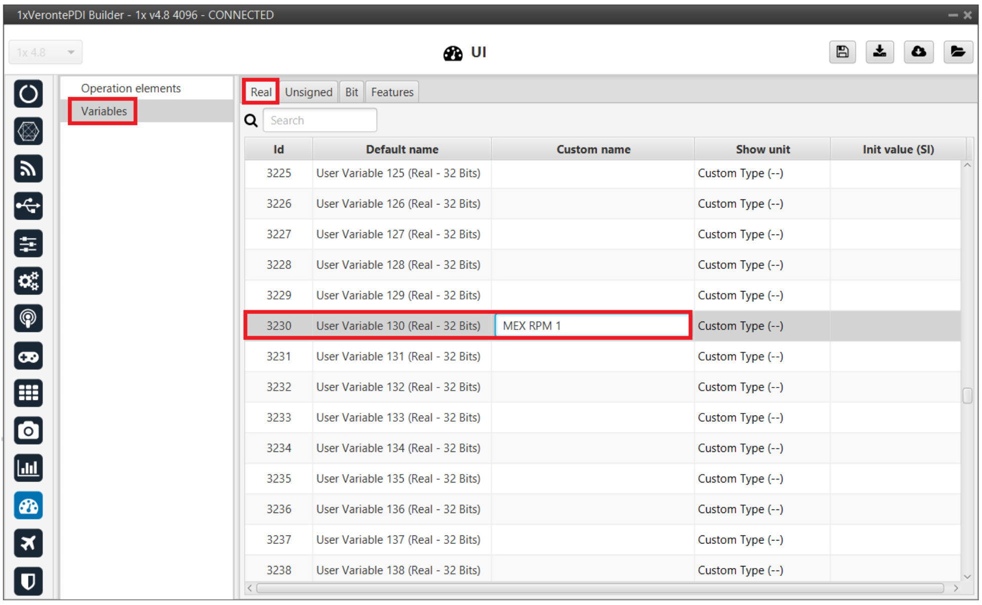
MEX PDI Builder - CAN Telemetry configuration

More information on the configuration of Telemetry messages can be found in the CAN Telemetry - Input/Output section of this manual.

1x PDI Builder side

1. Go to UI menu → Variables panel → **Real Vars tab**.

   Rename a Real User Variable (32 bits) that will be used to store the value received from **MEX**:



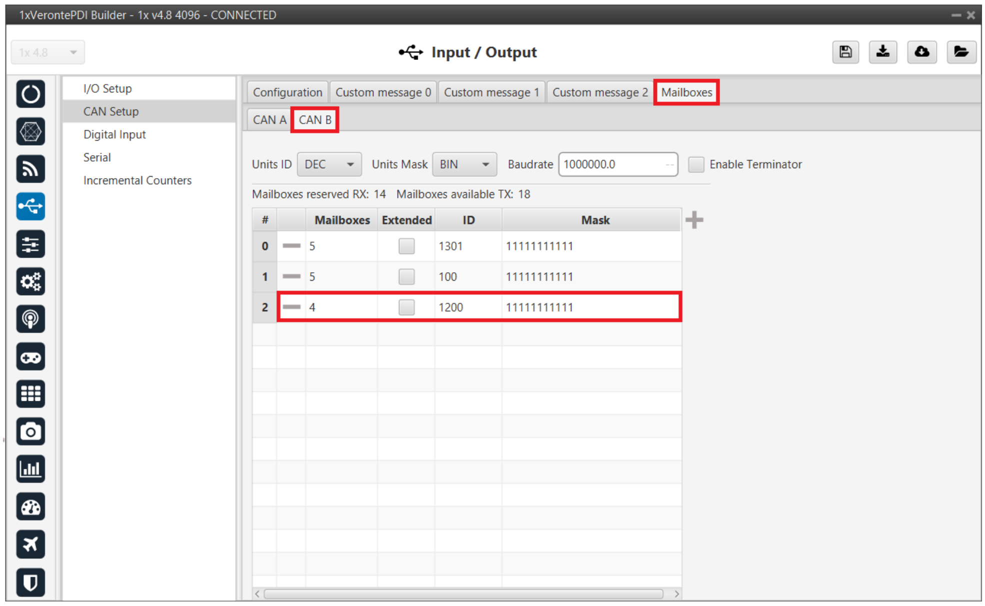
1xVerontePDI Builder - 1x v4.8 4096 - CONNECTED

Id	Default name	Custom name	Show unit	Init value (SI)
3225	User Variable 125 (Real - 32 Bits)		Custom Type (--)	
3226	User Variable 126 (Real - 32 Bits)		Custom Type (--)	
3227	User Variable 127 (Real - 32 Bits)		Custom Type (--)	
3228	User Variable 128 (Real - 32 Bits)		Custom Type (--)	
3229	User Variable 129 (Real - 32 Bits)		Custom Type (--)	
3230	User Variable 130 (Real - 32 Bits)	MEX RPM 1	Custom Type (--)	
3231	User Variable 131 (Real - 32 Bits)		Custom Type (--)	
3232	User Variable 132 (Real - 32 Bits)		Custom Type (--)	
3233	User Variable 133 (Real - 32 Bits)		Custom Type (--)	
3234	User Variable 134 (Real - 32 Bits)		Custom Type (--)	
3235	User Variable 135 (Real - 32 Bits)		Custom Type (--)	
3236	User Variable 136 (Real - 32 Bits)		Custom Type (--)	
3237	User Variable 137 (Real - 32 Bits)		Custom Type (--)	
3238	User Variable 138 (Real - 32 Bits)		Custom Type (--)	

1x PDI Builder - User Variable renamed

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

   Configure, in **CAN B**, some mailboxes to receive the message with **ID 1200**:



1xVerontePDI Builder - 1x v4.8 4096 - CONNECTED

Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

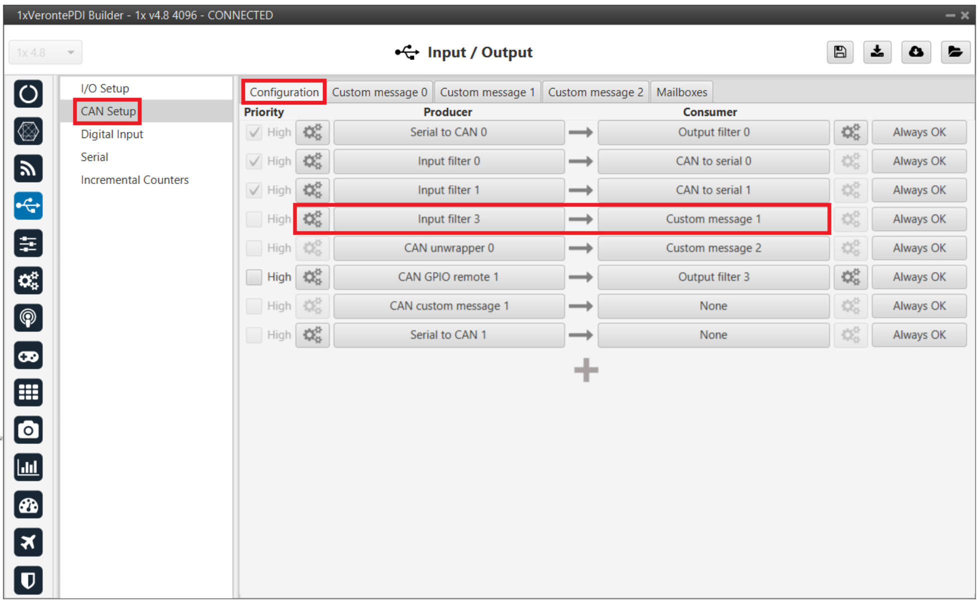
Mailboxes reserved RX: 14 Mailboxes available TX: 18

#	Mailboxes	Extended	ID	Mask
0	5	□	1301	11111111111
1	5	□	100	11111111111
2	4	□	1200	11111111111

1x PDI Builder - Mailboxes configuration

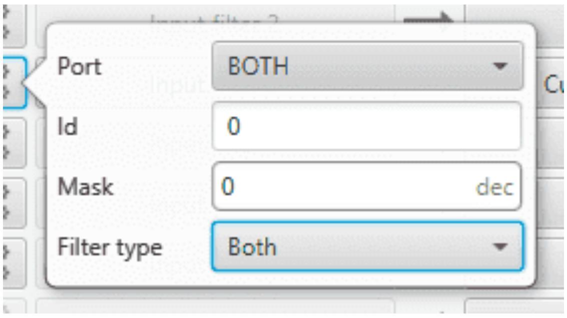
3. Go to Input/Output menu → CAN Setup panel → **Configuration tab**.

Connect an **Input filter** to a **Custom message consumer**. Both CAN buses of the Autopilot 1x can be used, as well as normal IDs and extended IDs:



Priority	Producer		Consumer	
☑ High	Serial to CAN 0	→	Output filter 0	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 3	→	Custom message 1	Always OK
□ High	CAN unwrapper 0	→	Custom message 2	Always OK
□ High	CAN GPIO remote 1	→	Output filter 3	Always OK
□ High	CAN custom message 1	→	None	Always OK
□ High	Serial to CAN 1	→	None	Always OK

1x PDI Builder - CAN Setup configuration



Port: BOTH

Id: 0

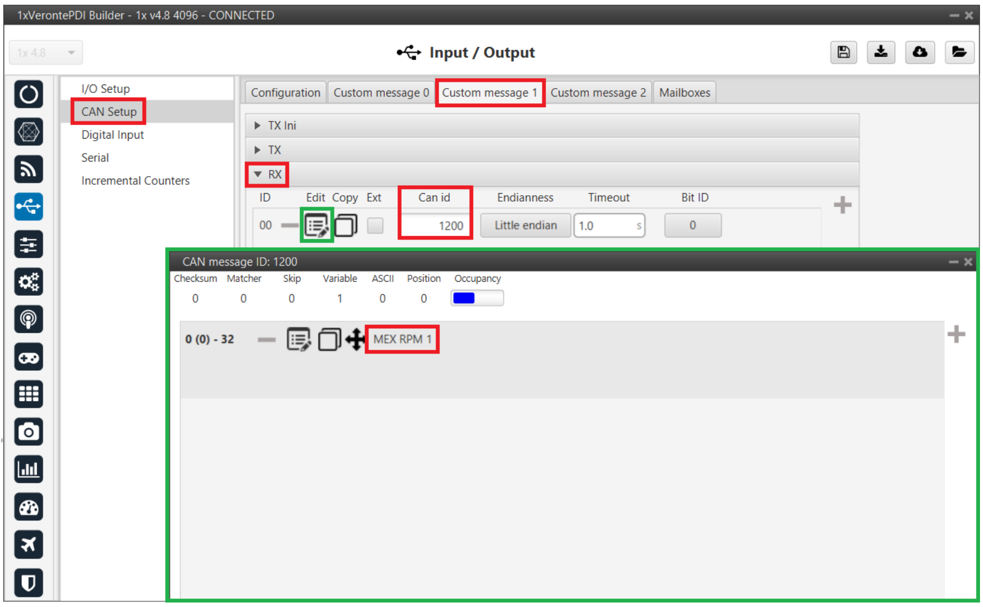
Mask: 0 dec

Filter type: Both

1x PDI Builder - Input filter configuration

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

   Configure the message reading and store the received value in the user variable renamed above:



ID	Edit	Copy	Ext	Can id	Endianness	Timeout	Bit ID
00			□	1200	Little endian	1.0 s	0

CAN message ID: 1200

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	0	0	1	0	0	

0 (0) - 32 MEX RPM 1

1x PDI Builder - Custom Message configuration

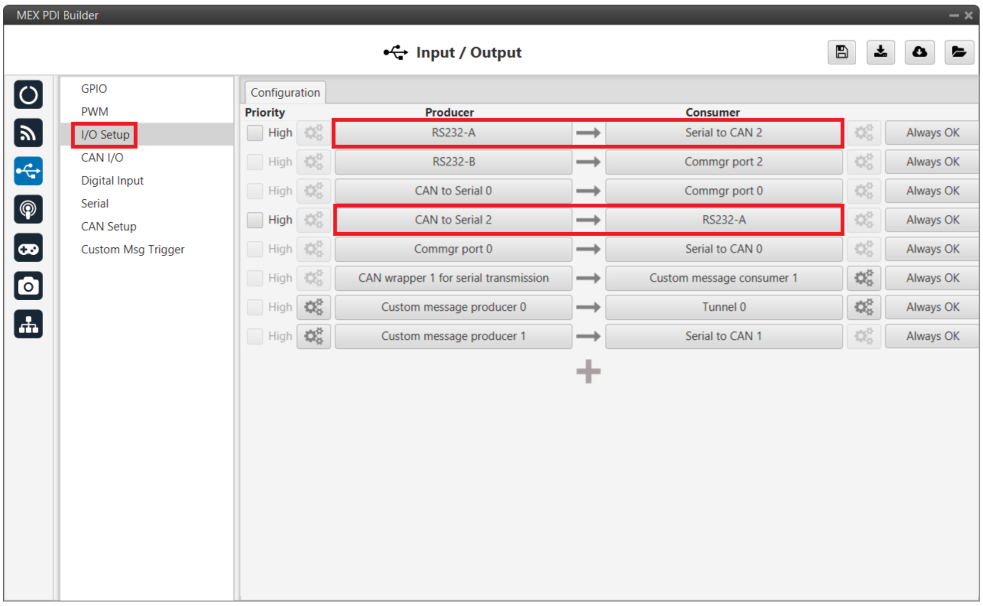
Serial communication

The user can send information through the serial ports of **MEX**. For this purpose, please read the following steps:

MEX PDI Builder side

1. **I/O Setup configuration**
   - Reception of serial information on RS232-A (producer) is stored in Serial to CAN 2 (consumer).
   - Transmission of serial information is sent using the CAN to Serial 2 (producer) through RS232-A (consumer).

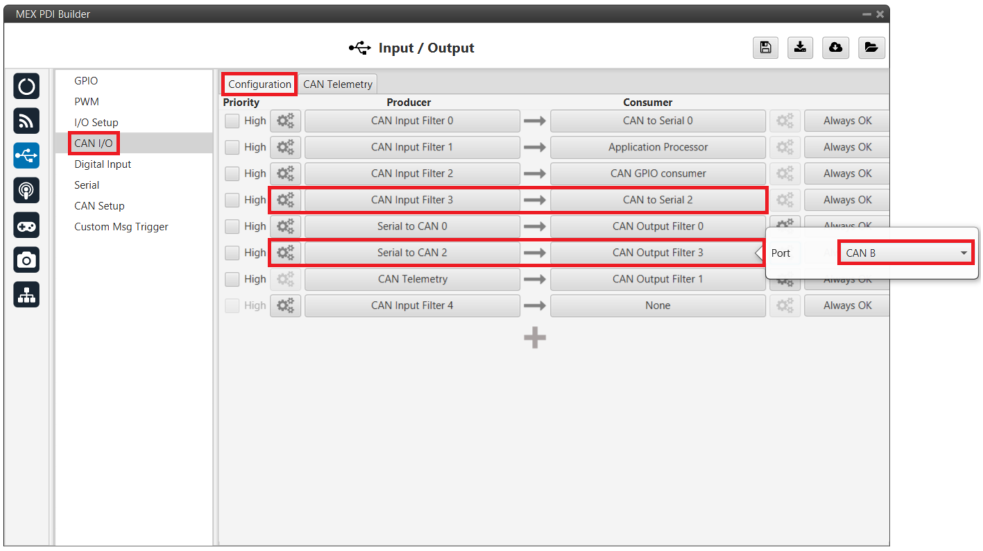
   ('Serial to CAN' and 'CAN to Serial' are explained in the CAN I/O - Input/Output section of this manual).



Priority	Producer		Consumer	
High	RS232-A	→	Serial to CAN 2	Always OK
High	RS232-B	→	Commgr port 2	Always OK
High	CAN to Serial 0	→	Commgr port 0	Always OK
High	CAN to Serial 2	→	RS232-A	Always OK
High	Commgr port 0	→	Serial to CAN 0	Always OK
High	CAN wrapper 1 for serial transmission	→	Custom message consumer 1	Always OK
High	Custom message producer 0	→	Tunnel 0	Always OK
High	Custom message producer 1	→	Serial to CAN 1	Always OK

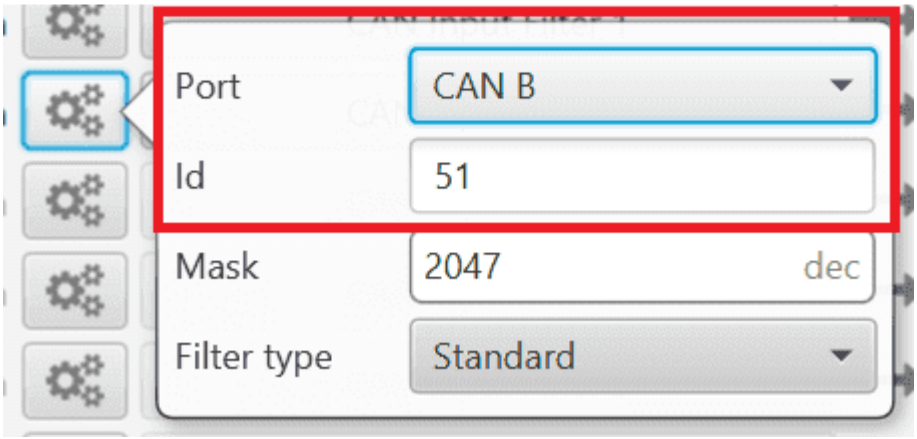
MEX PDI Builder - I/O Setup configuration

2. **CAN I/O configuration**
   - The information that will be **sent over serial port RS232-A** is going to be received on the **MEX** over its **CAN B** port. A **CAN Id of 51** is added to the **CAN Input Filter**. The incoming information (from **Veronte Autopilot 1x**) is processed in '**CAN to Serial 2**'.
   - The information **coming from port RS232-A** and processed as '**Serial to CAN 2**' is going to be linked to a **CAN Output Filter**. The information of '**Serial to CAN 2**' is going to be sent over **CAN B** with a **CAN ID of 50** (to be read by **Veronte Autopilot 1x**).



Priority	Producer		Consumer	
High	CAN Input Filter 0	→	CAN to Serial 0	Always OK
High	CAN Input Filter 1	→	Application Processor	Always OK
High	CAN Input Filter 2	→	CAN GPIO consumer	Always OK
High	CAN Input Filter 3	→	CAN to Serial 2	Always OK
High	Serial to CAN 0	→	CAN Output Filter 0	Always OK
High	Serial to CAN 2	→	CAN Output Filter 3	Port: CAN B
High	CAN Telemetry	→	CAN Output Filter 1	Always OK
High	CAN Input Filter 4	→	None	Always OK

MEX PDI Builder - CAN I/O configuration



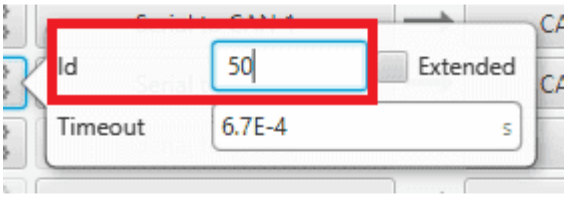
Port: CAN B

Id: 51

Mask: 2047 dec

Filter type: Standard

MEX PDI Builder - CAN Input Filter configuration



Id: 50

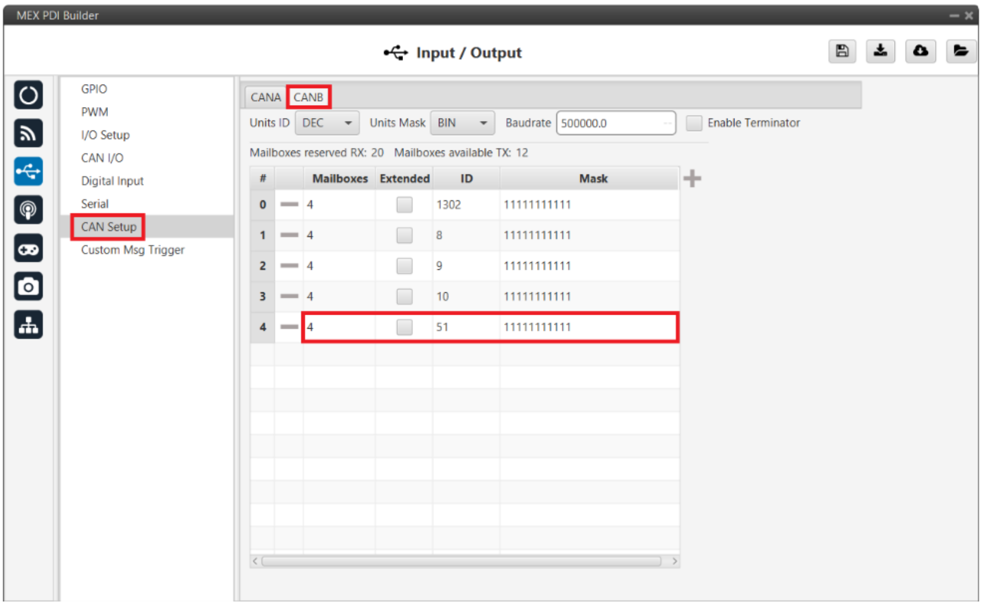
☐ Extended

Timeout: 6.7E-4 s

MEX PDI Builder - Serial to CAN configuration

3. **CAN Setup (mailboxes) configuration**

   Mailboxes need to be defined for the reception of CAN messages. In the example above, mailboxes for **ID 51** need to be added on **CAN B** port.



Units ID: DEC   Units Mask: BIN   Baudrate: 500000.0   ☐ Enable Terminator

Mailboxes reserved RX: 20   Mailboxes available TX: 12

#	Mailboxes	Extended	ID	Mask
0	4	☐	1302	11111111111
1	4	☐	8	11111111111
2	4	☐	9	11111111111
3	4	☐	10	11111111111
4	4	☐	51	11111111111

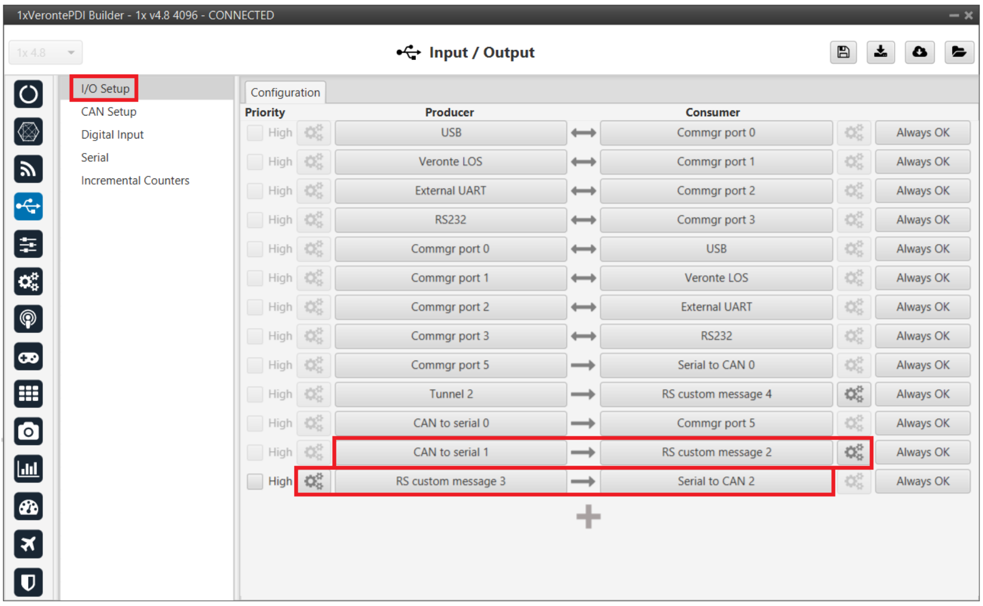
MEX PDI Builder - CAN Setup configuration

1x PDI Builder side

1. **I/O Setup configuration**

   First, o n the **I/O Setup panel** link an '**RS custom message**' to a '**Serial to CAN**' with the serial data that **Autopilot 1x** is going to send to **MEX**.

Then, link a '**CAN to Serial**' to another '**RS custom message**' with the expected serial messages that **MEX** will receive in the selected serial port.



1xVerontePDI Builder - 1x v4.8 4096 - CONNECTED

Input / Output

I/O Setup
CAN Setup
Digital Input
Serial
Incremental Counters

Configuration

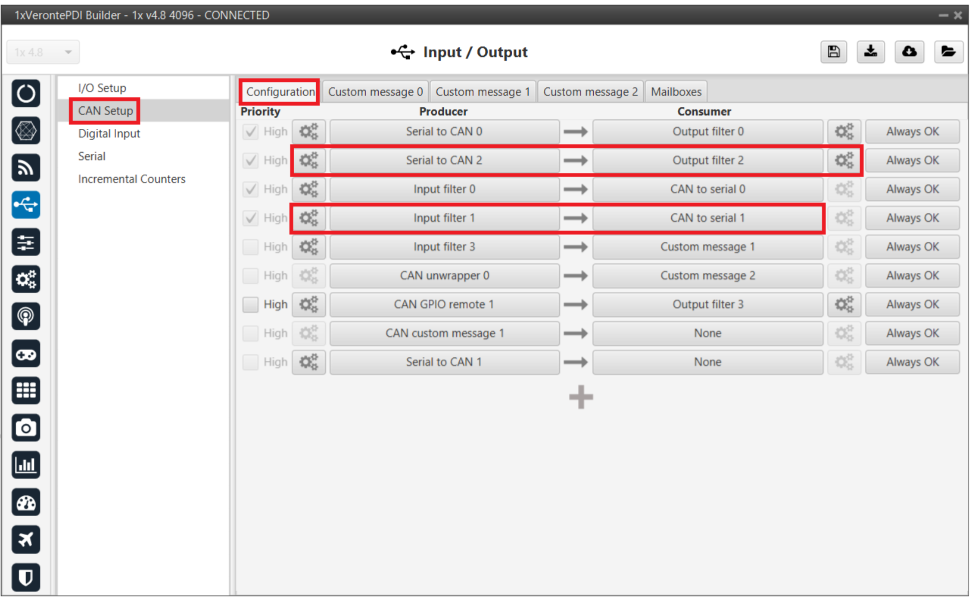
Priority	Producer		Consumer	
□ High	USB	⟷	Commgr port 0	Always OK
□ High	Veronte LOS	⟷	Commgr port 1	Always OK
□ High	External UART	⟷	Commgr port 2	Always OK
□ High	RS232	⟷	Commgr port 3	Always OK
□ High	Commgr port 0	⟷	USB	Always OK
□ High	Commgr port 1	⟷	Veronte LOS	Always OK
□ High	Commgr port 2	⟷	External UART	Always OK
□ High	Commgr port 3	⟷	RS232	Always OK
□ High	Commgr port 5	→	Serial to CAN 0	Always OK
□ High	Tunnel 2	→	RS custom message 4	Always OK
□ High	CAN to serial 0	→	Commgr port 5	Always OK
□ High	CAN to serial 1	→	RS custom message 2	Always OK
□ High	RS custom message 3	→	Serial to CAN 2	Always OK

1x PDI Builder - I/O Setup configuration

2. **CAN Setup configuration**

   Here, the same IDs employed in **MEX** for the CAN Input and CAN Output Filters are going to be employed on **Veronte Autopilot 1x** side, but they must be inverted.

   Therefore, the **Input filter** linked to the chosen '**CAN to Serial**' needs to have **ID 50**. And the **Output filter** linked to the chosen '**Serial to CAN**' will have **ID 51**.



1xVerontePDI Builder - 1x v4.8 4096 - CONNECTED

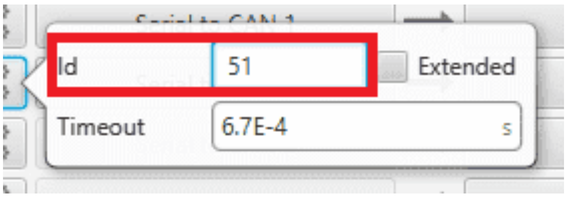
Input / Output

I/O Setup
CAN Setup
Digital Input
Serial
Incremental Counters

Configuration | Custom message 0 | Custom message 1 | Custom message 2 | Mailboxes

Priority	Producer		Consumer	
☑ High	Serial to CAN 0	→	Output filter 0	Always OK
☑ High	Serial to CAN 2	→	Output filter 2	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 3	→	Custom message 1	Always OK
□ High	CAN unwrapper 0	→	Custom message 2	Always OK
□ High	CAN GPIO remote 1	→	Output filter 3	Always OK
□ High	CAN custom message 1	→	None	Always OK
□ High	Serial to CAN 1	→	None	Always OK

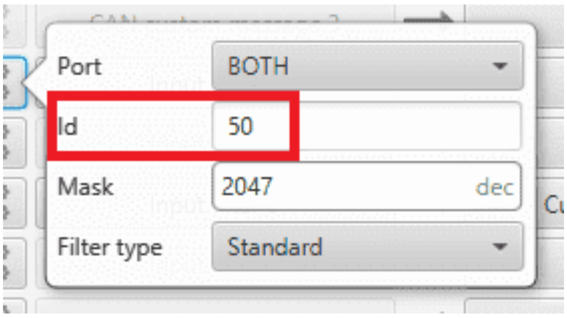
1x PDI Builder - CAN Setup configuration



Id 51 □ Extended

Timeout 6.7E-4 s

1x PDI Builder - Serial to CAN configuration



Port BOTH

Id 50

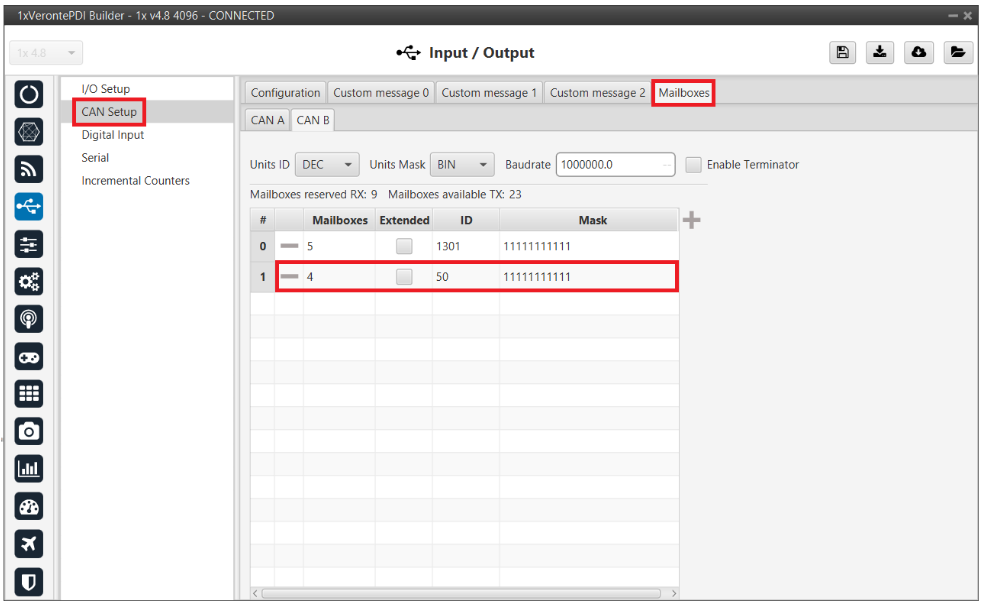
Mask 2047 dec

Filter type Standard

1x PDI Builder - Input filter configuration

3. **Mailboxes configuration**

   Some **mailboxes** with **ID 50** have to be created on whichever chosen reception CAN bus.



Units ID DEC Units Mask BIN Baudrate 1000000.0 □ Enable Terminator

Mailboxes reserved RX: 9 Mailboxes available TX: 23

#	Mailboxes	Extended	ID	Mask
0	5	□	1301	11111111111
1	4	□	50	11111111111

1x PDI Builder - Mailboxes configuration

External devices

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

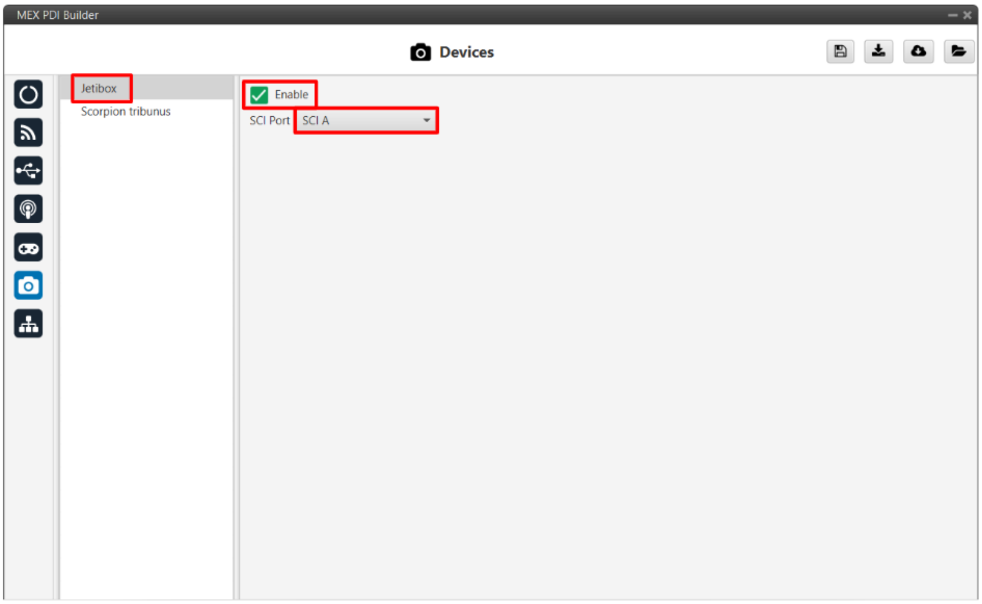
- Jetibox
- Scorpion tribunus
- Veronte products

Jetibox

MEX is able to simulate a Jetibox to read telemetry from Legacy Jeti devices, the following steps need to be taken:

1. Go to Devices menu → **Jetibox panel**.

   Enable it and select a **SCI Port**, in this example SCI A is selected:

   ☑ Enable

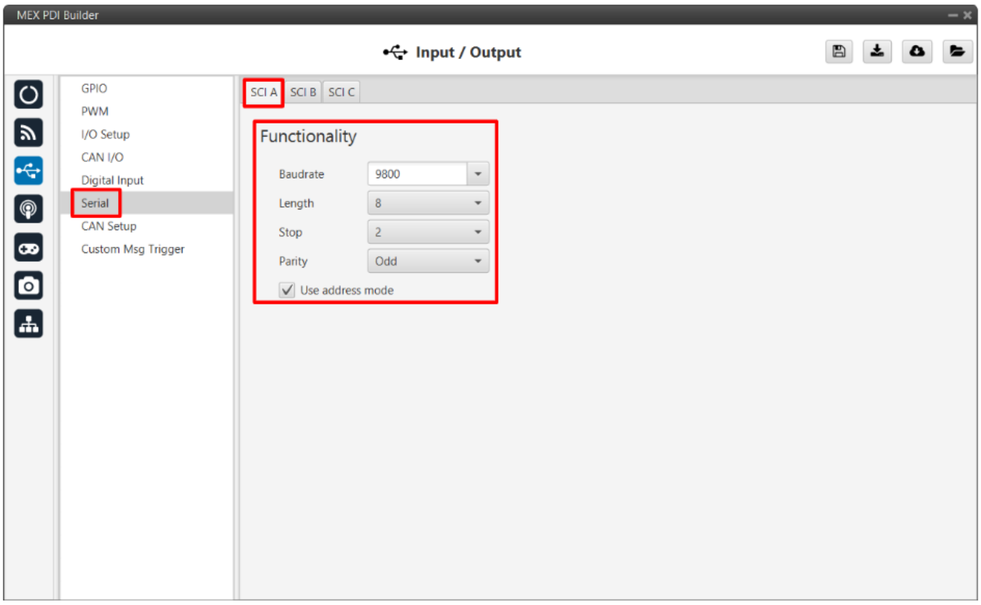
   SCI Port: SCI A

   Jetibox - Devices configuration

2. Go to Input/Output menu → **Serial panel**.

   Configure the serial port (the one selected in step **1**) as follows:

   - **Baudrate**: 9800
   - **Length**: 8
   - **Stop**: 2 stop bits
   - **Parity**: Odd
   - **Use address mode**: Enabled



Functionality

Baudrate	9800
Length	8
Stop	2
Parity	Odd

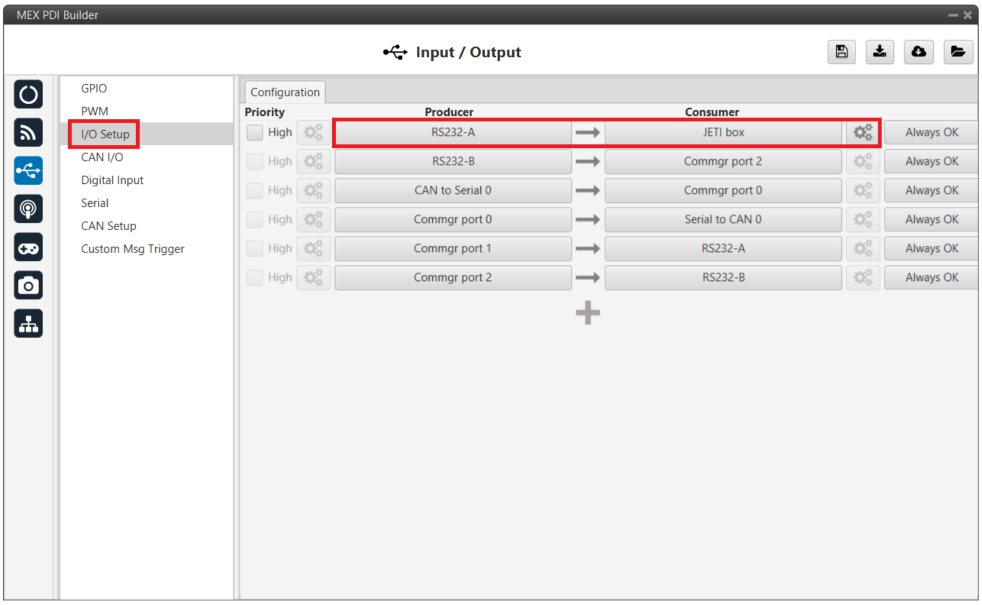
☑ Use address mode

Jetibox - Serial configuration

> **Note**
>
> The serial port will be completely reserved for this, so it cannot be used for any other purpose and the I/O Setup configuration affecting it will be ignored.

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

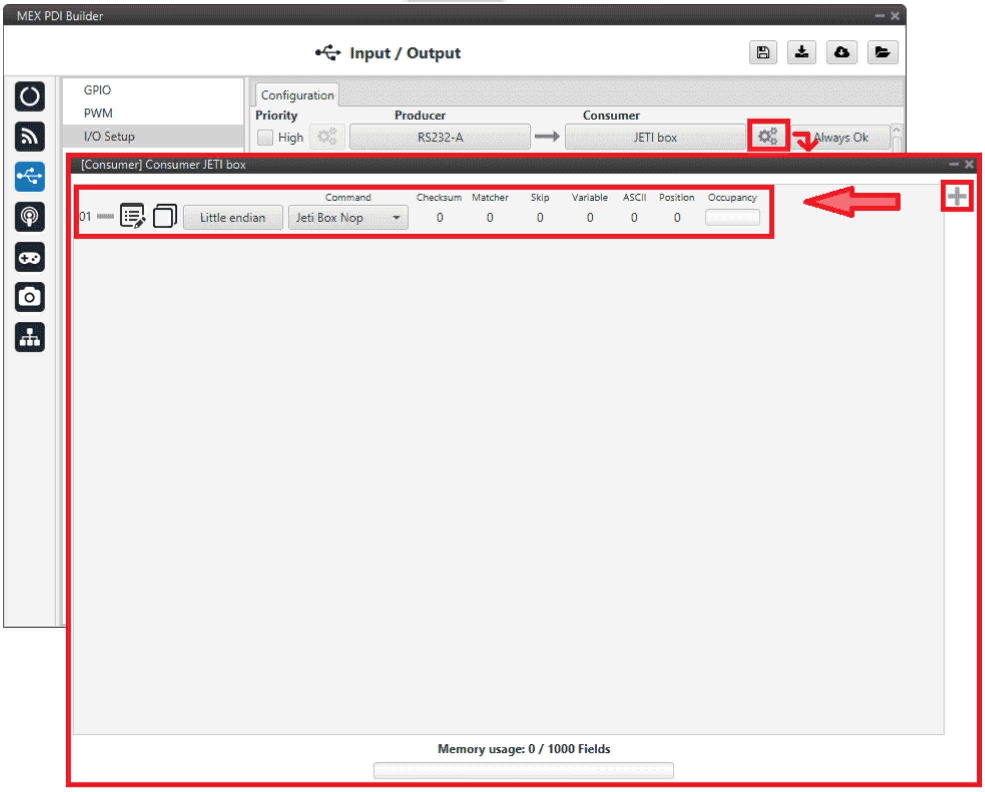
   Link the specific **JETI box** consumer to the **serial port** that has been configured in **step 2**:



Priority	Producer		Consumer	
□ High	RS232-A	→	JETI box	Always OK
□ High	RS232-B	→	Commgr port 2	Always OK
□ High	CAN to Serial 0	→	Commgr port 0	Always OK
□ High	Commgr port 0	→	Serial to CAN 0	Always OK
□ High	Commgr port 1	→	RS232-A	Always OK
□ High	Commgr port 2	→	RS232-B	Always OK

Jetibox - I/O Setup configuration

4. Configure the Jetibox I/O consumer to retrieve the data. Click on the **configuration button** (⚙):



Priority	Producer	Consumer	
□ High	RS232-A	JETI box	Always Ok

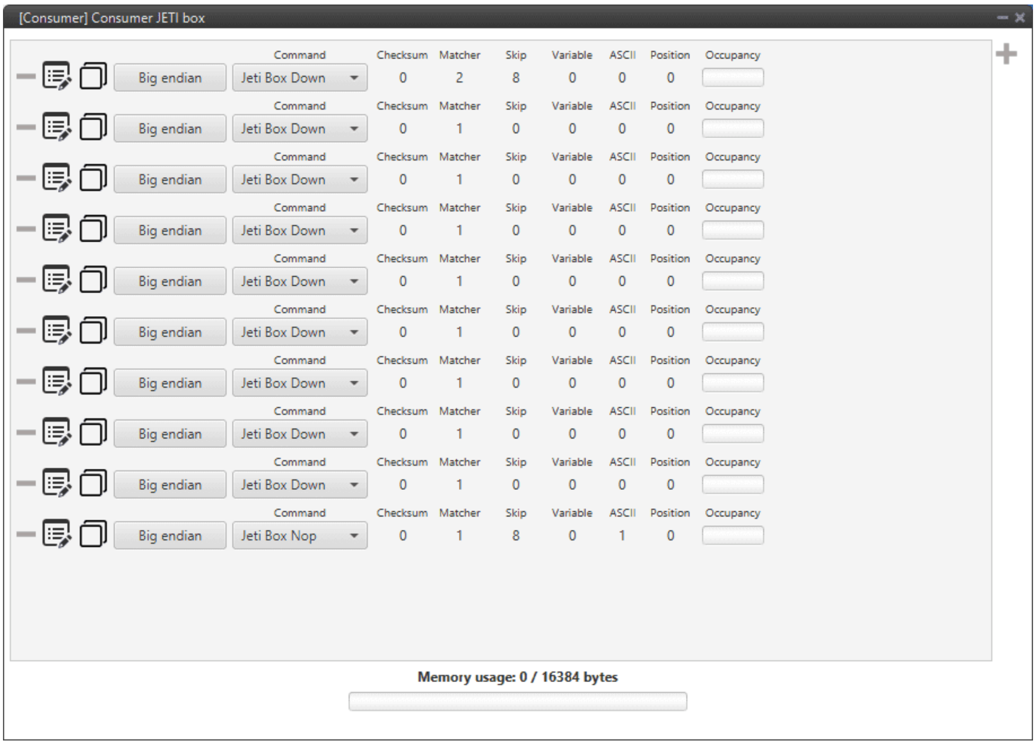
[Consumer] Consumer JETI box

		Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
01	Little endian	Jeti Box Nop	0	0	0	0	0	0	

Memory usage: 0 / 1000 Fields

Jetibox - JETI box Consumer configuration

Below is an example of the custom messages needed to read the Actual Voltage from a Jeti MasterSpin 220 (use **Big endian** in all messages):



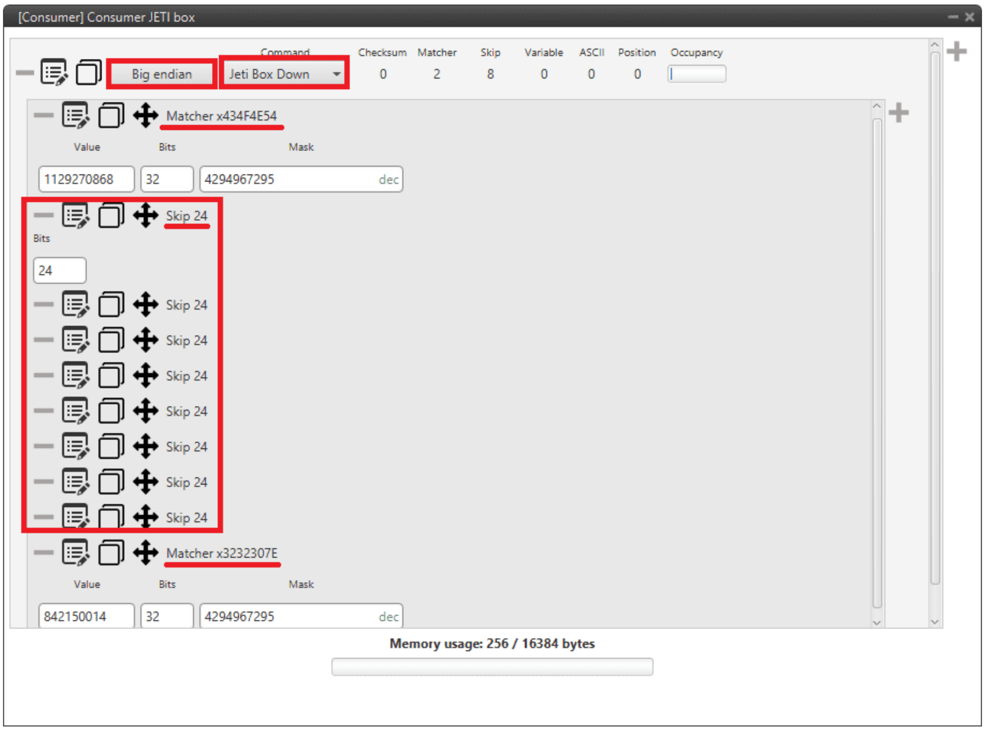
[Consumer] Consumer JETI box

	Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
Big endian	Jeti Box Down	0	2	8	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Nop	0	1	8	0	1	0	

Memory usage: 0 / 16384 bytes

JETI box custom message example

1. Expected text: "CONTROLLER TYPE MasterSpin 220~"
   - Command: **Jeti Box Down**
     - Matcher(32) "CONT" 0x434F4E54 (1129270868)
     - Skip(24*8) 192
     - Matcher(32) "220~" 0x3232307E (842150014)



[Consumer] Consumer JETI box

	Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
Big endian	Jeti Box Down	0	2	8	0	0	0	

Matcher x434F4E54

Value	Bits	Mask
1129270868	32	4294967295 dec

Skip 24

Bits: 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Skip 24

Matcher x3232307E

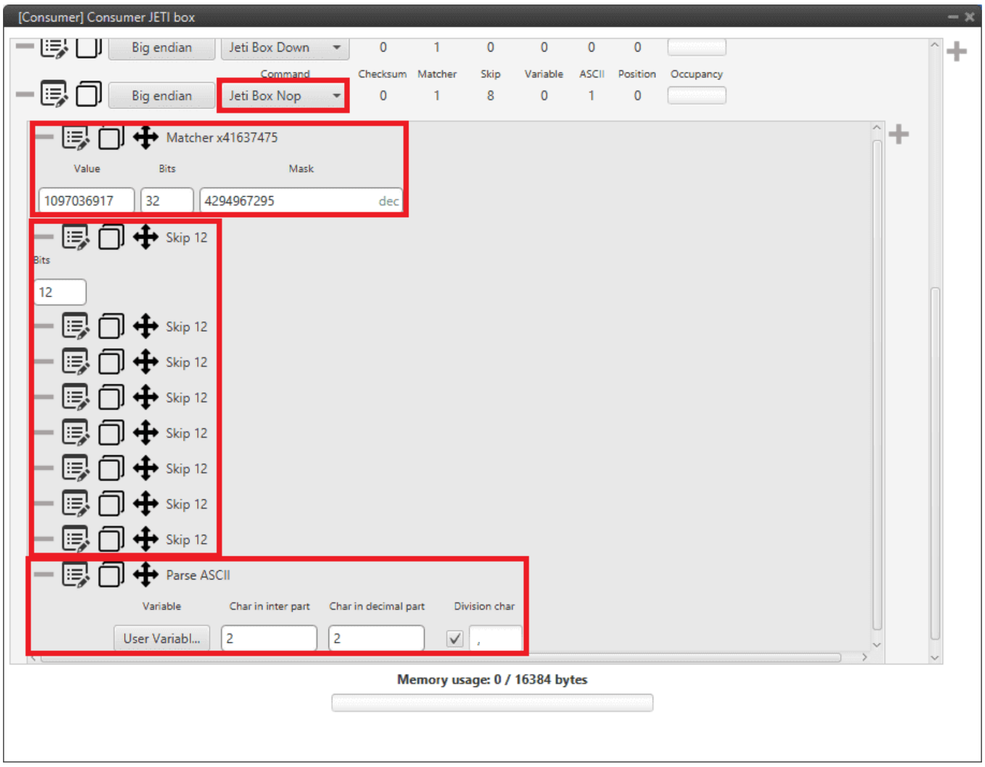
Value	Bits	Mask
842150014	32	4294967295 dec

Memory usage: 256 / 16384 bytes

JETI box first custom message example

2. Expected text: "MeasureOrSetting MEASURE ~"
   - Command: **Jeti Box Down**
     - Matcher(32) "Meas" 0x4D656173 (1298489715)
3. Expected text: "Max Temperature"...
   - Command: **Jeti Box Down**
     - Matcher(32) "Max " 0x4D617820 (1298233376)
4. Expected text: "Min Temperature"...
   - Command: **Jeti Box Down**
     - Matcher(32) "Min " 0x4D696E20 (1298755104)
5. Expected text: "Actual Temperature"...
   - Command: **Jeti Box Down**
     - Matcher(32) "Actu" 0x41637475 (1097036917)

6. Expected text: "MaxCurrent"...
   - Command: **Jeti Box Down**
     - Matcher(32) "MaxC" 0x4D617843 (1298233411)
7. Expected text: "MinCurrent"...
   - Command: **Jeti Box Down**
     - Matcher(32) "MinC" 0x4D696E43 (1298755139)
8. Expected text: "Max Voltage"...
   - Command: **Jeti Box Down**
     - Matcher(32) "Max " 0x4D617820 (1298233376)
9. Expected text: "Min Voltage"...
   - Command: **Jeti Box Down**
     - Matcher(32) "Min " 0x4D696E20 (1298755104)
10. Expected text: "Actual Voltage 11,86 V "
    - Command: **Jeti Box Nop**
      - Matcher(32) "Actu" 0x41637475 (1097036917)
      - Skip(12*8) 96
      - Parse ascii: int(2), decimal(2), separartor(',')



[Consumer] Consumer JETI box

	Command	Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
Big endian	Jeti Box Down	0	1	0	0	0	0	
Big endian	Jeti Box Nop	0	1	8	0	1	0	

Matcher x41637475

Value	Bits	Mask
1097036917	32	4294967295 dec

Skip 12

Bits: 12

Skip 12

Skip 12

Skip 12

Skip 12

Skip 12

Skip 12

Skip 12

Parse ASCII

Variable	Char in inter part	Char in decimal part	Division char
User Variabl...	2	2	☑ ,

Memory usage: 0 / 16384 bytes

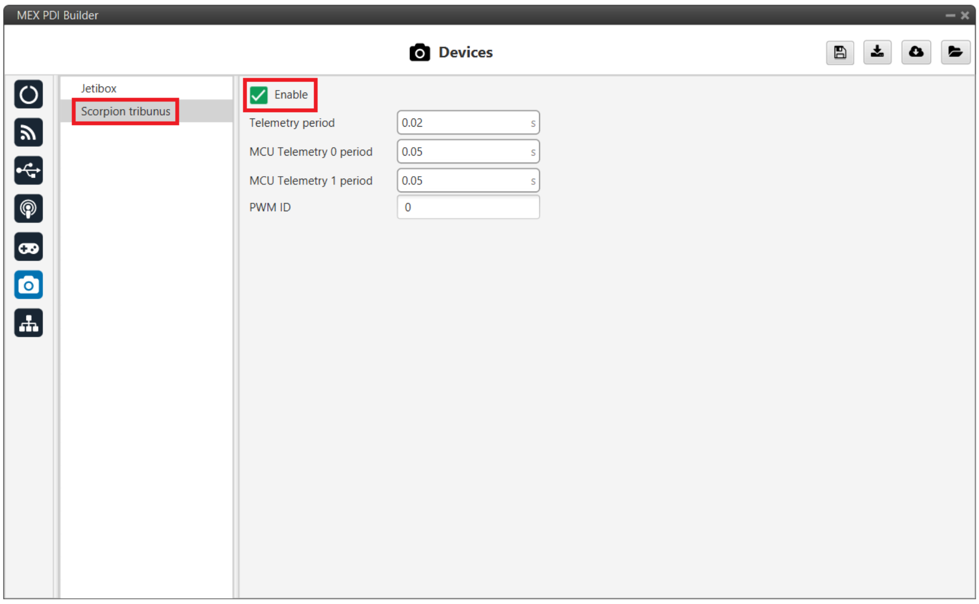
JETI box last custom message example

Scorpion tribunus

MEX is able to read telemetry from Tribunus ESCs by connecting it to one of its serial ports. The following steps configure **MEX** to read this telemetry:

1. Go to Devices menu → **Scorpion tribunus panel**.

   Enable it and configure it as shown:



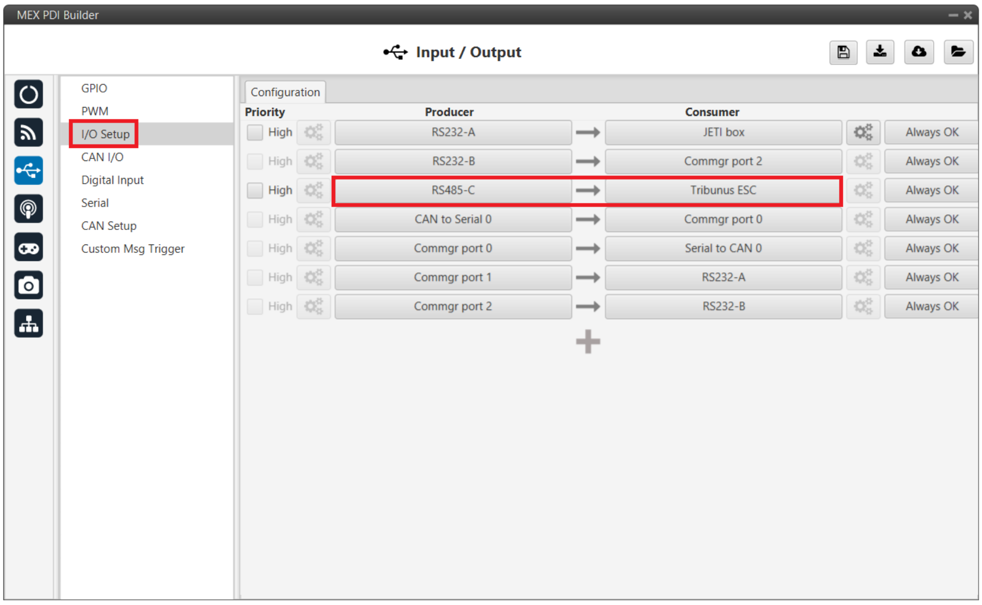
☑ Enable

Telemetry period	0.02 s
MCU Telemetry 0 period	0.05 s
MCU Telemetry 1 period	0.05 s
PWM ID	0

Scorpion tribunus - Devices configuration

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

   Link the specific Tribunus ESC consumer to the desired port:



Priority	Producer		Consumer	
□ High	RS232-A	→	JETI box	Always OK
□ High	RS232-B	→	Commgr port 2	Always OK
□ High	RS485-C	→	Tribunus ESC	Always OK
□ High	CAN to Serial 0	→	Commgr port 0	Always OK
□ High	Commgr port 0	→	Serial to CAN 0	Always OK
□ High	Commgr port 1	→	RS232-A	Always OK
□ High	Commgr port 2	→	RS232-B	Always OK

Scorpion tribunus - I/O Setup configuration

Veronte products

Connection with Autopilot 1x via CAN

No configuration is necessary for **MEX**, the configuration for communication with **Autopilot 1x** is set by default.

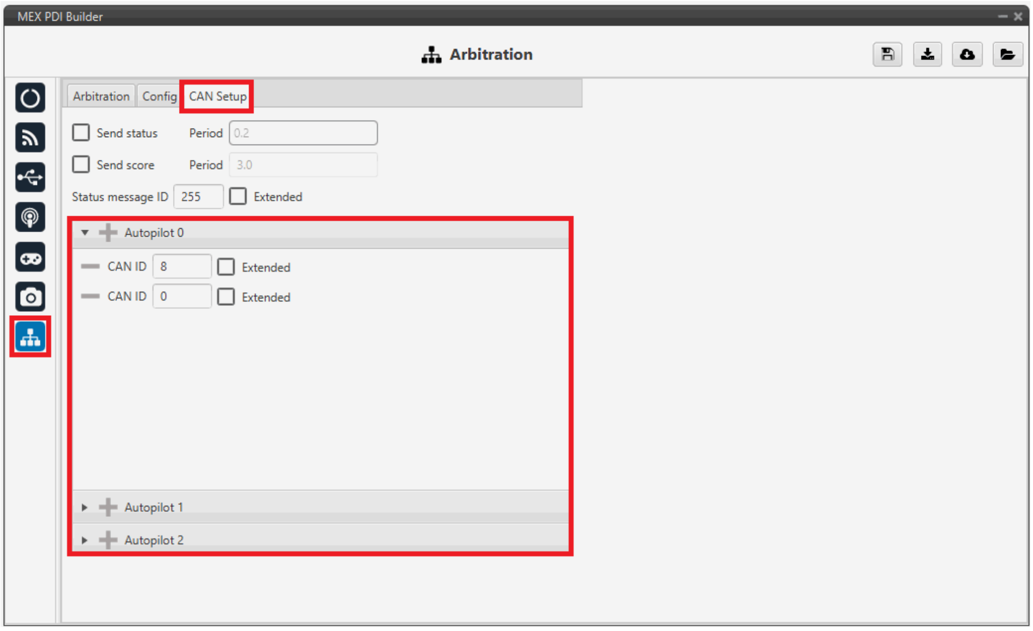
The configuration required in **1x PDI Builder** to communicate with **MEX** via CAN is explained in the CEX/MEX - Integration examples section of the **1x PDI Builder** user manual.

Connection with Autopilot 4x via CAN

CAN Reception IDs

First of all, users can setup the receiving CAN Ids for each one of the three possible **Veronte Autopilots 1x** sending data to **MEX** in the **Arbitration menu**.

Go to Arbitration menu → **CAN Setup panel**:



□ Send status Period 0.2
□ Send score Period 3.0
Status message ID 255 □ Extended
Autopilot 0
CAN ID 8 □ Extended
CAN ID 0 □ Extended
Autopilot 1
Autopilot 2

Arbitration - CAN Reception IDs

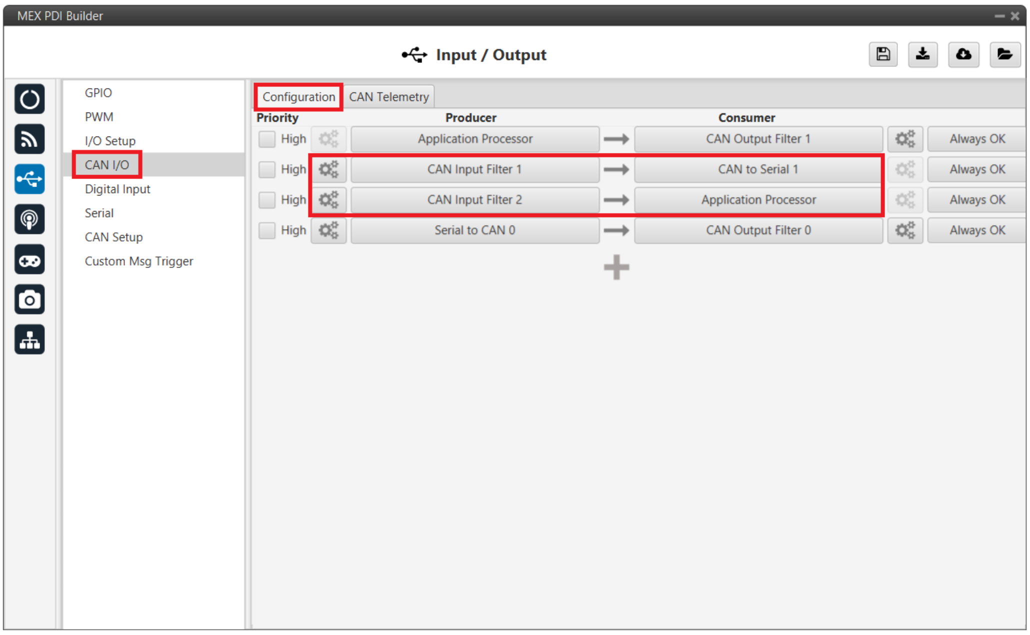
> ⓘ **Note**
>
> If **arbitration is not enabled**, and therefore only an **Autopilot 1x** is being used, no CAN Ids need to be configured here.

CAN I/O Interconnections

Once CAN IDs are set, users shall configure:

- **CAN Input Filters** to be used (as communication with **MEX** has to be **always** through a **Filter**).
- The connection between input filters and data Consumers.

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

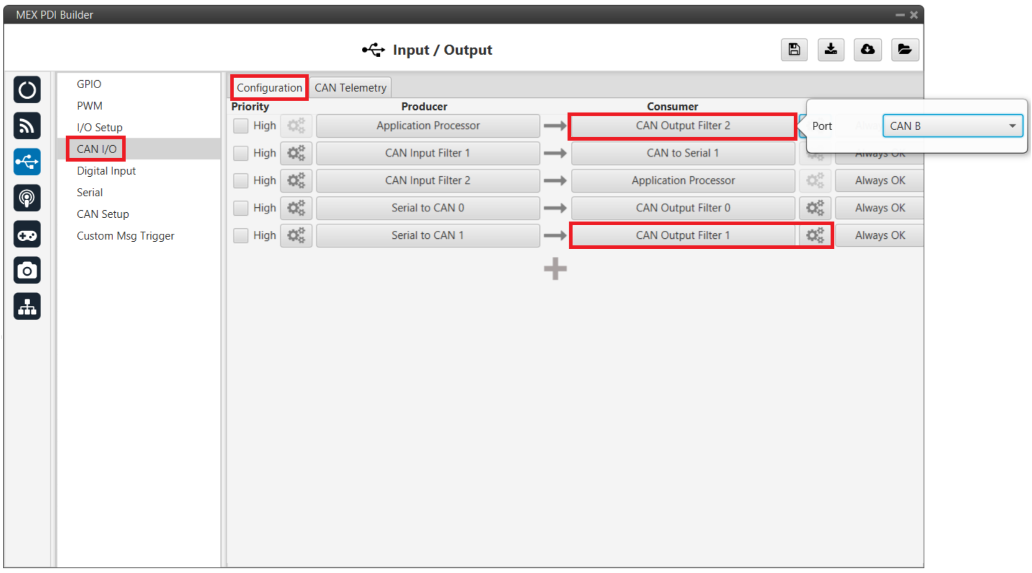


Priority	Producer		Consumer	
☐ High	Application Processor	⟶	CAN Output Filter 1	Always OK
☐ High	CAN Input Filter 1	⟶	CAN to Serial 1	Always OK
☐ High	CAN Input Filter 2	⟶	Application Processor	Always OK
☐ High	Serial to CAN 0	⟶	CAN Output Filter 0	Always OK

Arbitration - CAN Input Filters

For more information on CAN I/O configuration, see the CAN I/O - Input/Output section of the present manual.

Next step is to connect each of the desired data Producers to a **CAN Output Filter**, and configure both, the Producer and the CAN Output Filter:



Priority	Producer	Consumer	
High	Application Processor	CAN Output Filter 2	
High	CAN Input Filter 1	CAN to Serial 1	Always OK
High	CAN Input Filter 2	Application Processor	Always OK
High	Serial to CAN 0	CAN Output Filter 0	Always OK
High	Serial to CAN 1	CAN Output Filter 1	Always OK

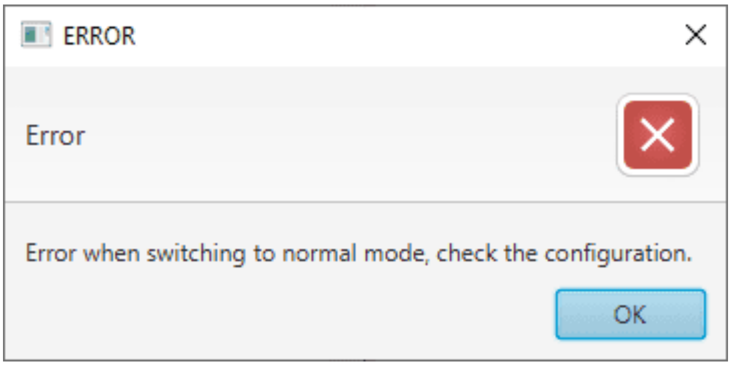
Port: CAN B

Arbitration - CAN Output Filters

Troubleshooting

Maintenance mode (loaded with errors)

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

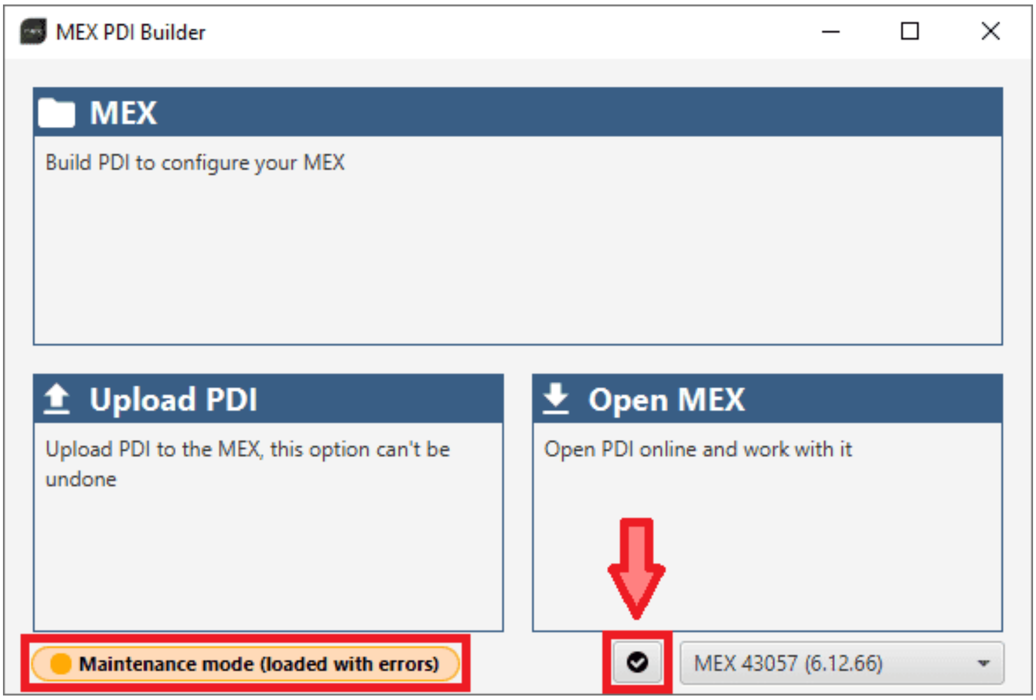


Error

Error when switching to normal mode, check the configuration.

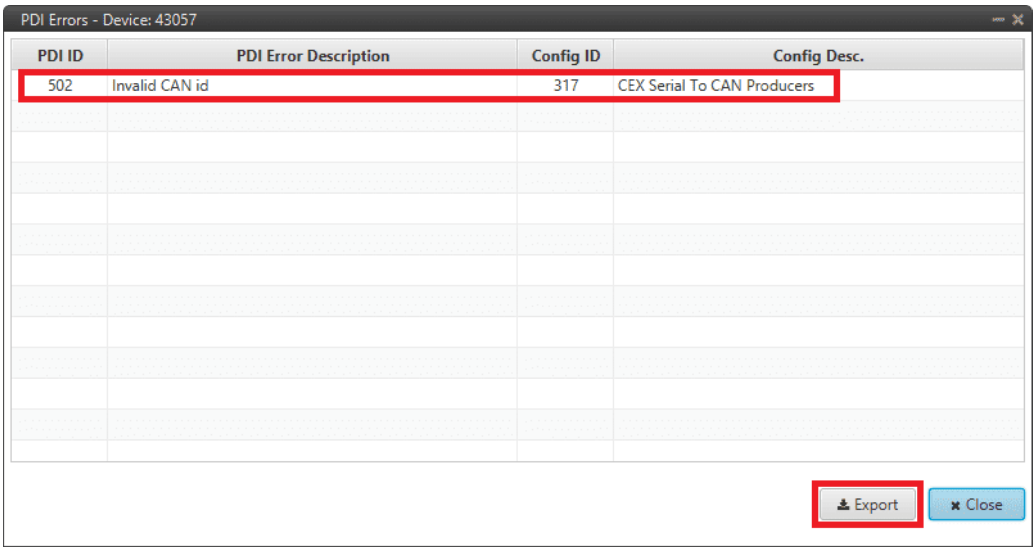
Error message

Therefore, **MEX** unit will be in 'Maintenance mode (loaded with errors)':


MEX PDI Builder
MEX
Build PDI to configure your MEX
Upload PDI
Upload PDI to the MEX, this option can't be undone
Open MEX
Open PDI online and work with it
Maintenance mode (loaded with errors)
MEX 43057 (6.12.66)

Maintenance mode (loaded with errors)

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:


PDI Errors - Device: 43057

PDI ID	PDI Error Description	Config ID	Config Desc.
502	Invalid CAN id	317	CEX Serial To CAN Producers

Maintenance mode (loaded with errors) - PDI Errors panel

- **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 **1x 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.

Finally, it is possible to access the **MEX** configuration to fix this error.

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**.

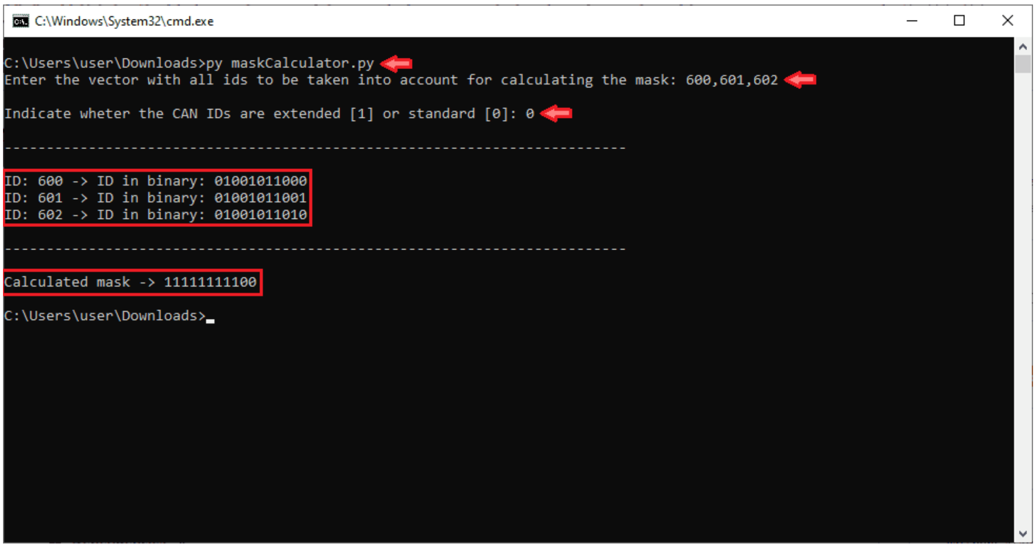
FAQ

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:\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>
```

Example of maskCalculator program