MC01B PDI Builder

Release 6.8.67

Embention

2023-08-30

CONTENTS

MC01B PDI Builder is an application employed to configure the MC01B, the brushless variant of MC01.

CHAPTER

ONE

QUICK START

MC01B PDI Builder is the main tool for setting all configurable parameters of the MC01B.

Configuration for Veronte MC01B should be created according to the needs of the belonging system.

Once MC01B has been detected on Veronte Link, download and install **MC01B PDI Builder**.

1.1 Download

Once a MC01B device has been purchased, a GitHub release should be created for the customer with the application.

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

1.2 Installation

To install **MC01B PDI Builder** on Windows, execute MC01BPDIBuilder.exe and follow the steps:

1. Click on **Next**:

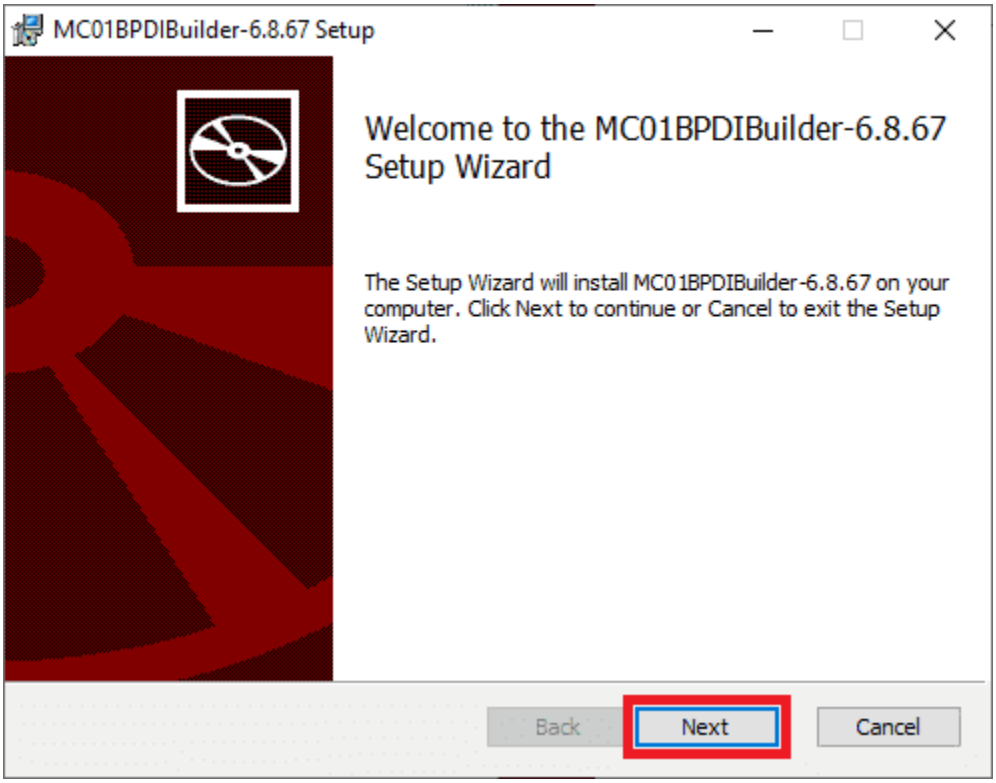


Fig. 1: **Windows Installation - Step 1**

2. Select the directory where where the software will be installed (with the **Change** button), then click on **Next**:

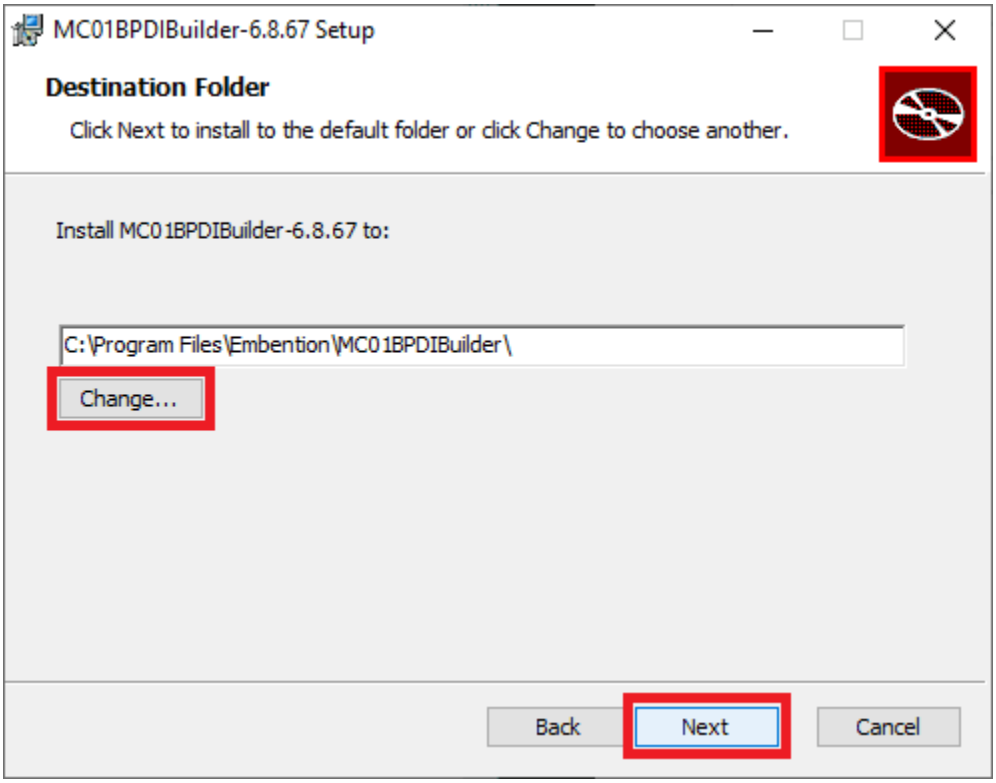


Fig. 2: **Windows Installation - Step 2**

3. Finally, click on **Install** (administrator rights are needed):

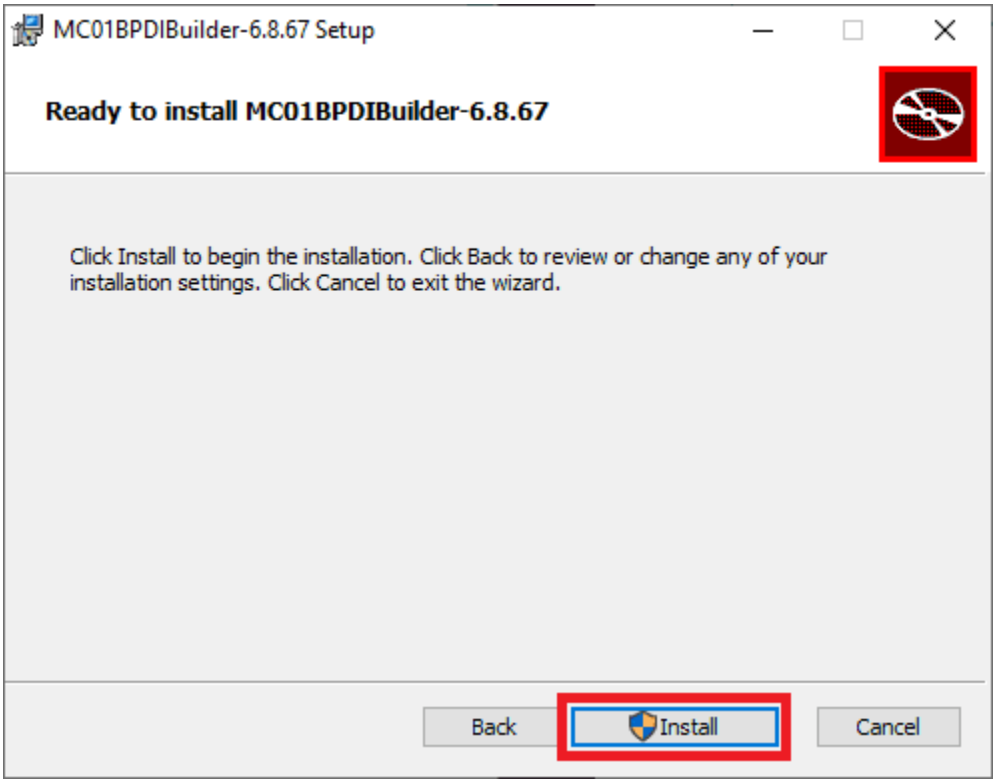


Fig. 3: **Windows Installation - Step 3**

4. After a few seconds, the following window will appear indicating the process was succesful. Click on **Finish**:

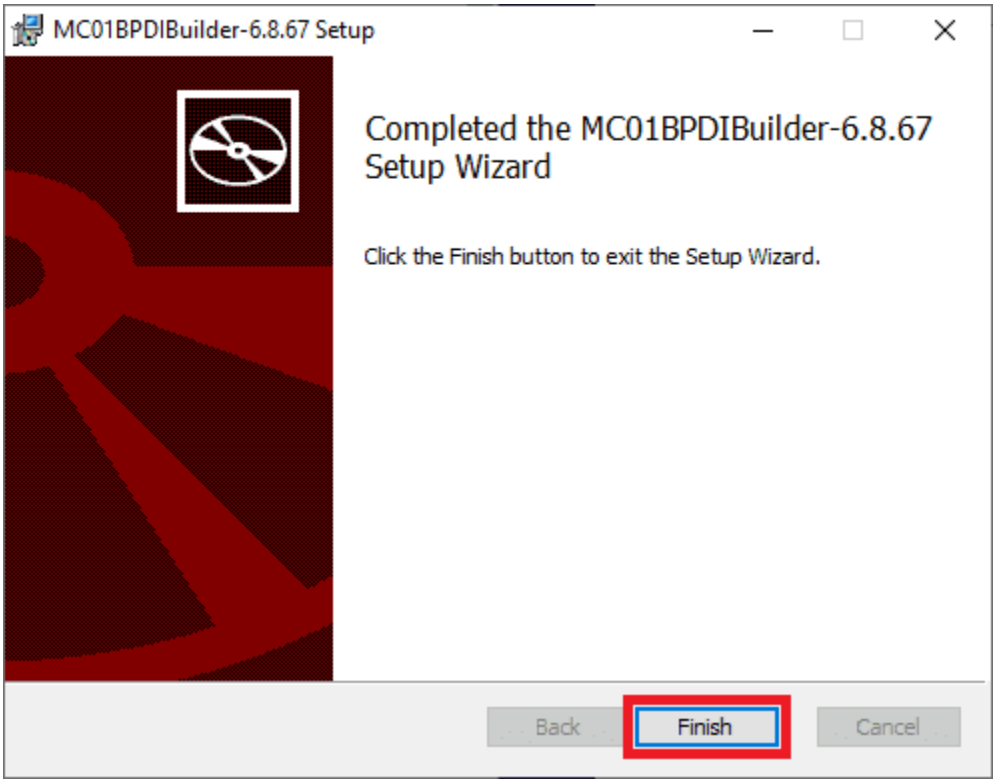
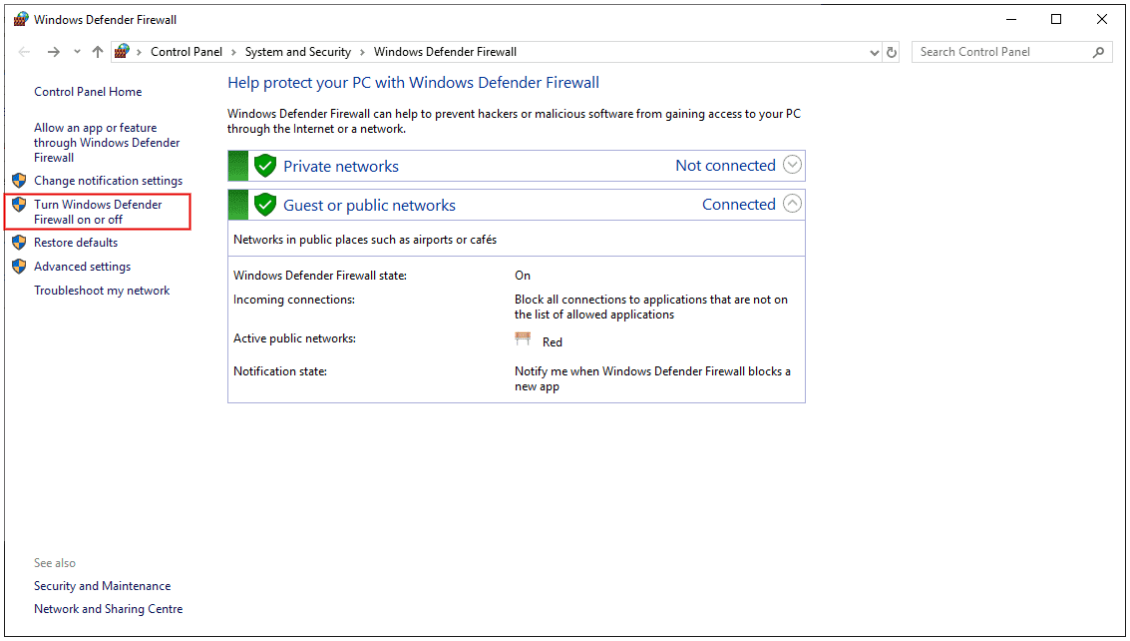
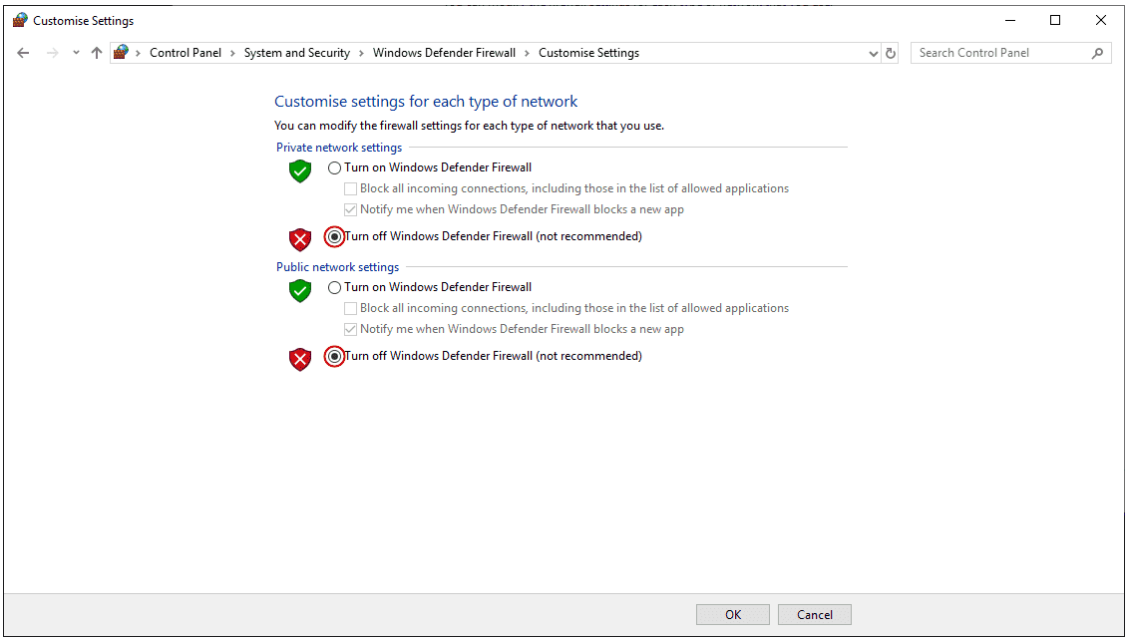


Fig. 4: **Windows Installation - Step 4**

Note: In case of any issue during installation, please disable Windows Defender and Firewall. To disable Firewall, go to "Control Panel" and "Firewall of windows", then click on **Turn off**.




Customise Settings
Control Panel › System and Security › Windows Defender Firewall › Customise Settings
Search Control Panel
Customise settings for each type of network
You can modify the firewall settings for each type of network that you use.
Private network settings
Turn on Windows Defender Firewall
Block all incoming connections, including those in the list of allowed applications
Notify me when Windows Defender Firewall blocks a new app
Turn off Windows Defender Firewall (not recommended)
Public network settings
Turn on Windows Defender Firewall
Block all incoming connections, including those in the list of allowed applications
Notify me when Windows Defender Firewall blocks a new app
Turn off Windows Defender Firewall (not recommended)
OK
Cancel

CHAPTER

TWO

CONFIGURATION

This section explains each option and parameter available in MC01B PDI Builder.

Once the installation is finished, open **MC01B PDI Builder** and select the unit:

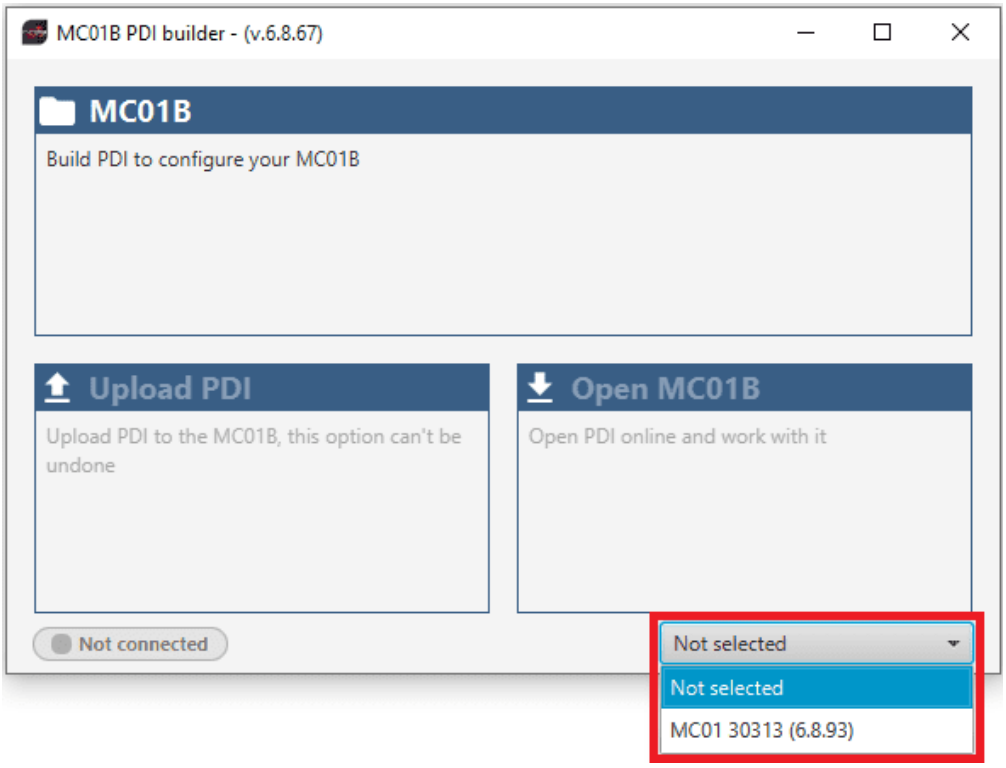


Fig. 1: **MC01B ID**

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

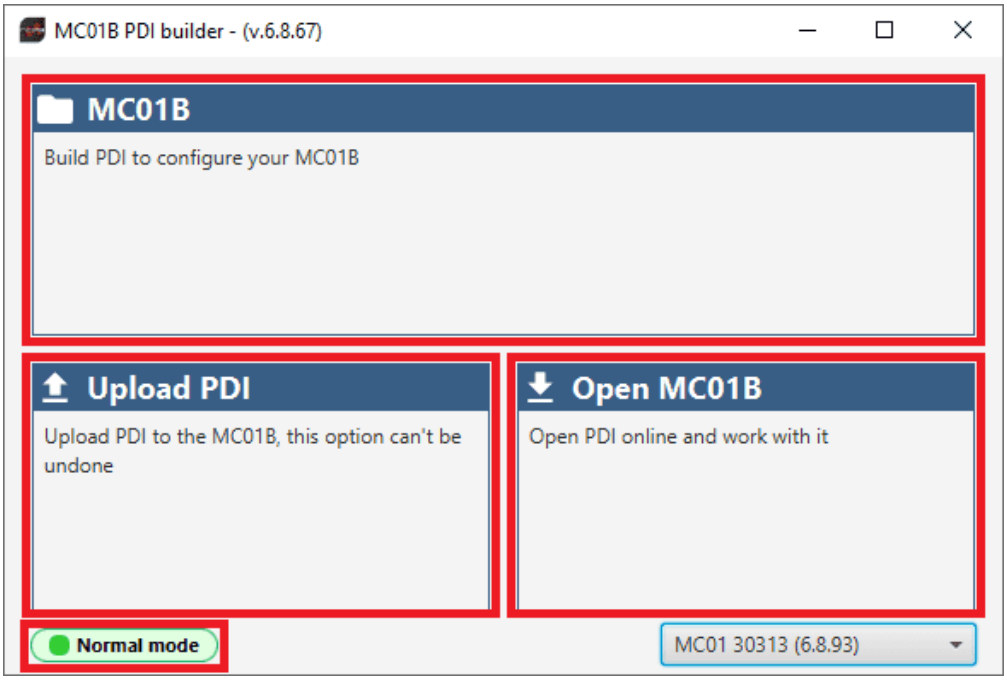


Fig. 2: **MC01B PDI Builder**

MC01B unit can also appears as: Maintenance mode (loaded with errors) or Normal mode - Disconnected.

Note: **Maintenance mode (loaded with errors)** appears when something is wrong in the configuration.

The user can access now to 3 configuration options:

- **MC01B**: It allows the user to work with **offline** configurations. A previously exported MC01B 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 **MC01B PDI configuration** can be imported to the current **MC01B** flash memory.
- **Open MC01B**: By clicking on this option, **MC01B PDI Builder** configuration menu opens with the configuration (the PDI files) loaded in the MC01B. Then, the user can modify it online.

Note: PDI files are MC01B configuration files. These files allow for modular control with improved version management. These PDI files are split in 2 folders. Each folder hold several .xml files:

- **vmc-brushless**: Contains configuration for MC01B. All control system and parameters are stored here.
- **xsd**: This folder holds .xsd files. An XSD file is a definition file specifying the elements and attributes that can be part of an XML document. This ensures that data is properly interpreted, and errors are caught, resulting in appropriate XML validation. **Users should never delete, replace or modify it**.

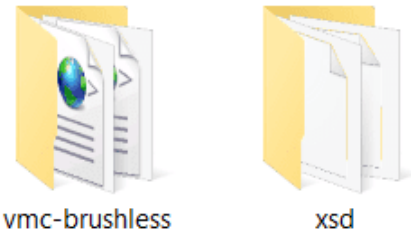


Fig. 3: **PDI files**

Finally, click on '**Open MC01B**' to open the configuration and start editing online.

Fig. 4: **Open MC01B**

Note: When MC01B PDI is open, the unit changes to **Maintenance mode**.

The different 'buttons' that can be seen in the initial menu of the **MC01B PDI Builder** are explained below.

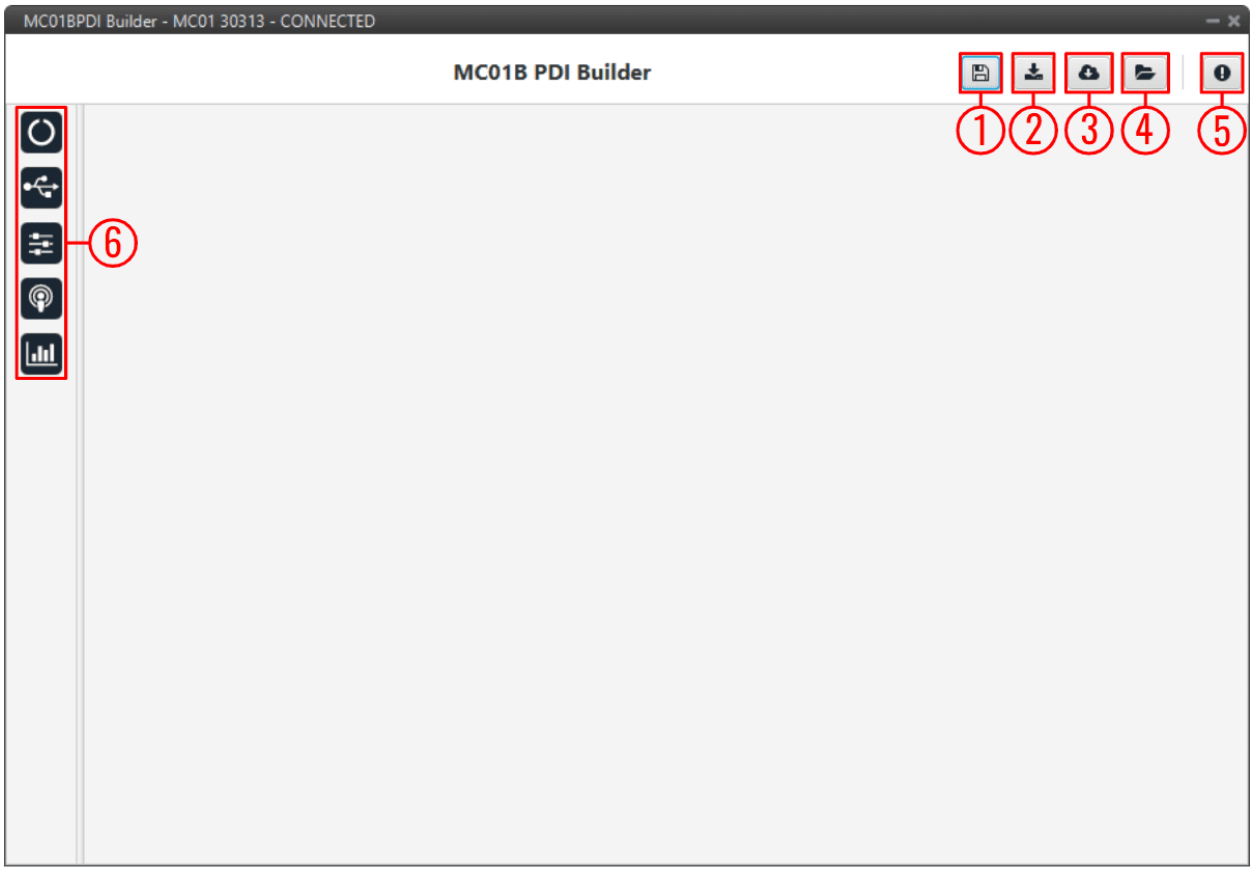


Fig. 5: **Initial menu**

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

   Note: This button will only appear if a MC01B 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 repo and modify it. After that, if the save button is pressed, this configuration will be uploaded to the connected MC01B.

4. **Import PDI from local storage**: The user can import a configuration file from the local storage and modify it. After that, if the save button is pressed, this configuration will be loaded into the connected MC01B.

5. **Feedback**: Users can report a problem they have encountered by **creating an issue in their own 'Joint Collaboration Framework'**. The 'Download' button downloads a zipped folder with the current MC01B configuration and more information needed for Embention to resolve the issue. It is advisable to attach this folder when creating the issue.

   Note: The user's '**Joint Collaboration Framework**' is simply a **own Github repository for each customer**.

If the user has any questions about this Joint Collaboration Framework, please see Joint Collaboration Framework user manual.

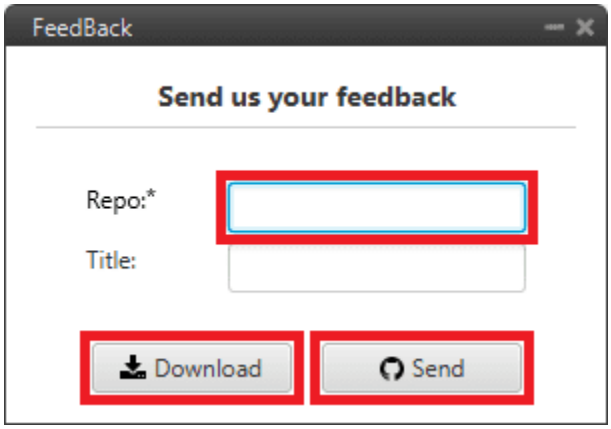


Fig. 6: **Feedback window**

6. These are the different functions of MC01B. They are explained in the following sections:
   - *VMC Brushless*
   - *Input/Output*
   - *Control*
   - *Communications*
   - *Telemetry*

2.1 VMC Brushless

2.1.1 Motor

This section sets the electrical parameters of the motor that will define the dynamics of the controller.

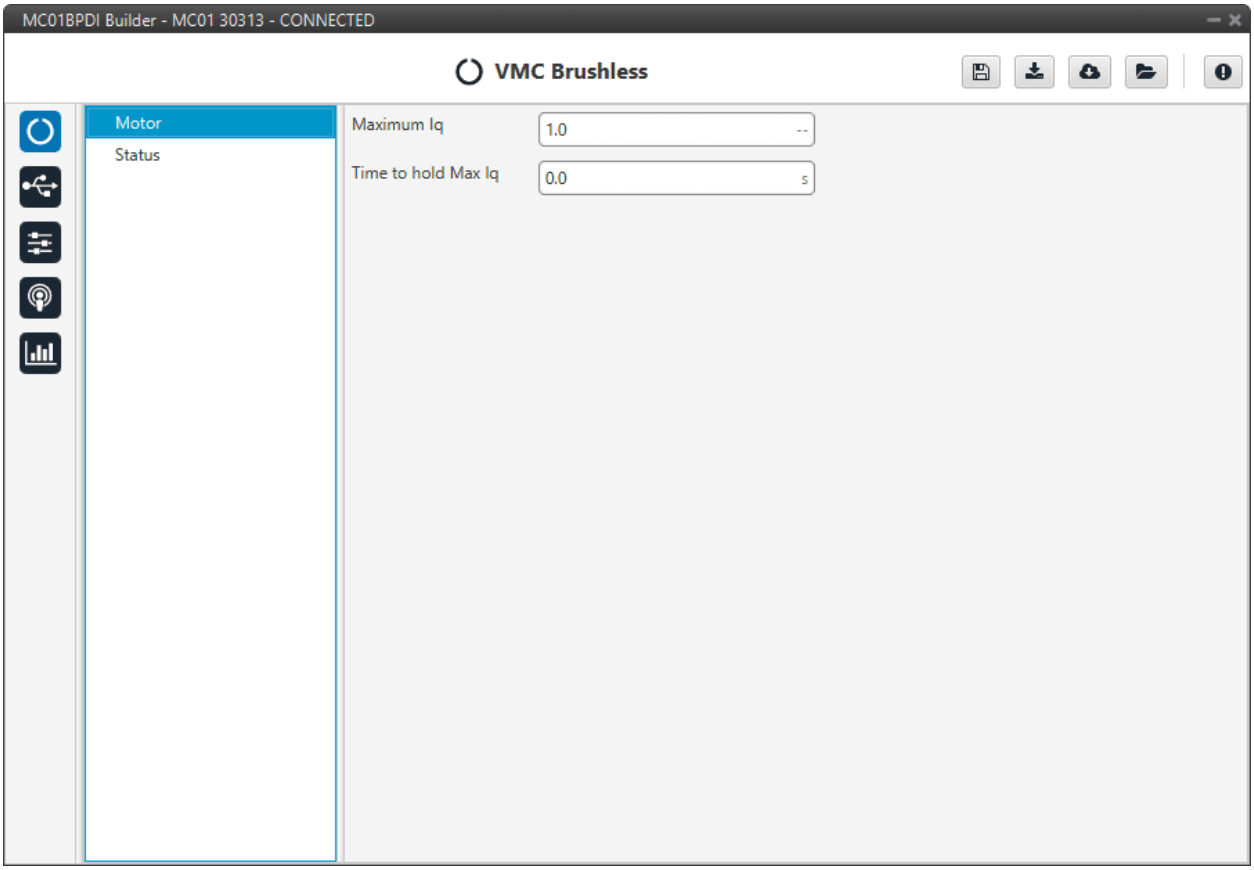


Fig. 7: **Motor section**

- **Maximum Iq**: Maximum permissible quadrature current that the motor can drive. It is a normalized value from *0A* to *3.75A*.
- **Time to hold Max Iq**: Defines the time while the maximum intensity (*Maximum Iq*) can be overpassed.

Warning: In case of overpassing the *Maximum Iq* during the stablished period, the MC01 will shut down.

2.1.2 Status

- **Enable VCP Status Message** enables the periodic sending of the status message that Veronte Link uses to recognise the Veronte MC01B.
- **Period**: Enter a desired period to send repeatedly the status message.

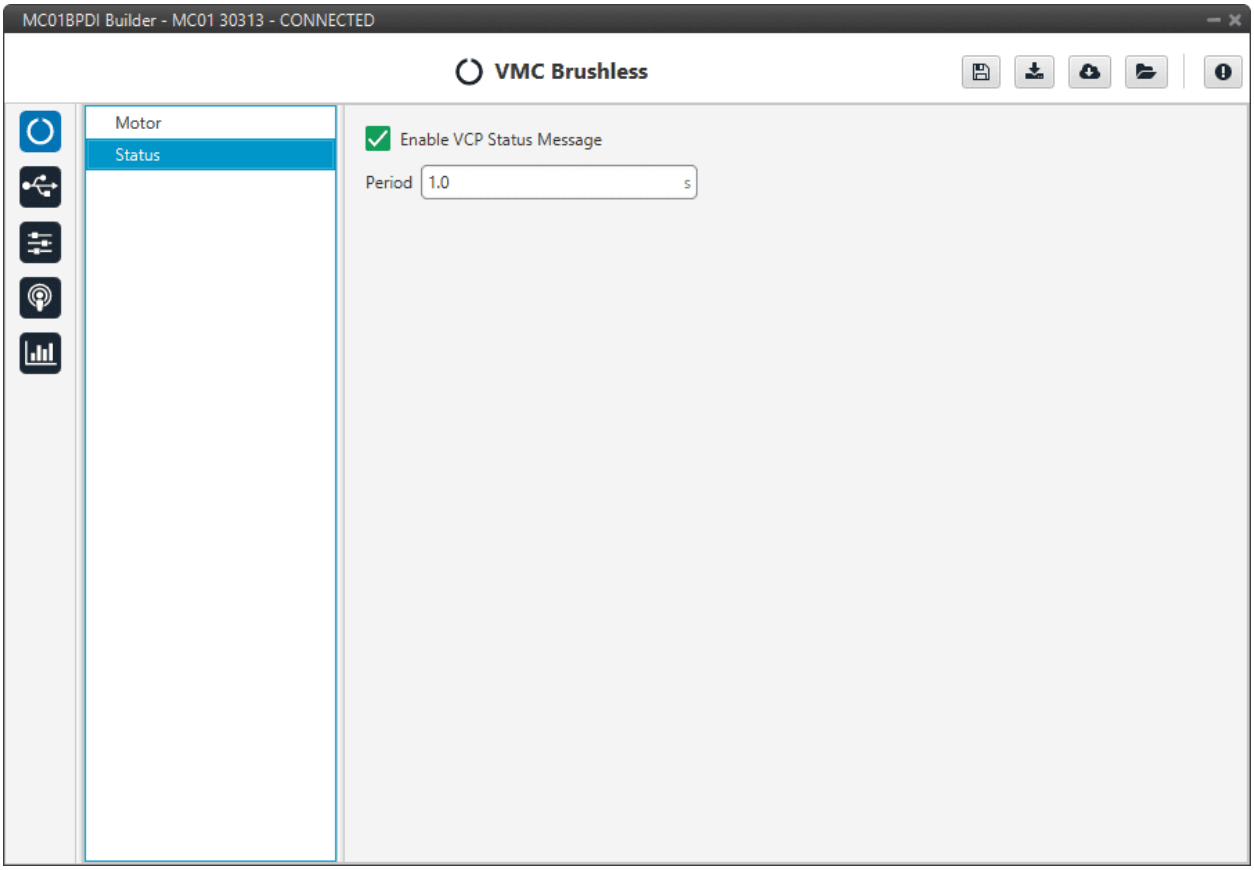


Fig. 8: **Status section**

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

2.2 Input/Output

MC01B can send and receive messages through a CAN bus with Serial protocol, this communication system is named as Serial-Over-CAN. This is necessary to establish the communication with a Veronte Autopilot 1x.

Note: **TX** corresponds to transmitted messages and **RX** to received messages.

First of all, it is necessary to set the appropriate **Baudrate** (i.e. the communication speed, expressed in bits per second) of the CAN bus.

2.2.1 SCIA Serial CAN

Serial messages received/sent through the SCIA (RS-232) port can be sent via CAN.

This menu internally configures the following connection:

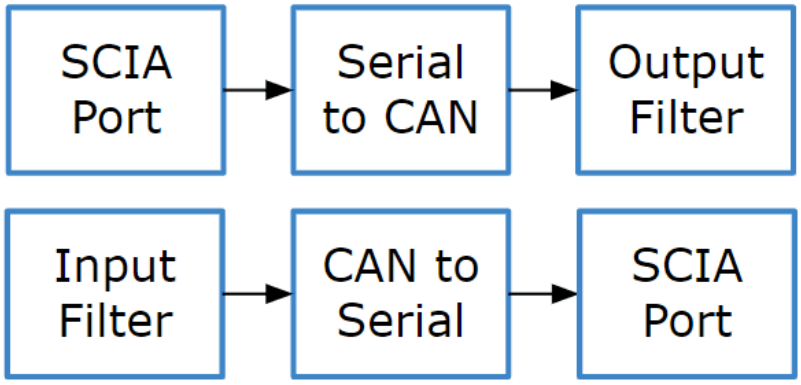


Fig. 9: **Input/Output menu - SCIA Serial CAN schema**

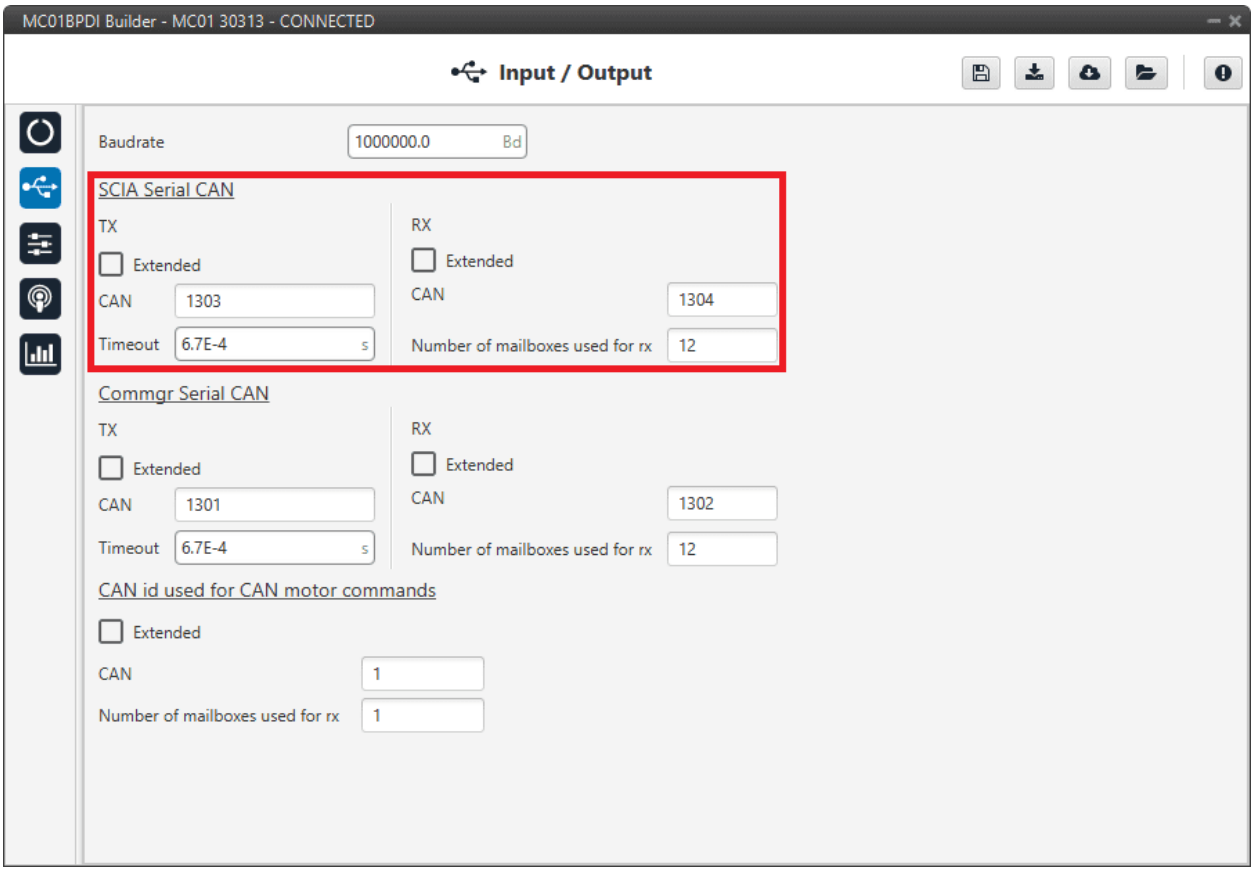


Fig. 10: **Input/Output menu - SCIA Serial CAN section**

- **TX**:

  - **Extended**: If enabled, the frame format will be this, 'Extended', i.e. with a 29-bit identifier. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
  - **CAN**: CAN Id of the message to be sent must be set. The value set has to be decimal format.
  - **Timeout**: This is the threshold time between receptions to consider that it is not being received correctly.
- **RX**: Configuration for receiving messages (as Mailboxes in the Veronte Autopilot 1x configuration)
  - **Extended**: Select this option for 29-bit IDs.
  - **CAN**: 11-bits (Standard) or 29-bits (Extended) ID used to identify the CAN message to be received. The value set has to be decimal format.
  - **Number of mailboxes used for rx**.

2.2.2 Commgr Serial CAN

Serial messages received/sent through the Commgr port can be sent via CAN.

Warning: This configuration is already done by default in order to allow communication via CAN between 1x Autopilot and MC01B. **It must not be modified**.

This menu internally configures the following connection:

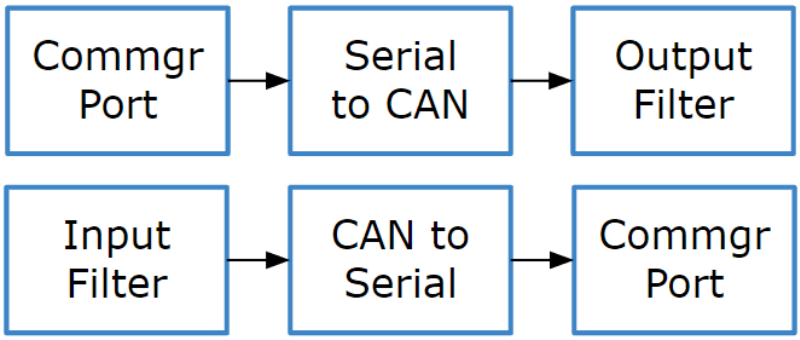


Fig. 11: **Input/Output menu - Commgr Serial CAN schema**

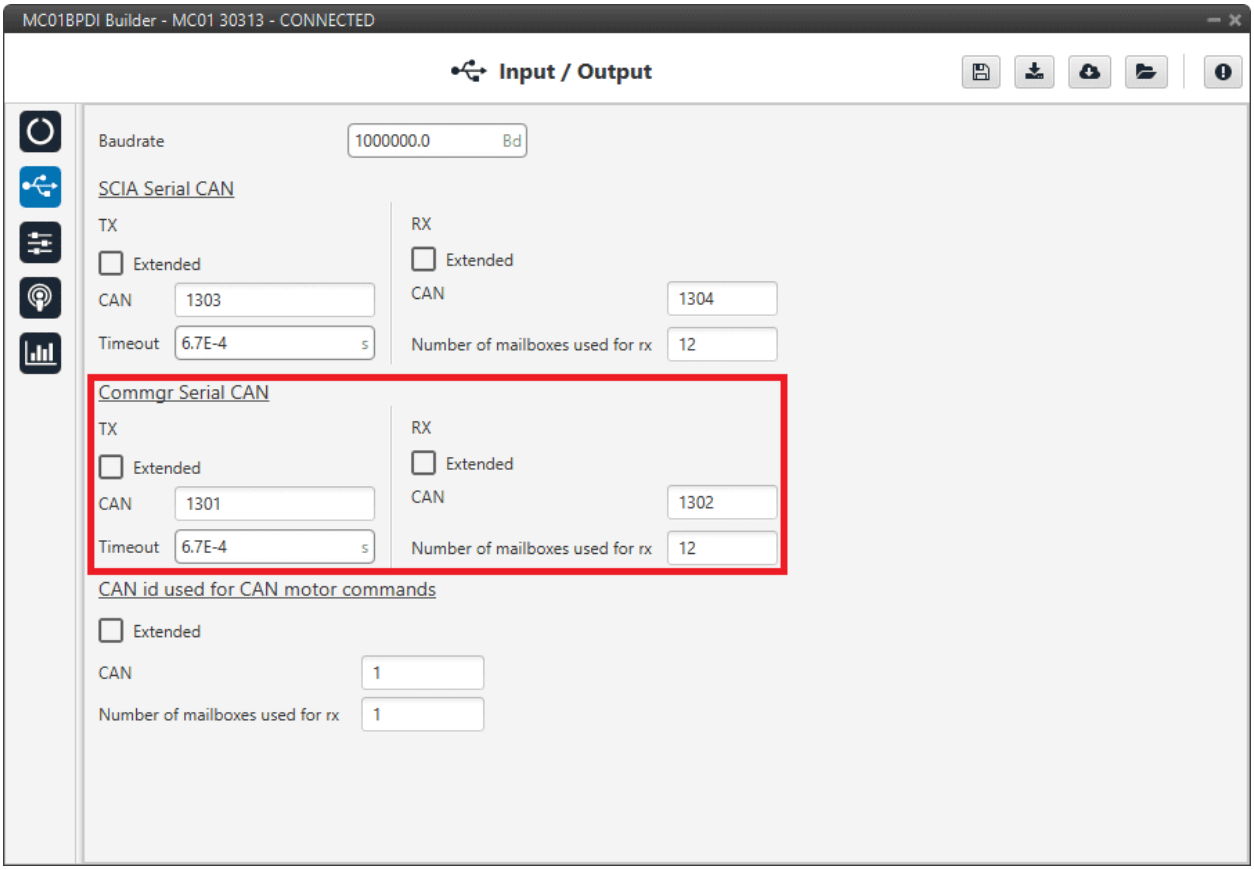


Fig. 12: **Input/Output menu - Commgr Serial CAN section**

- **TX**:
  - **Extended**: If enabled, the frame format will be this, 'Extended', i.e. with a 29-bit identifier. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
  - **CAN**: CAN Id of the message to be sent must be set. The value set has to be decimal format.
  - **Timeout**: This is the threshold time between receptions to consider that it is not being received correctly.
- **RX**: Configuration for receiving messages (as Mailboxes in the Veronte Autopilot 1x configuration)
  - **Extended**: Select this option for 29-bit IDs.
  - **CAN**: 11-bits (Standard) or 29-bits (Extended) ID used to identify the CAN message to be received. The value set has to be decimal format.
  - **Number of mailboxes used for rx**.

2.2.3 CAN id used for CAN motor commands

MC01B requires a CAN id to send the desired position to the motor.

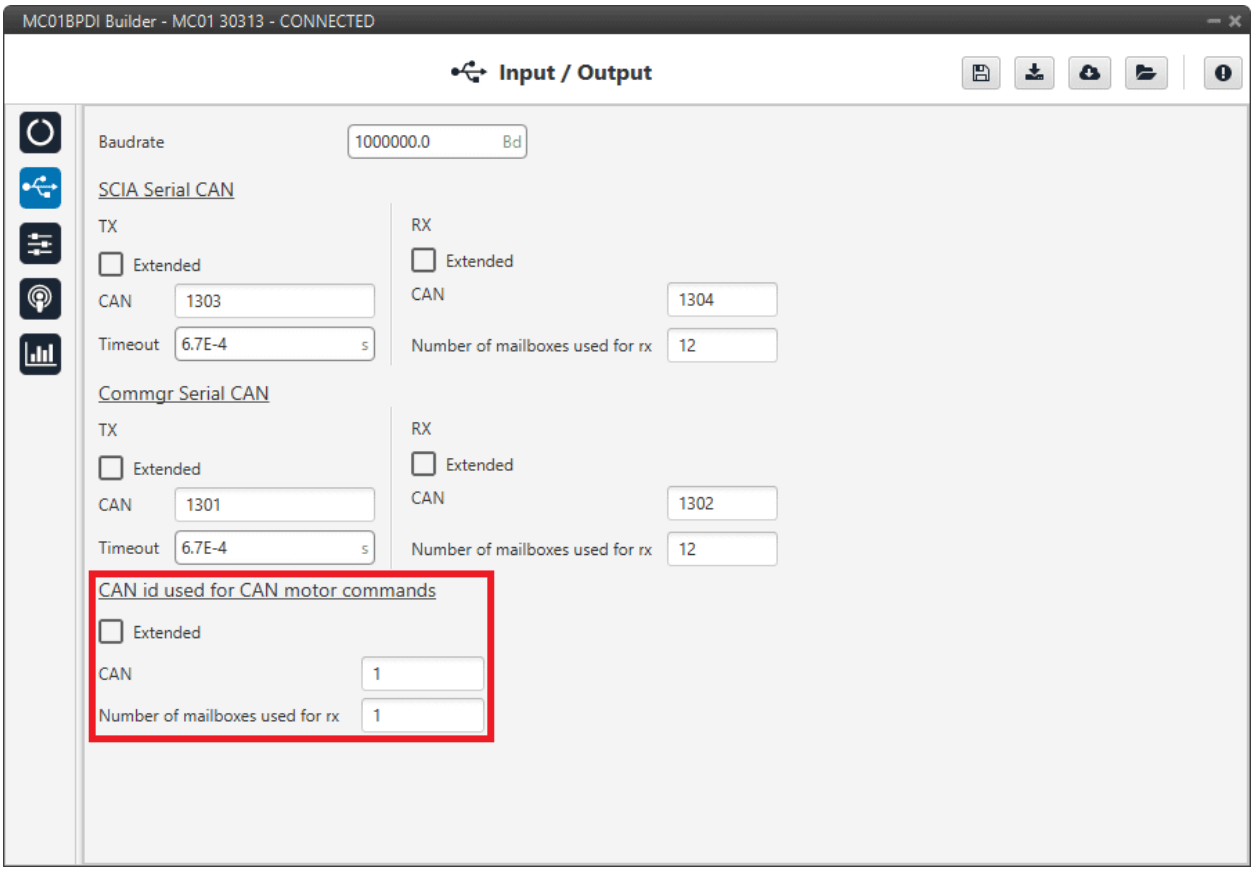


Fig. 13: **Input/Output menu - CAN id used for CAN motor commands section**

- **Extended**: If enabled, the frame format will be this, 'Extended', i.e. with a 29-bit identifier. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
- **CAN**: CAN Id through which the MC01B will send the position of the motor.
- **Number of mailboxes for rx**.

2.3 Control

2.3.1 Control Brushless

First, the **PI** (Proportional and Integral) controller is presented.

The form of the PI is the classical parallel form:

$$u = K_p \cdot \left(1 + \frac{1}{T_i} \cdot s\right)$$

Where:

- K_p is the **Proportional loop gain**.
- **Integral gain** refers to the quotient $\frac{1}{T_i}$.
- **Lower** and **Upper saturation gain** are the **limits** to which the PI limit its output.

Control Brushless is the main control menu, where users must enter the control parameters according to their motor.

The basic blocks that define the control are four *PI control loops* that should be tuned with the motor characterization.

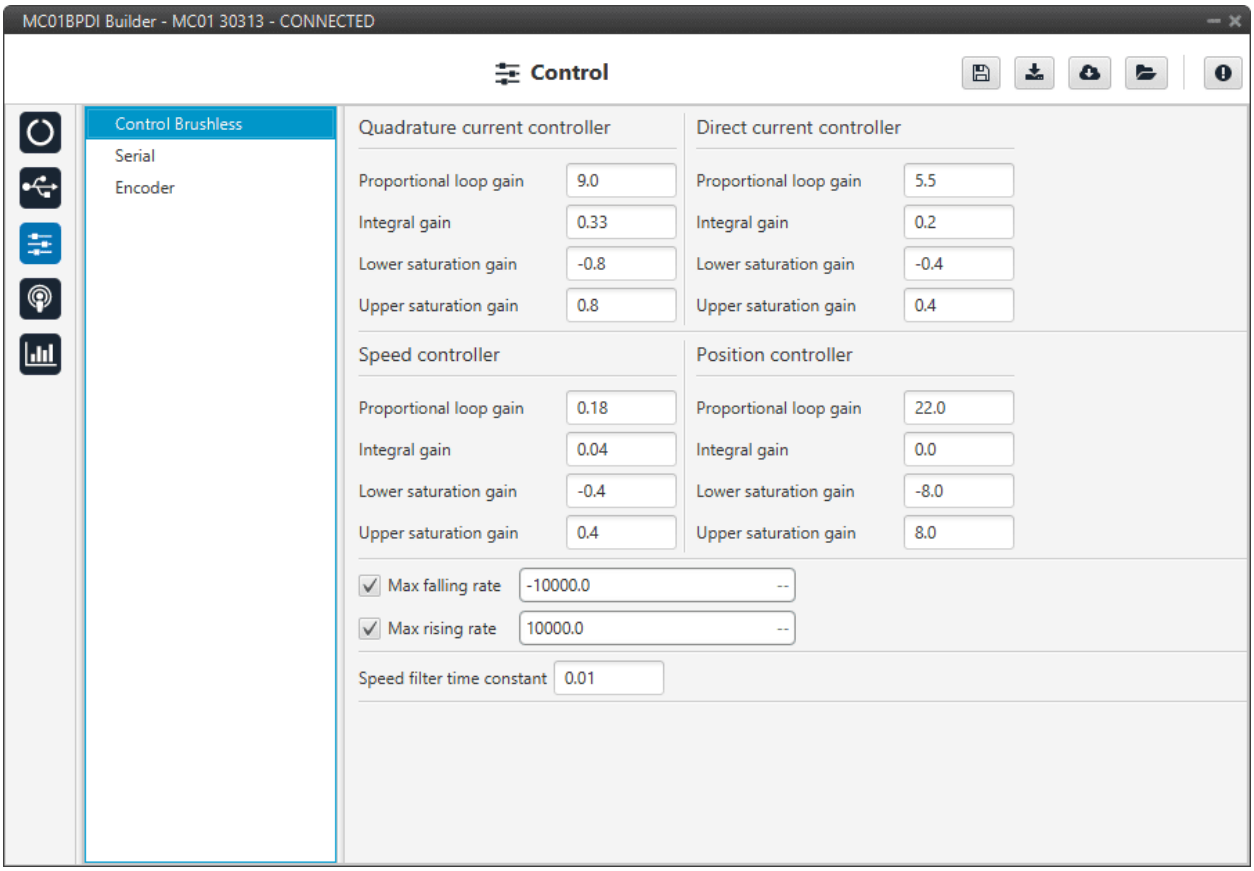


Fig. 14: **Control Brushless section**

Both quadrature and direct current PI control must be defined. Also, the quadrature and direct PI gain settings are usually the same, but it will depend on whether the motor parameters are the same on both axes.

- **Quadrature current controller**: Torque regulator. It is controlled by a *PI control*.
- **Direct current controller**: Flux regulator. This is controlled by a *PI control*.

- **Speed controller**: It is controlled by a *PI control*.
- **Position controller**: It is controlled by a *PI control*.

Maximum desired motor acceleration:

- **Max falling rate**: Deceleration rate. Expressed in rad/s^2.
- **Max rising rate**: Acceleration rate. Expressed in rad/s^2.

- **Speed filter time constant**: This is the time constant of the first order filter of the speed calculation. **The lower value, the lower filtering**.

  The time constant is approximately defined as:

$$\tau = \frac{1}{2\,\pi\,f_c}$$

  Where:

  - τ: time constant
  - f_c : cutoff frequency

2.3.2 Serial

MC01B can only use one serial peripheral, SCIA port, the serial parameters that fit the serial protocol requirements can be edited in this menu:

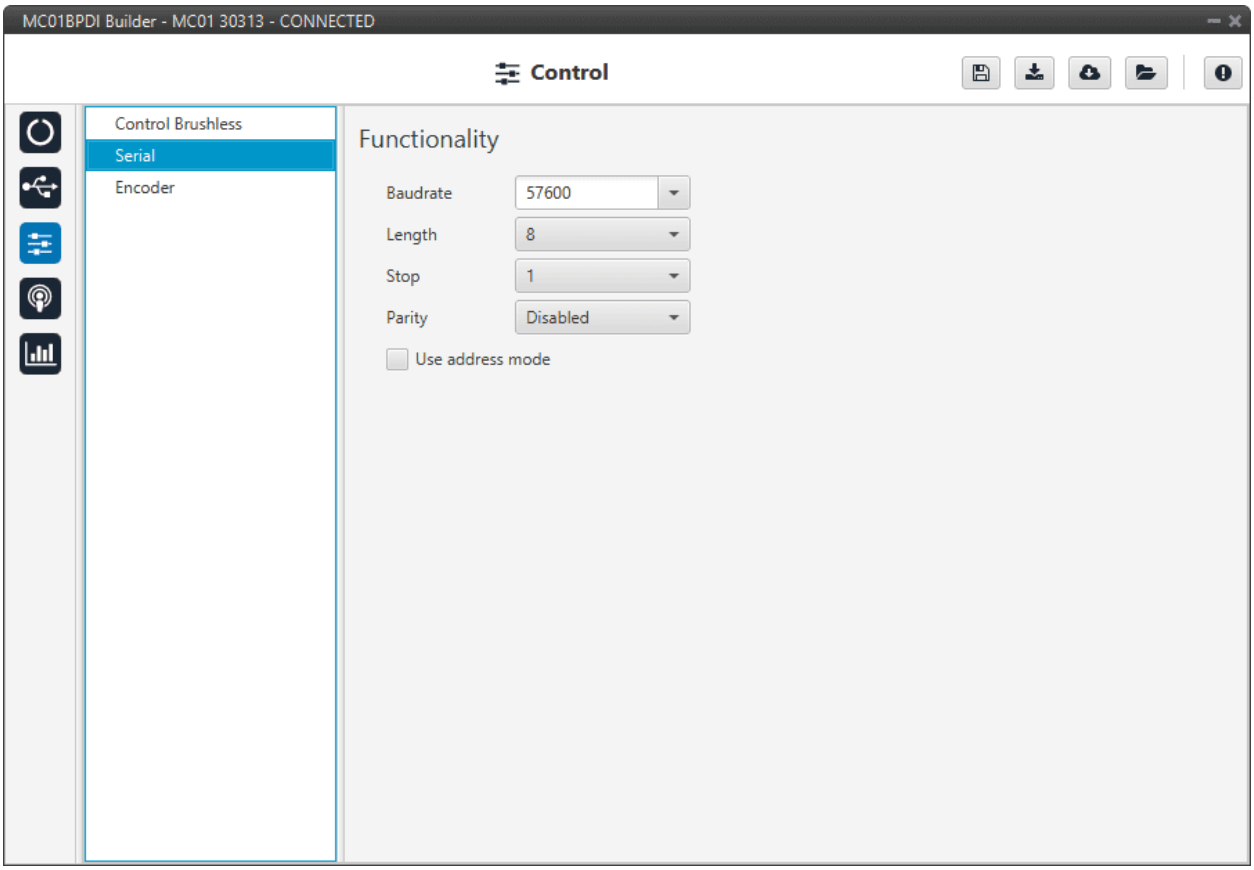


Fig. 15: **Serial section**

- **Baudrate**: This field specifies how fast data is sent over a serial line.
- **Length**: Defines the 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 will be sent with each data character.

  The bits of each character (including parity bit) will be even or odd according to **parity** mode (**odd**, **even** or **disabled**).

- **Use address mode**: 9-bit data framing uses the bit typically associated with parity error detection to identify address messages. Sent serial data that does not have the address bit set will be ignored (unless the device had previously identified an address message associated with it).

  This option can be disabled or enabled.

2.3.3 Encoder

In this section, the user must define the number of bits employed by the encoder to communicate the position of the motor.

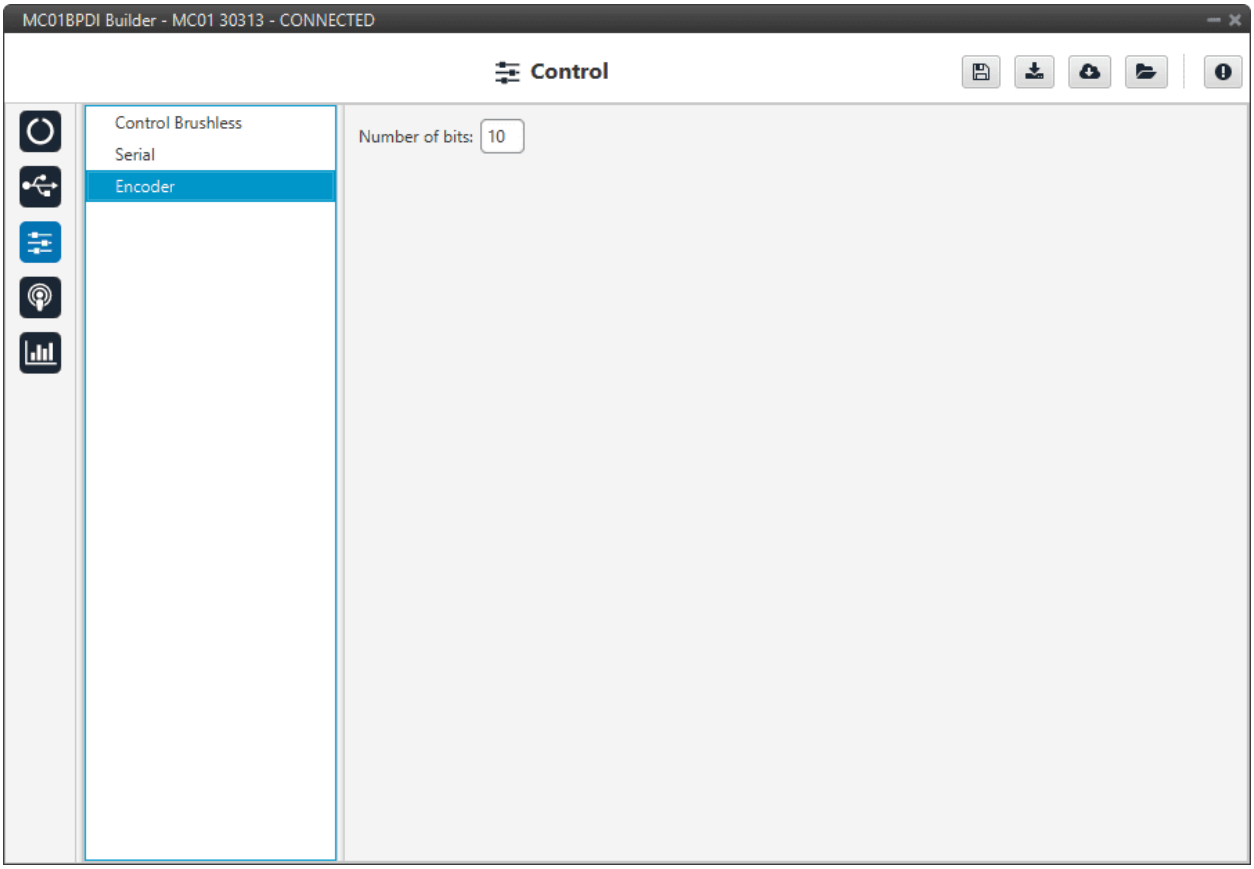


Fig. 16: **Encoder section**

2.4 Communications

This menu configures the SCIA port parameters in the same way as in the *Serial section* of the **Control** menu.

Note: Modifications carried out here will be applied in the parameters of the serial menu and vice versa.

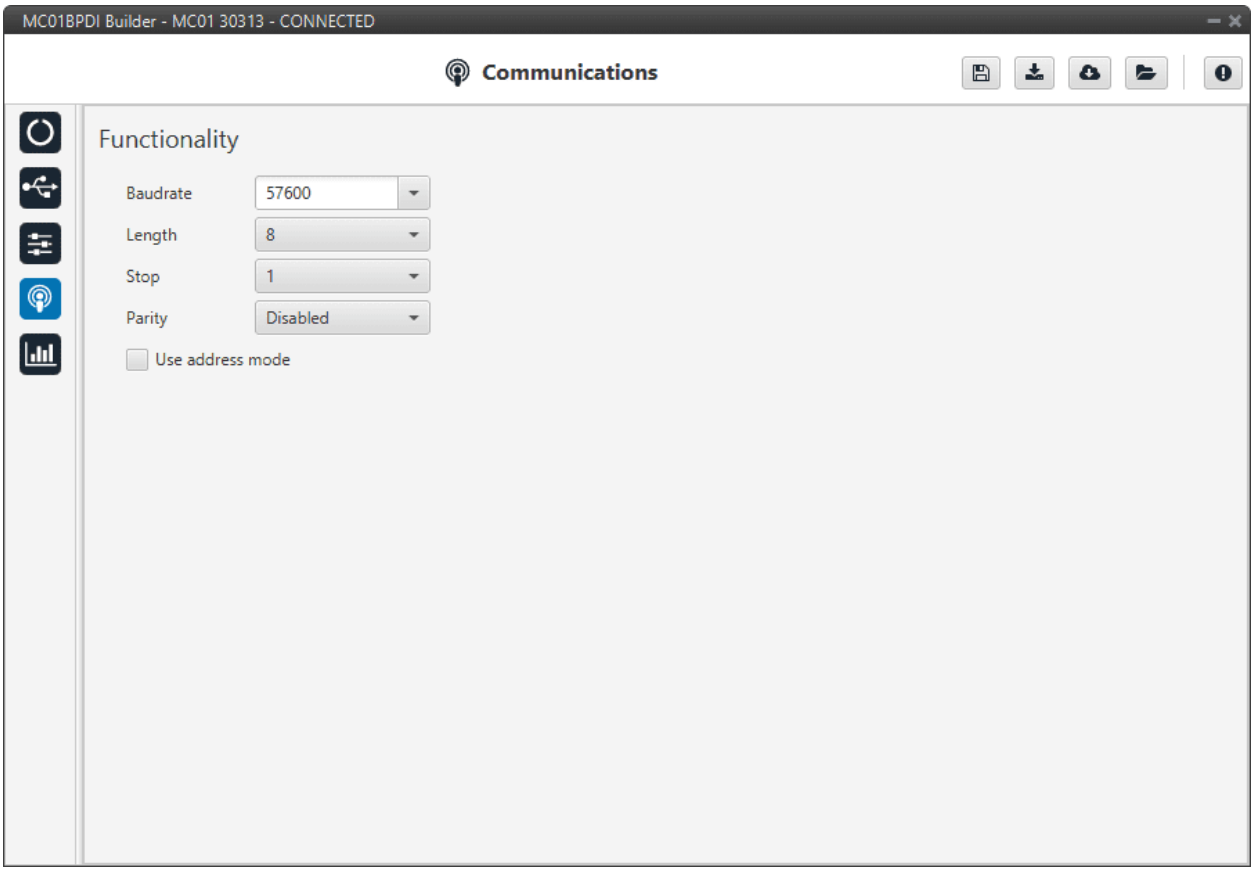


Fig. 17: **Communications menu**

2.5 Telemetry

In the Telemetry menu, the user chooses the variable or group of variables to be sent via the CAN bus. The following items can be configured:

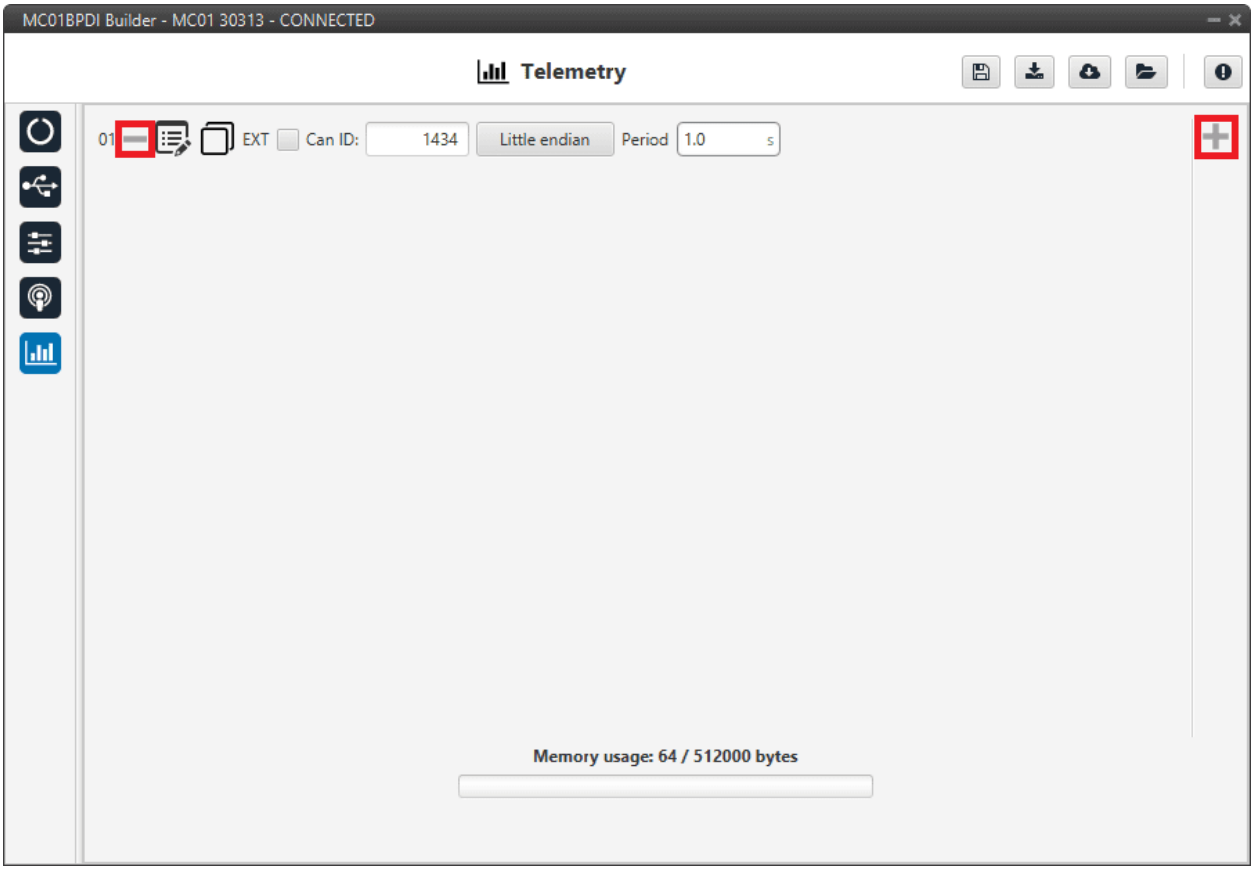


Fig. 18: **Telemetry menu**

By clicking on the + icon more messages can be added and by clicking on the − icon, messages can be deleted. Messages are configured clicking on the icon:

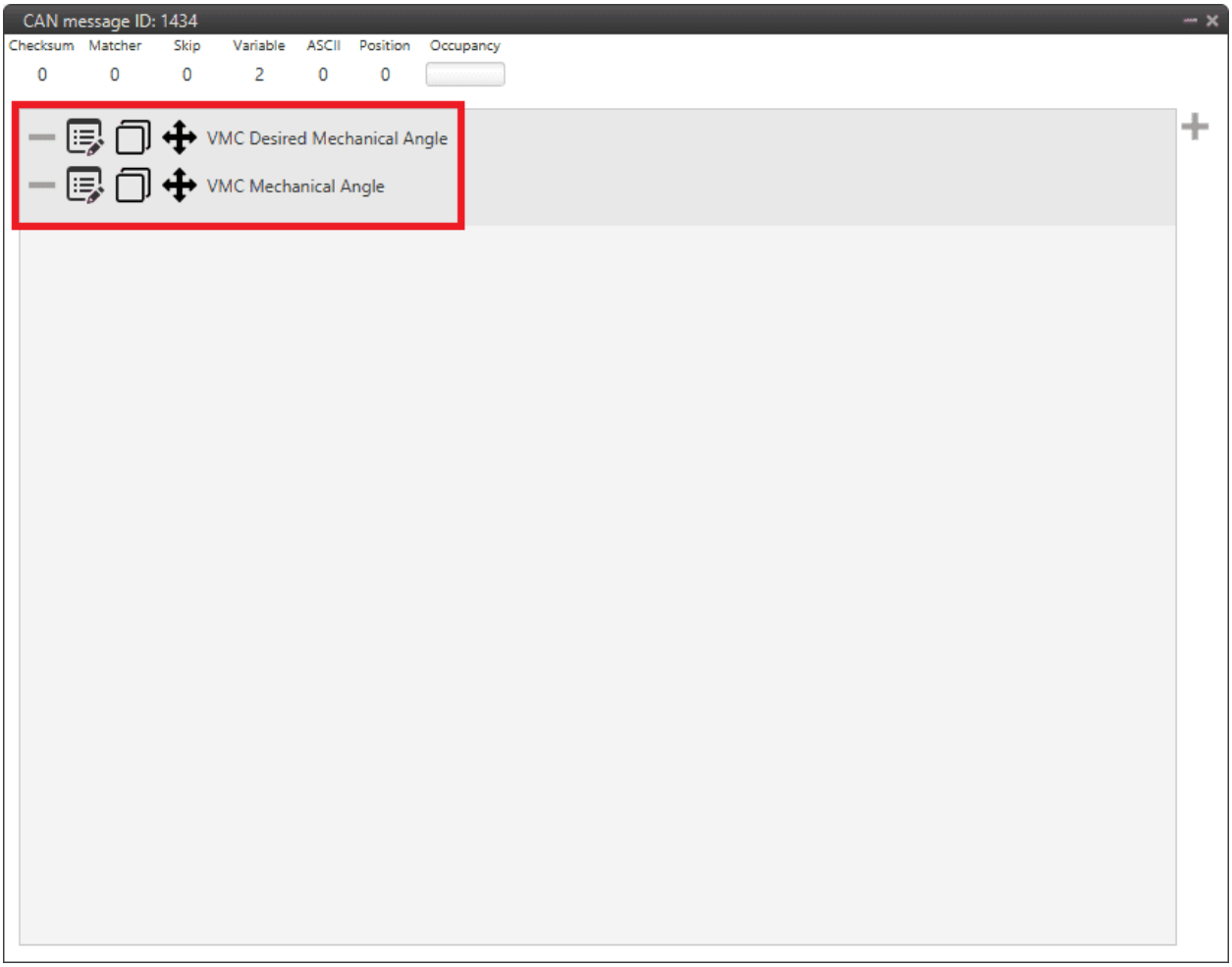


Fig. 19: **Telemetry menu - Custom messages example**

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