Veronte CAN Expander

Embention

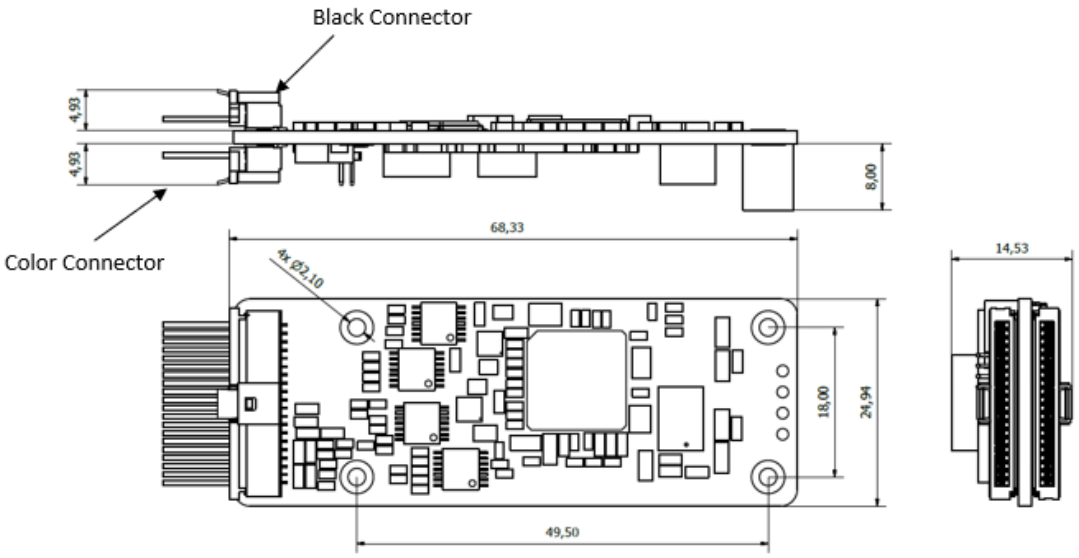
Dec 21, 2022

CONTENTS

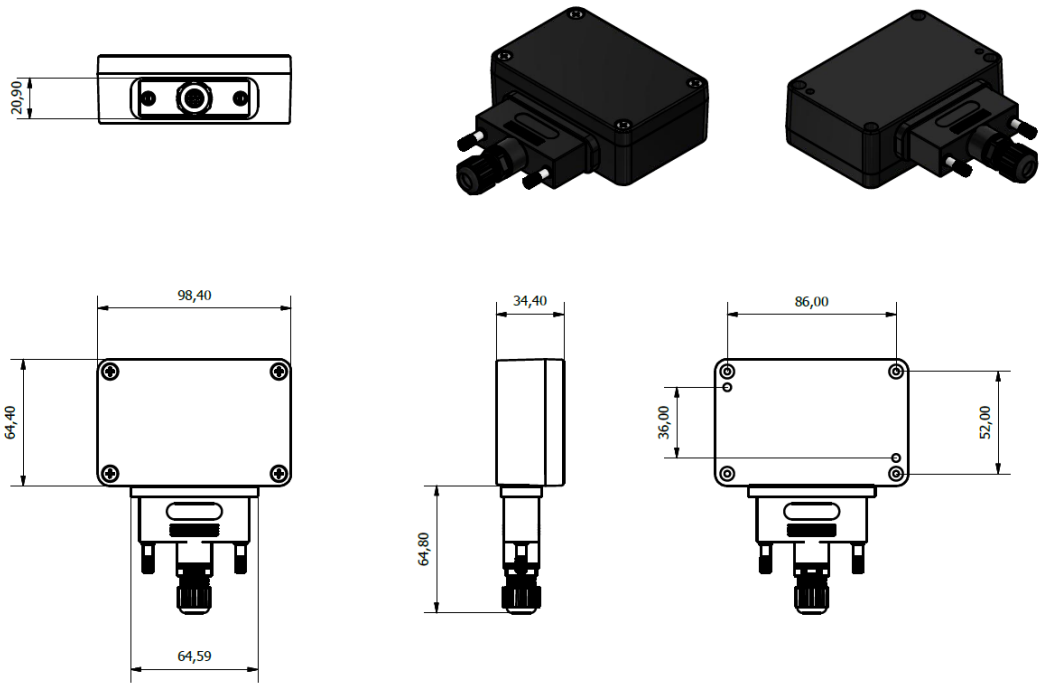
CHAPTER

ONE

ASSEMBLY

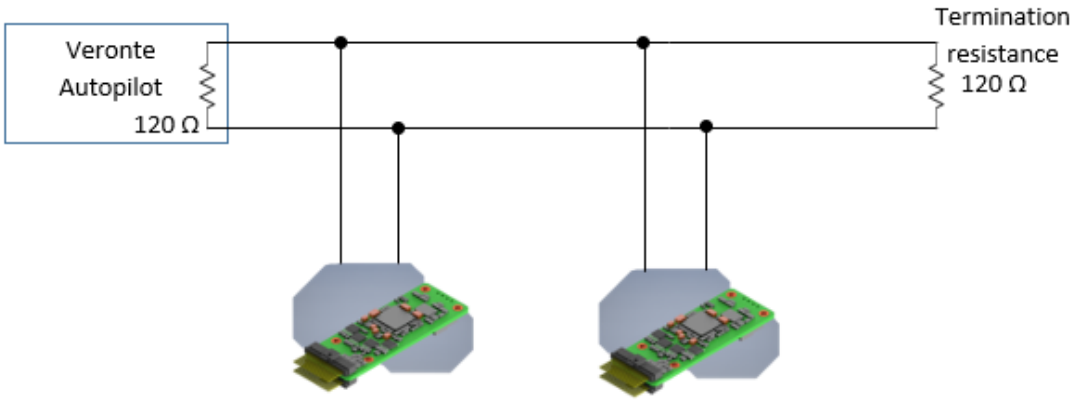


Veronte CEX dimensions



Veronte CEM dimensions

CAN Expander does not integrate a termination resistance in order to allow for multiple CAN Expander connected to the same line. Considering Veronte Autopilot includes one entrance resistance of 120 Ω, a second resistance needs to be placed at the end of the line (again 120 Ω).



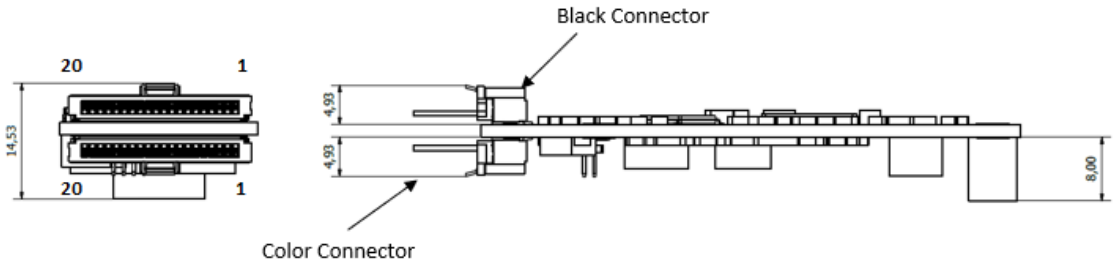
CAN assembly diagram example

CHAPTER

TWO

PINOUT

2.1 Veronte CEX



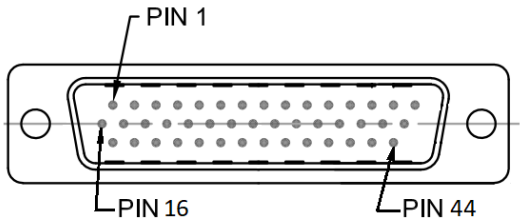
Veronte CEX pinout

Warning: Please note the colour code of the 2 connectors as wrong connections can result in permanent damage of CEX.

Connector A (Colored) - T1M-20-T-SH-L		
PIN Nº	I/O	Color
1	Power supply 1	Brown
2	Power supply 2	Red
3	GND	Orange
4	CAN A (H)	Yellow
5	CAN A (L)	Green
6	CAN B (L)	Blue
7	CAN B (H)	Purple
8	UART A (TX)	Gray
9	UART A (RX)	White
10	GND	Black
11	UART B (TX)	Brown
12	UART B (RX)	Red
13	GND	Orange
14	I2C SCL	Yellow
15	I2C SDA	Green
16	GND	Blue
17	3.3V (0.1A max)	Purple
18	GND	Gray
19	5V (0.1A max)	White
20	GND	Black

Connector B (Black) - T1M-20-T-SH-L	
PIN Nº	I/O
1	PWM 1
2	PWM 2
3	PWM 3
4	PWM 4
5	PWM 5
6	PWM 6
7	PWM 7
8	PWM 8
9	ECAP 1
10	ECAP 2
11	ECAP 3
12	ECAP 4
13	ANALOG 1 (3.3V)
14	ANALOG 2 (3.3V)
15	ANALOG 3 (5V)
16	ANALOG 4 (5V)
17	ANALOG 5 (12V)
18	ANALOG 6 (12V)
19	ANALOG 7 (36V)
20	ANALOG 8 (36V)

2.2 Veronte CEM



Veronte CEM - 780-M44-103L001 connector

PIN Nº	I/O	PIN Nº	I/O
1	PWM 1	23	GND
2	PWM 2	24	Power supply 1
3	PWM 3	25	Power supply 2
4	PWM 4	26	GND
5	PWM 5	27	CAN A (H)
6	PWM 6	28	CAN A (L)
7	PWM 7	29	CAN B (L)
8	PWM 8	30	CAN B (H)
9	GND	31	232_TX
10	ECAP 1	32	232_RX
11	ECAP 2	33	GND
12	ECAP 3	34	RX+
13	ECAP 4	35	RX-
14	GND	36	TX-
15	ANALOG 1 (3.3V)	37	TX+
16	ANALOG 2 (3.3V)	38	GND
17	ANALOG 3 (5V)	39	I2C SCL
18	ANALOG 4 (5V)	40	I2C SDA
19	ANALOG 5 (12V)	41	3.3V (0.1A max)
20	ANALOG 6 (12V)	42	GND
21	ANALOG 7 (36V)	43	5V (0.1A max)
22	ANALOG 8 (36V)	44	GND

CHAPTER

THREE

I/O SPECIFICATIONS

- All inputs are ESD protected
- Double, redundant, power supply inputs.
- Input voltage/current
  - Power input: 6V to 60V (DC)
  - Power consumption: 3W
- Vmax = 60V
  - IMax: 1.5A, Inom: 0.3A
- CAN
  - Complies with CAN Bus 2.0A and 2.0B Standards
  - Opto-Isolated (4kV)
  - Speed up to 1Mbps
- UART
  - TTL 3.3V signals up to 115200 baud
- I2C
  - 3.3V Signals up to 400KHz
- 3.3V Output
  - 100 mA fuse protected
- 5V Output
  - 100 mA fuse protected
- PWM Output
  - Voltage is 5V
  - Current I (oh) = 16mA and I (ol) = -16mA
  - Micro Edge Positioning (MEP) step size = 150ps
- Digital Inputs (ECAP)
  - Maximum voltage = 5V
  - Maximum input current = 2.5mA
  - Sampling rate: up to 1us

- Analog signals
  - Input impedance: 10GOhm
  - Resolution:
    - 0-3.3V pins: 0.00080V
    - 0-5V pins: 0.0012V
    - 0-12V pins: 0.0029V
    - 0-36V pins: 0.0087V

CHAPTER

FOUR

SOFTWARE

4.1 CAN Bus

Veronte CEX and CEM Communication Protocol over CAN Bus is defined as follow (v6.4.39 or higher):

- **cmd (8 bits):** Message Type
- **data (up to 56 bits):** Message Data

First 8 bits refers to the **Message Type** and are defined as follows:

Message Type	Value	Description
t_arbitration	0	Arbitration message
t_version	1	Version request / response
t_pwm_0_3_set	2	PWMs 0 to 3
t_pwm_4_7_set	3	PWMs 4 to 7
	4	Reserved
t_esc_tm	5	Scorpion Tribunus ESC telemetry data
t_esc_tm2	6	Jeti ESC telemetry data
t_bec_tm1	7	Jeti BEC telemetry data
t_bec_tm2	8	Jeti BEC telemetry data 2
t_temp_tm	9	Jeti Temperature sensor telemetry data
t_mcu_cmd	10	MCU command
t_pwm_8_11_set	11	PWMs 8 to 11
t_pwm_12_15_set	12	PWMs 12 to 15
t_pwm_16_19_set	13	PWMs 16 to 19
	14	Reserved
	15	Reserved
t_cmd_maint	16	Command to go to Maintenance Mode
t_stick_sel	17	Command for Stick selection
t_mcu_tm1	18	MCU telemetry data 1
t_mcu_tm2	19	MCU telemetry data 2

The following bytes refer to the **Message Data**.

Next sections decribe each of the possible messages.

4.1.1 Arbitration

CAN Expander Arbitration Status message is composed as follow:

- Message 1

Message Type	Bytes	Description
t_arbitration	1	Message type (0)
Flag	1	Status Flag [0xFF]
CAP	7 bits	Active Autopilot (Current)
Flag	1 bit	Arbitrating
Flag	1 bit	AP0 Alive
Flag	1 bit	AP1 Alive
Flag	1 bit	AP2 Alive
Flag	1 bit	AP3 Alive (External)
Flag	1 bit	AP0 Ready
Flag	1 bit	AP1 Ready
Flag	1 bit	AP2 Ready
Flag	1 bit	AP3 Ready (External)

- Message 2 (One for each Autopilot)

Message Type	Bytes	Description
t_arbitration	1	Message type (0)
AP ID	1	Autopilot [0, 3]
Score	4	Autopilot score as Float (32 bits)

4.1.2 Request Version

The command needed to ask the CEX version is the following:

- **cmd (8 bits):** t_version

And CEX will answer with:

- **cmd (8 bits):** t_version (1)
- **data0 (8 bits):** App -> 9 CEXv2
- **data1 (8 bits):** Version
- **data2 (8 bits):** Major
- **data3 (8 bits):** Minor
- **data4 (8 bits):**
  - **bits 7-3:** 0
  - **bit 1:** arbitration_enabled
  - **bit 0:** arbitrating

4.1.3 Command PWMs

Each PWM in Veronte CAN Expander has to be associated to a Sub Id that indicates which of the CAN Bus message's PWM is listening to.

That allows to control up to four PWMs using the same message if that is desired. Each message is composed by 4 PWMs maximum.

PWMs from 0 to 3 are sent in a message that includes 4 PWMs coded as 12-bit integers:

- **cmd (8 bits):** t_pwm_0_3_set (2)
- pwm0 (12 bits) PWM value for sub-id 0
- pwm1 (12 bits) PWM value for sub-id 1
- pwm2 (12 bits) PWM value for sub-id 2
- pwm3 (12 bits) PWM value for sub-id 3

PWMs from 4 to 7 are sent in a message that includes 4 PWMs coded as 12-bit integers:

- **cmd (8 bits):** t_pwm_4_7_set (3)
- pwm0 (12 bits) PWM value for sub-id 4
- pwm1 (12 bits) PWM value for sub-id 5
- pwm2 (12 bits) PWM value for sub-id 6
- pwm3 (12 bits) PWM value for sub-id 7

PWMs from 4 to 7 are sent in a message that includes 4 PWMs coded as 12-bit integers:

- **cmd (8 bits):** t_pwm_8_11_set (11)
- pwm0 (12 bits) PWM value for sub-id 8
- pwm1 (12 bits) PWM value for sub-id 9
- pwm2 (12 bits) PWM value for sub-id 10
- pwm3 (12 bits) PWM value for sub-id 11

PWMs from 4 to 7 are sent in a message that includes 4 PWMs coded as 12-bit integers:

- **cmd (8 bits):** t_pwm_12_15_set (12)
- pwm0 (12 bits) PWM value for sub-id 12
- pwm1 (12 bits) PWM value for sub-id 13
- pwm2 (12 bits) PWM value for sub-id 14
- pwm3 (12 bits) PWM value for sub-id 15

PWMs from 16 to 19 are sent in a message that includes 4 PWMs coded as 12-bit integers:

- **cmd (8 bits):** t_pwm_16_19_set (13)
- pwm0 (12 bits) PWM value for sub-id 16
- pwm1 (12 bits) PWM value for sub-id 17
- pwm2 (12 bits) PWM value for sub-id 18
- pwm3 (12 bits) PWM value for sub-id 19

4.1.4 MCU Telemetry (CEX to Veronte)

The telemetry sent by CEX over CAN Bus is composed by:

- Message 1

Message Type	Bytes	Description
t_mcu_tm1	1	Message type (18)
Battery SN	1	Battery Serial Number [0]
Battery SN	1	Battery Serial Number [1]
Temperature	1	Battery Temperature (as received from MCU)
Voltage	1	Low Cell Voltage (as received from MCU)
	4 bits	Reserved (Zeros)
Status Bit	1	PWM receiving Ok
Status Bit	1	CAN PWM receiving Ok
Status Bit	1	CAN B receiving
Status Bit	1	CAN A receiving

- Message 2

Message Type	Bytes	Description
t_mcu_tm2	1	Message type (19)
Battery SN	1	Battery Serial Number [2]
Battery SN	1	Battery Serial Number [3]
Battery SN	1	Battery Serial Number [4]
Battery SN	1	Battery Serial Number [5]
Battery SN	1	Battery Serial Number [6]
Battery SN	1	Battery Serial Number [7]

4.1.5 MCU Telemetry (Veronte to CEX)

The telemetry sent from Veronte to CEX must be configured as follows:

Message Type	Bytes	Description
t_mcu_cmd	1	Message type (10)
SUB-id	1	SUB-id A
LED Value	1	Value A
SUB-id	1	SUB-id B
LED Value	1	Value B
SUB-id	1	SUB-id C
LED Value	1	Value C

Each Veronte CAN Expander will use the SUB-id of the PWM associated to the “Scorpion Tribunus”/PWM ID to identify the value to be used.

4.1.6 Scorpion Tribunus ESC Telemetry

The telemetry read from the Scorpion ESC is sent as:

Message Type	Bytes	Description
t_esc_tm	1	Message type (5)
Voltage	1	Input voltage in range [0, 85]
Temperature	1	Temperature in Celsius
Errors	1	Error Flags from the ESC
Current	1	Current in Amps [0, 255]
Consumption	1	Consumption in mAmps [0, 25500]
RPMs	1	RPMs [0, 25500]
Throttle	1	Throttle as percentage*2 [0, 200]

4.1.7 JetiTM ESC Telemetry

The telemetry read from Jeti-TM compatible ESCs is sent as:

Message Type	Bytes	Description
t_esc_tm2	1	Message type (6)
Throttle	1	Throttle value [0, 200]
RPMs	2	Current RPMs
Voltage	10 bits	Input voltage in the range [0, 70] Volts
Temperature	10 bits	Temperature in the range [0, 575] Kelvin
Current	12 bits	Current in the range [0, 400.0] Amps

4.1.8 Jeti BEC Telemetry

The telemetry read from a BEC will be sent in 2 different messages:

- Message 1:

Message Type	Bytes	Description
t_bec_tm1	1	Message type (7)
Device ID	2	
Voltage In	12 bits	Input voltage in the range [0, 70] Volts
Voltage Out	12 bits	Output voltage in the range [0, 70] Volts
Temperature	12 bits	Temperature in the range [0, 575] Kelvin

- Message 2:

Message Type	Bytes	Description
t_bec_tm2	1	Message type (8)
Device ID	2	
Current	12 bits	Current in the range [0, 100.0] Amps

4.1.9 Jeti Temperature Sensor Telemetry

The telemetry read from a Temperature sensor will be send as:

Message Type	Bytes	Description
t_temp_tm	1	Message type (9)
Device ID	2	
Temperature 1	12 bits	Measured temperature 1 in the range [0, 750] Kelvin
Temperature 2	12 bits	Measured temperature 2 in the range [0, 750] Kelvin

---

4.1.10 Set Maintenance Mode Command

This command will configure the CEX in maintence mode, setting its configuration in a way that Communications can work over SCI-A, SCI-B or Serial-Over-CAN configured as:

- SCI-A and SCI-B: 115200 bauds, 8 data bits, 1 stop, no parity
- Serial over CAN:
  - TX Id: 1301
  - RX Id: 1301

The format of the command is:

- **cmd (8 bits):** t_cmd_maint (16)

4.1.11 Stick Selection Command

This command is used to enable or disable the CEX PPM reader. If the **address** received matches the CEX's one, CEX PPM reader will be enabled, otherwise it will be disabled.

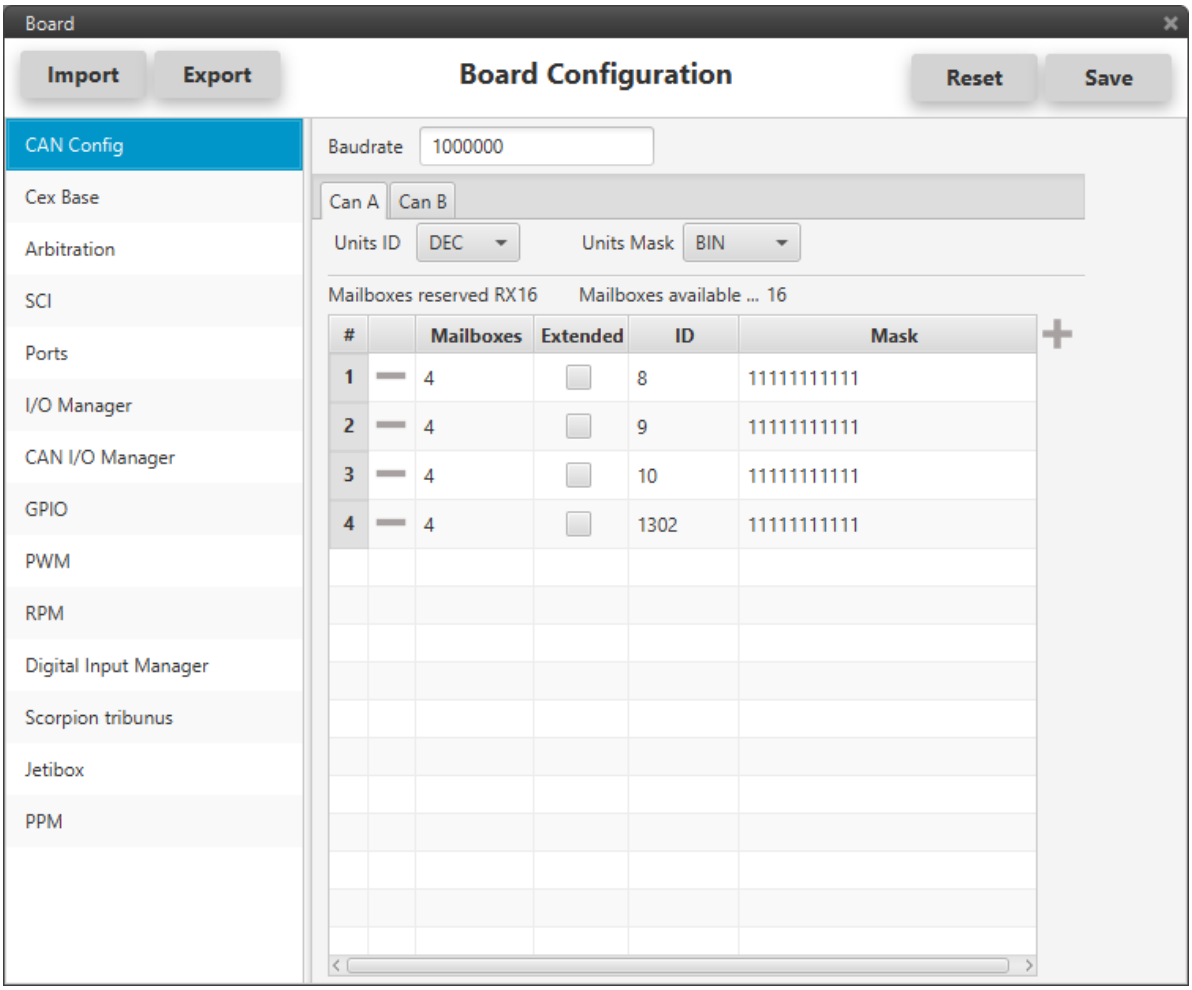
The format of the command is:

- **cmd (8 bits):** t_stick_sel (17)
- address (16 bits)

4.2 CEX CAN Setup

4.2.1 CAN Basic Setup

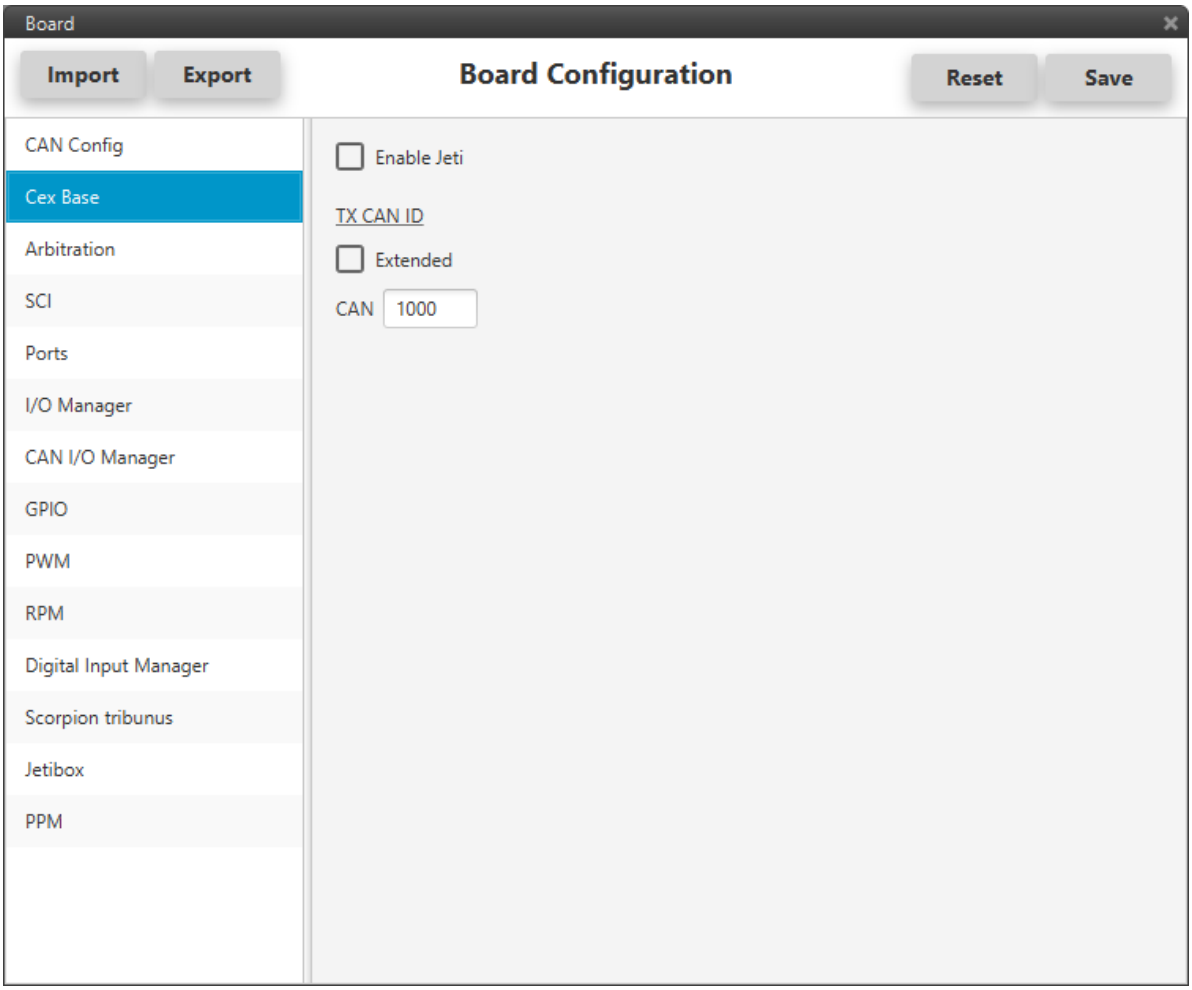
Main screen to configure bus speed and the reception mailboxes of each CAN bus:



CAN BUS configuration

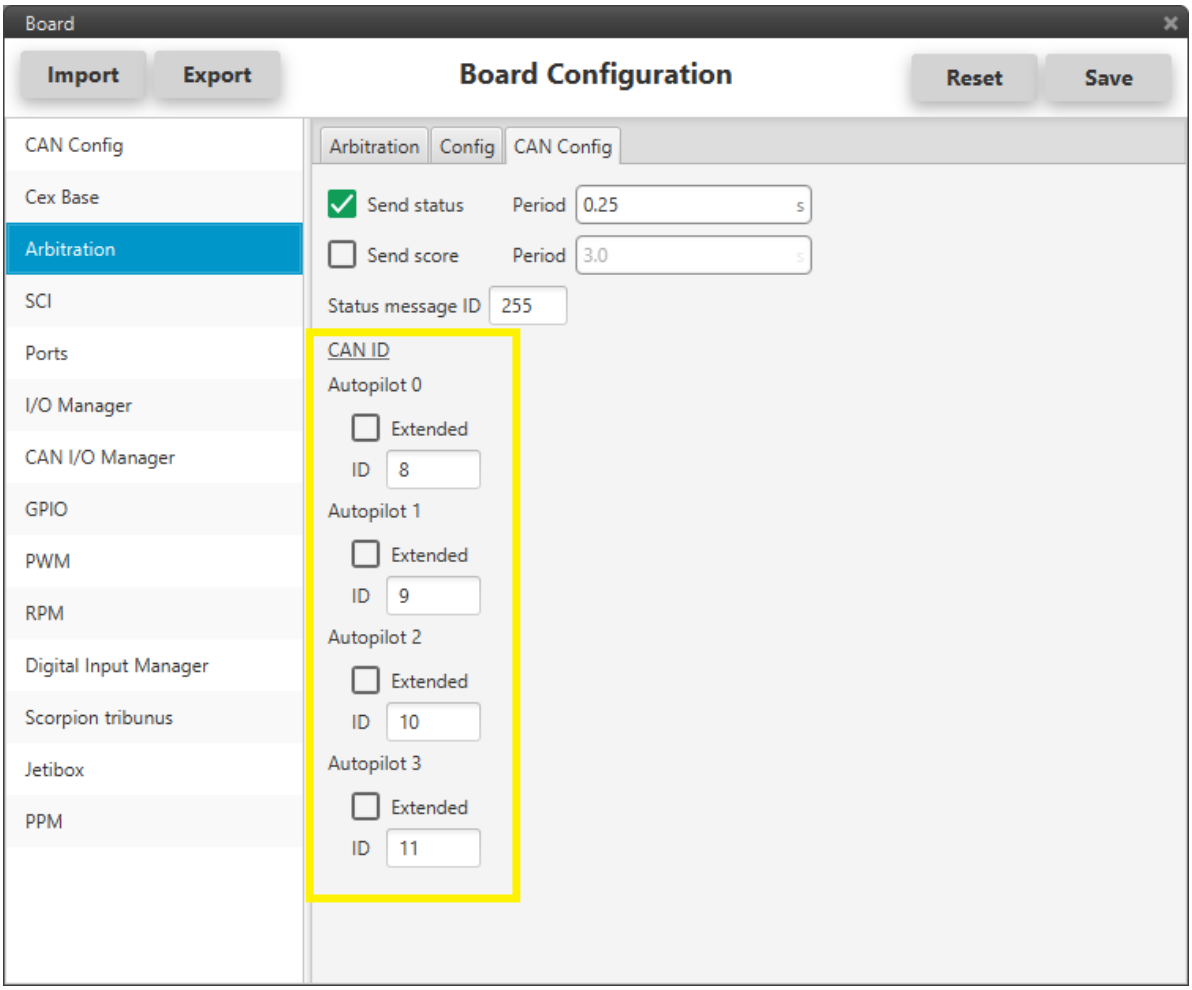
4.2.2 CEX CAN Transmission ID

CEX is able to send information about its version, Arbitration status (check the Arbiter manual for a description of these messages) and Jeti telemetry from devices connected to CEX. We can define the CAN Id used to those messages:



4.2.3 CAN Reception IDs

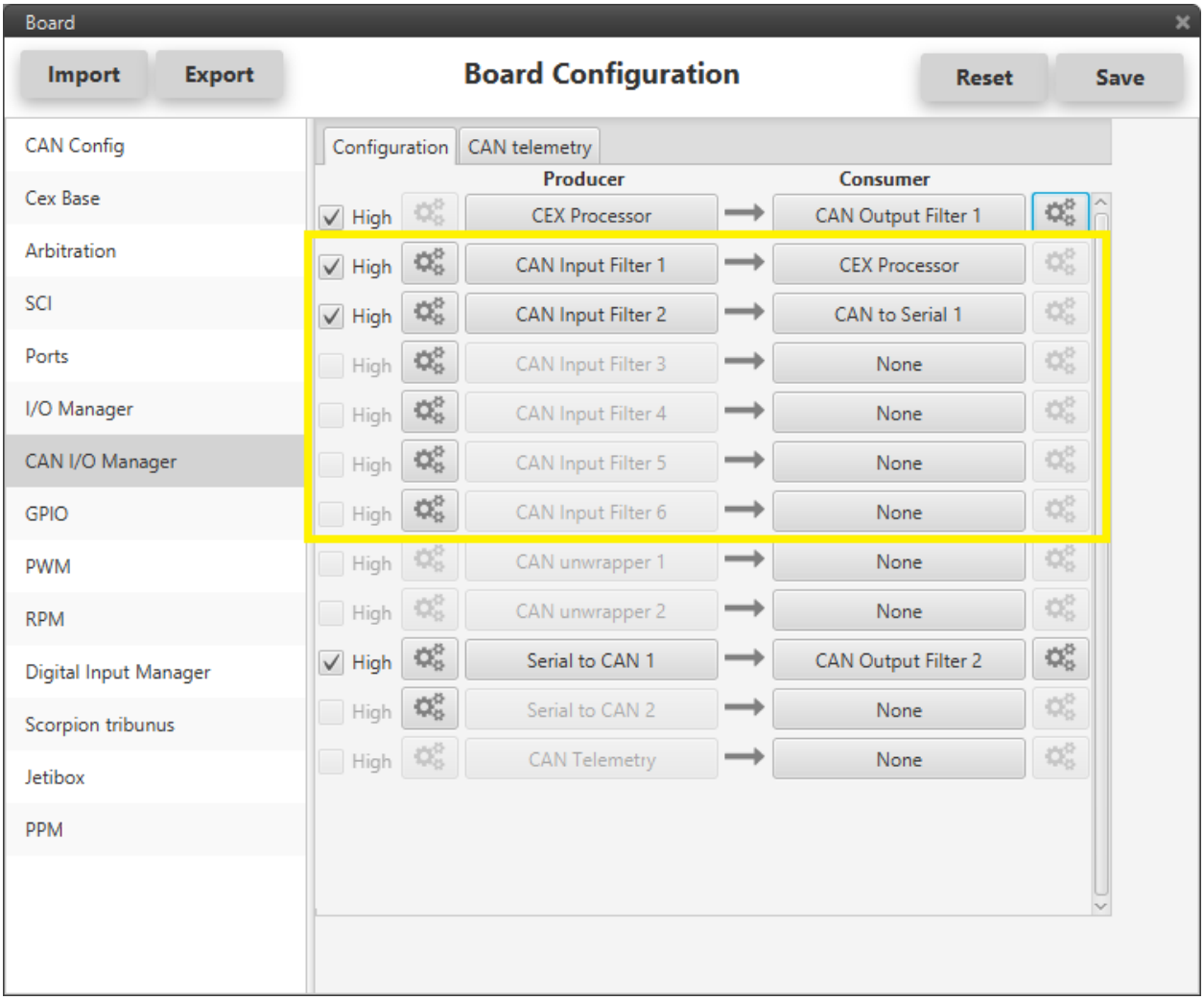
We CAN setup the recepcion CAN Ids for each of the 4 possible Veronte Autopilots sending data to CEX. **If arbitration is not enabled, only the configuration of the Autopilot 0 will be used**.



4.2.4 CAN I/O Interconnections

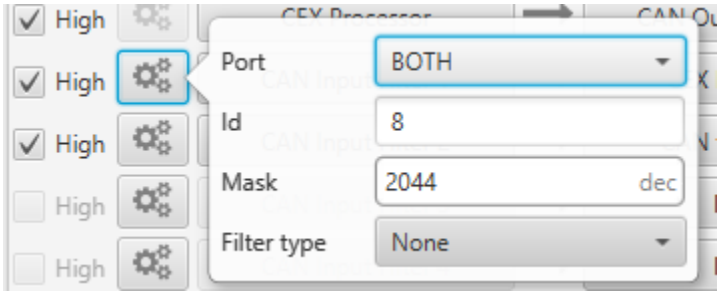
Once the CAN IDs are set, we shall configure:

- The Input Filters going to be used.
- The connection between input filters and data Consumers.

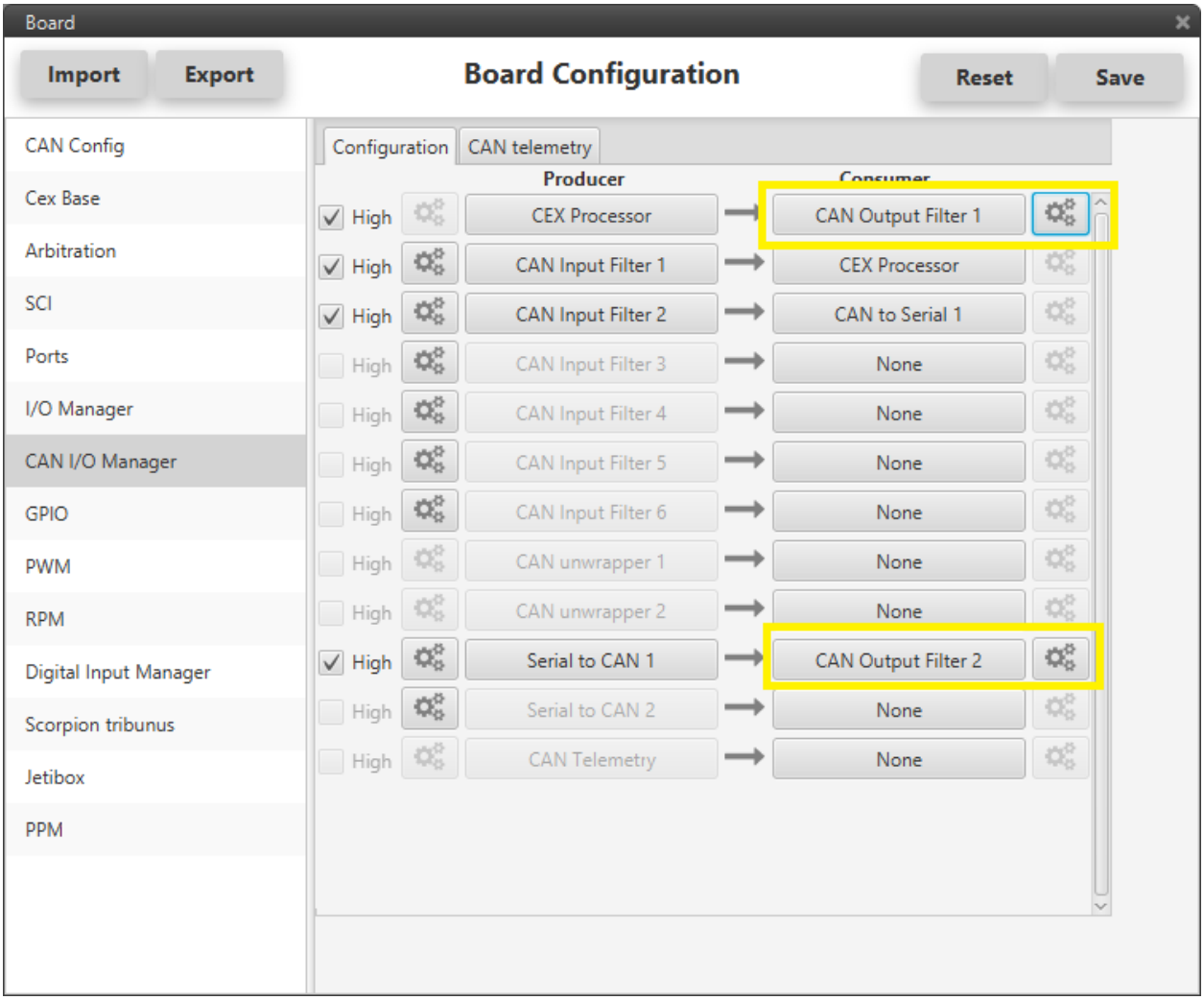


For each CAN Input Filter, it is requried to configure the can bus from which it listens (CAN-A, CAN-B or Both), the CAN id, the CAN Id mask and the type of frame (Standard, Extended or Both).

The Mask defines the bits that sould match. For example if we want to admit statndard Ids (11 bits) from 8 to 11 (100 to 111 in binary) we should set the mask to binary 11111111100, that is 2044 in decimal.



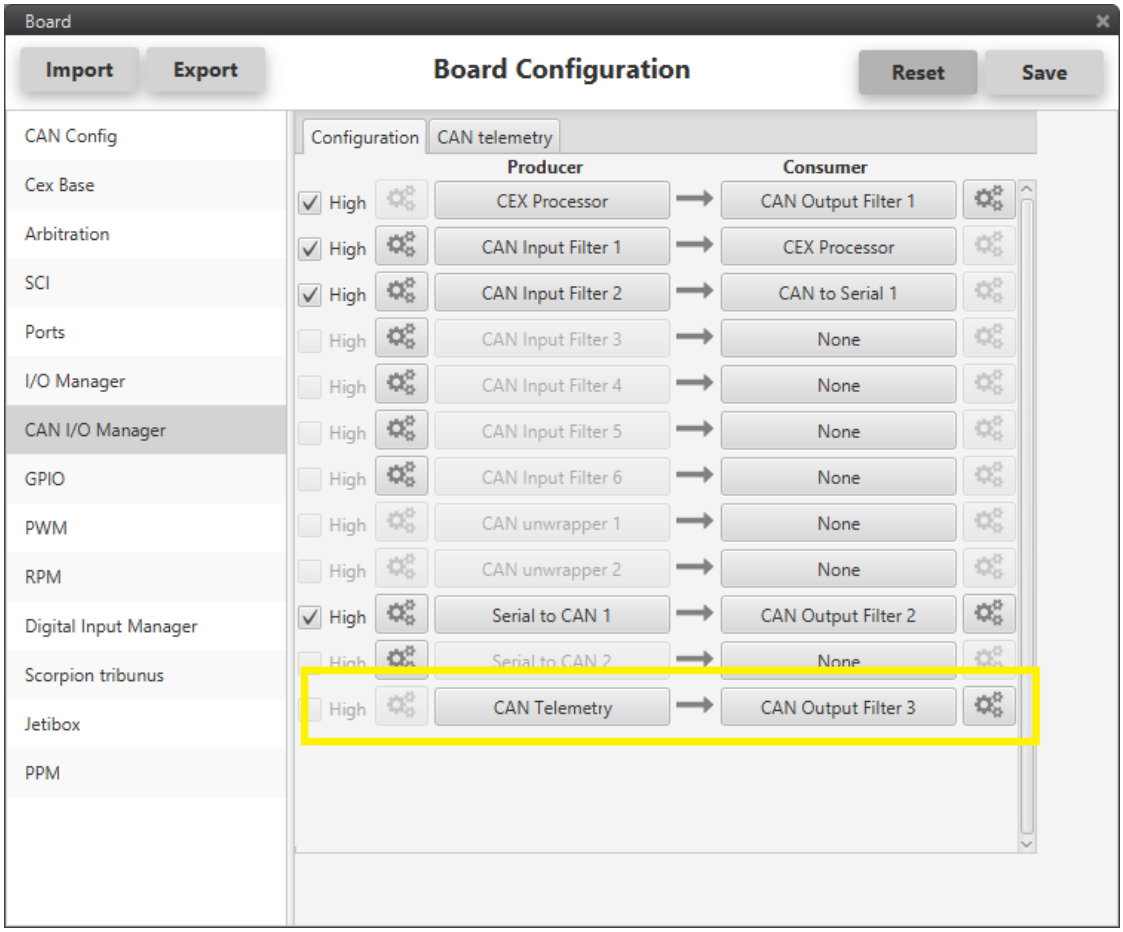
Next step is to connect each of the desired data Producers to an Output Filter, and configure both the Producer and the Output Filter:



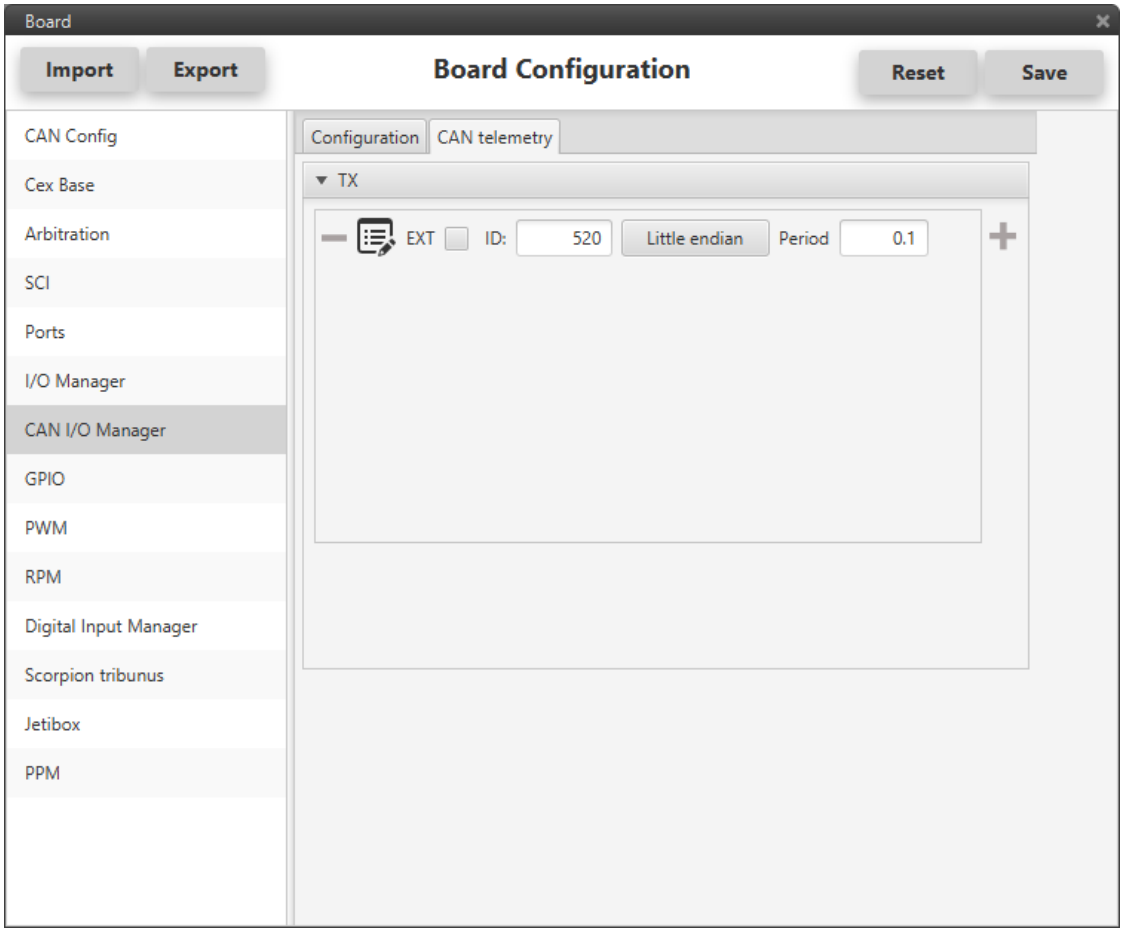
4.2.5 CAN Telemetry

CEX is able to send telemetry via CAN. To enable this feature CAN Telemetry producer shall:

1. Connect it to an Ouptut Filter as follows:



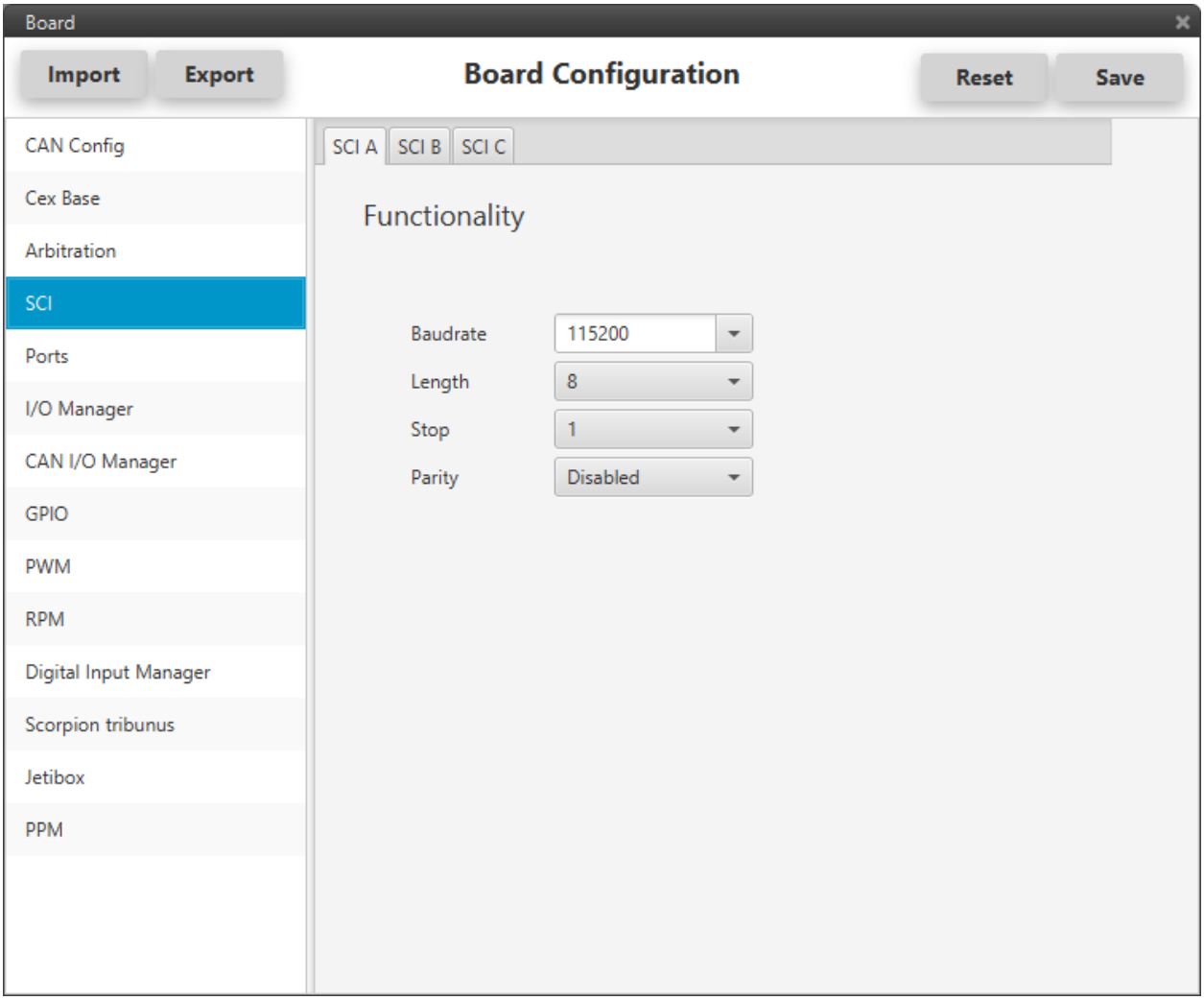
2. Select the fields to send in the Telemetry tab. **Note that each message is limited to 64 bits and larger messages will be cut to that size**.



4.3 Serial Setup

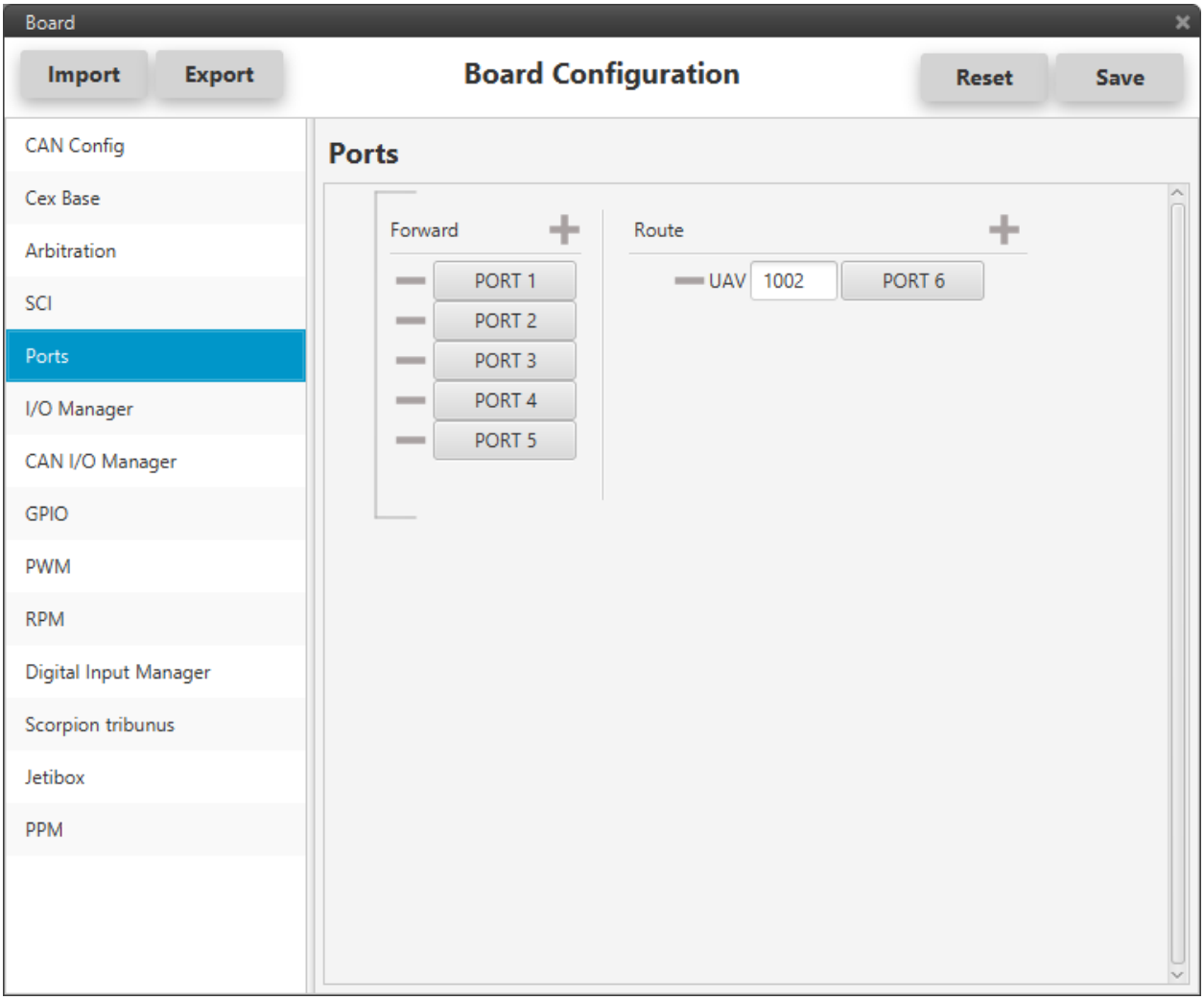
CEX can use up to three serial peripherals, which can be individually configured with given:

- Baudrate
- Data length: 4 to 8 bits
- Number of Stop bits: 1, 1.5, 2
- Parity: disabled, odd, even
- 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.



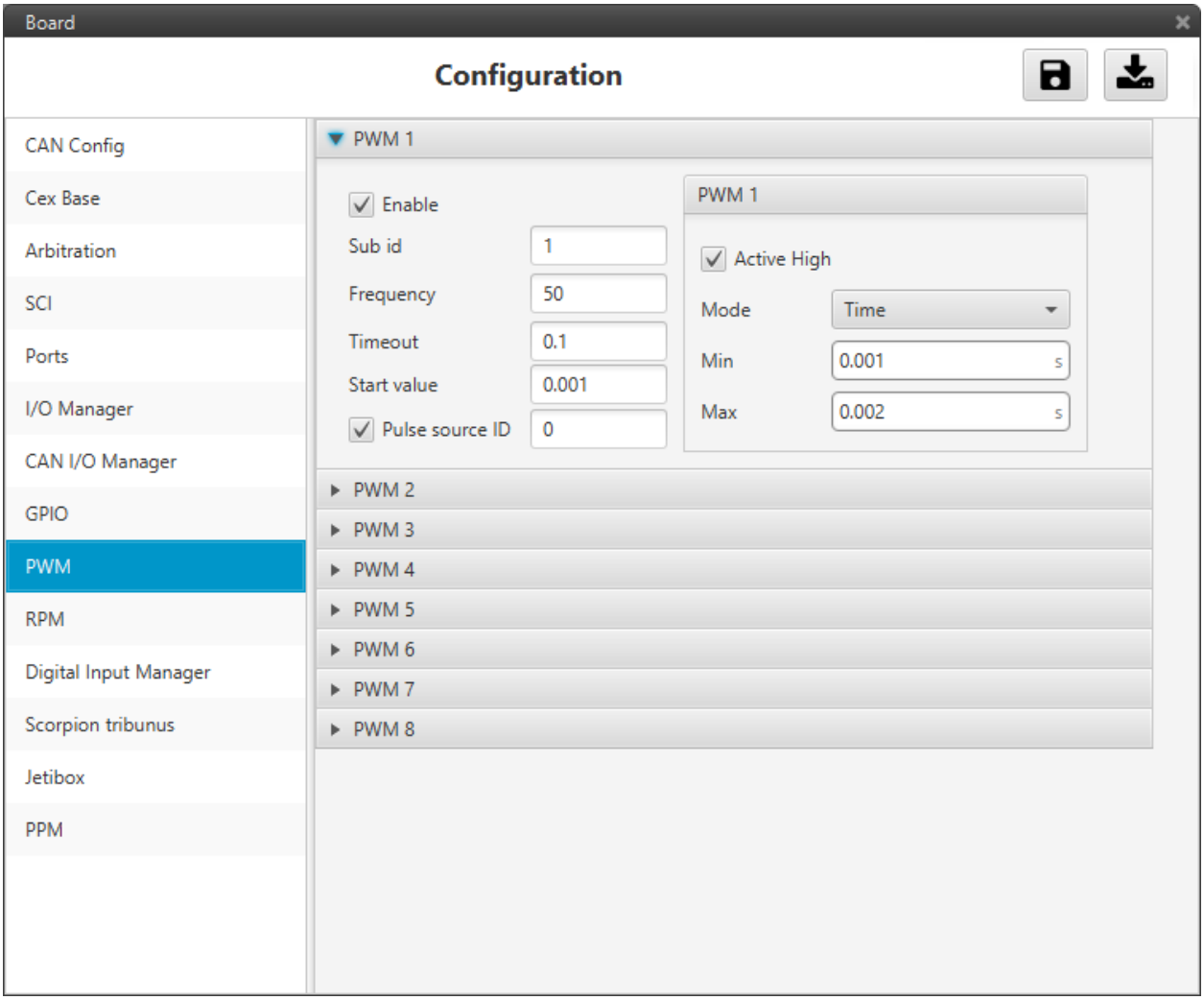
4.4 Ports

CEX can be configured to route its communications to any specific port given a destination address:



4.5 PWMs

In this tab we can configure each PWM:



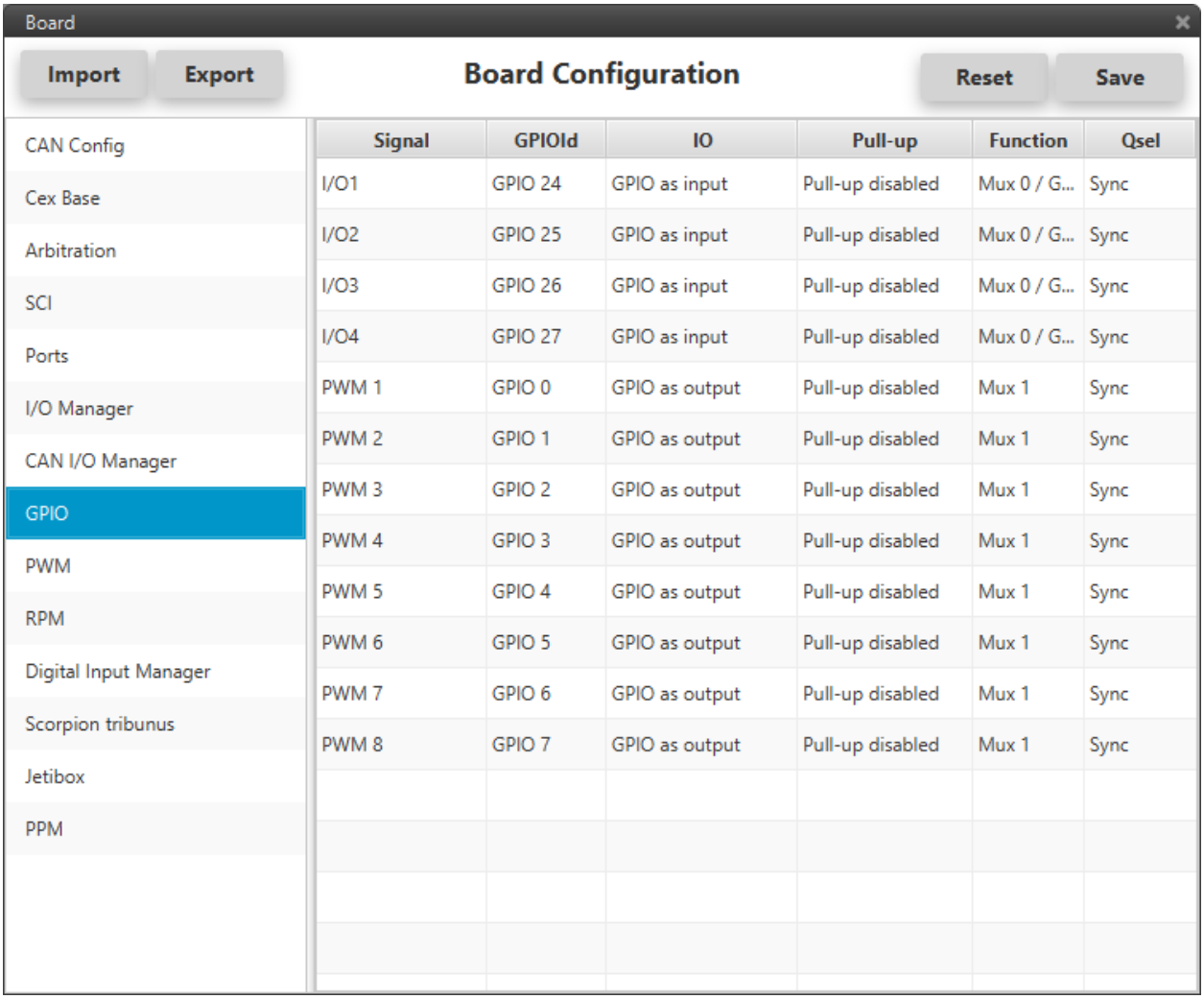
Note that the PWMs in CEX works in normalized mode, so when the input value is 0 the output value will be the minimum configured, and when the input value is 4095 (12 bits all with ones), the output will be the maximum configured. This approach allows usage of the maximum resolution for the commanded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

The configuration items are:

- **Enable:** Define if the PWM is enabled or not.
- **Sub id:** Define from which CAN PWM message element reads its value [0, 19].
- **Frequency:** PWM output frequency.
- **Timeout:** If a PWM message is not received in less than this time, the PWM will 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 Digital Input Manager tab.
- PWM specifics
  - **Active High:** Polarity high or low.
  - **Mode:** Time os Duty cicle
  - **Min:** Minimum value. That will output when the PWM message specifies 0
  - **Max:** Minimum value. That will output when the PWM message specifies 4095

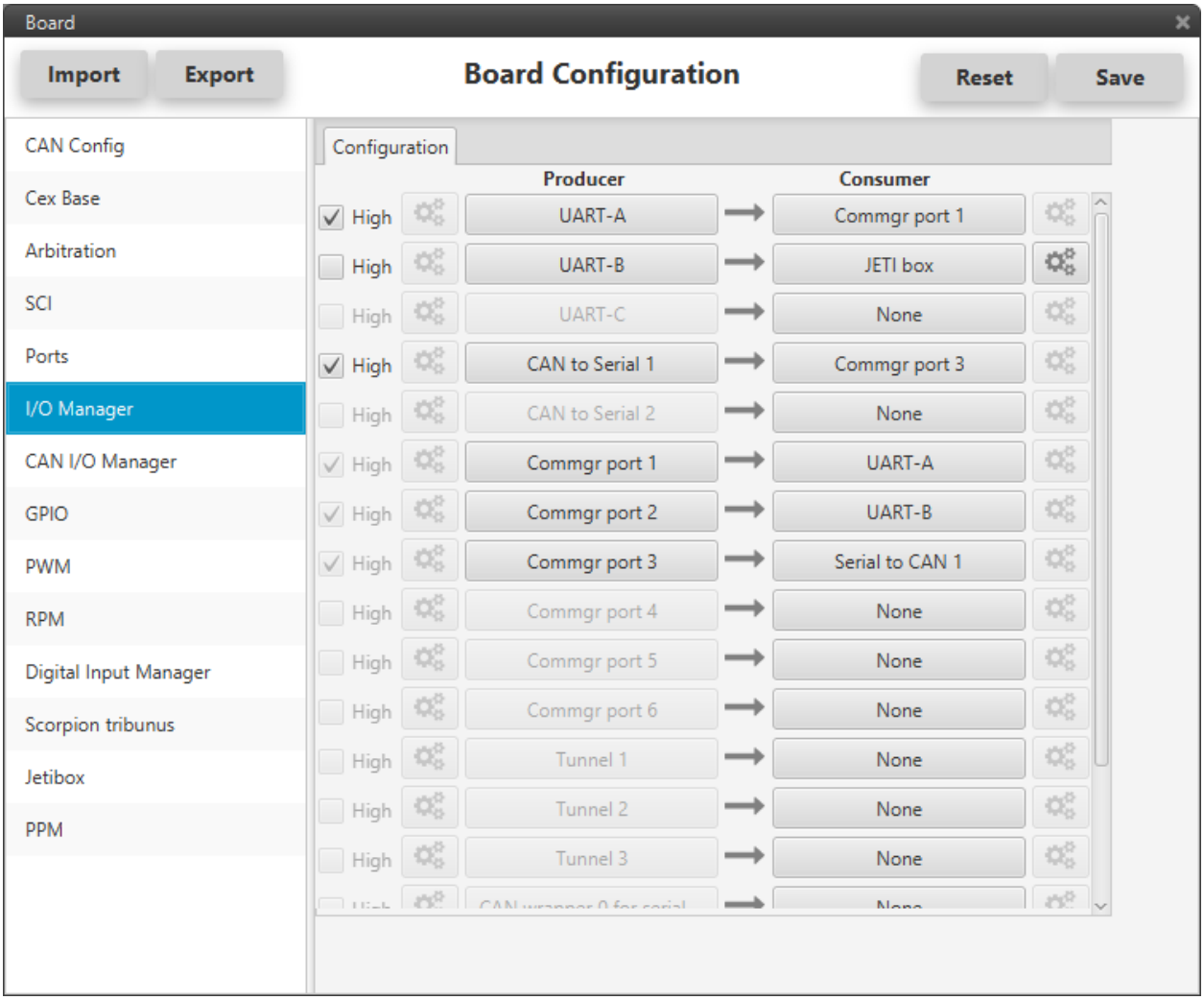
4.6 GPIO Manager

In this tab we can configure each individual GPIO behavior:



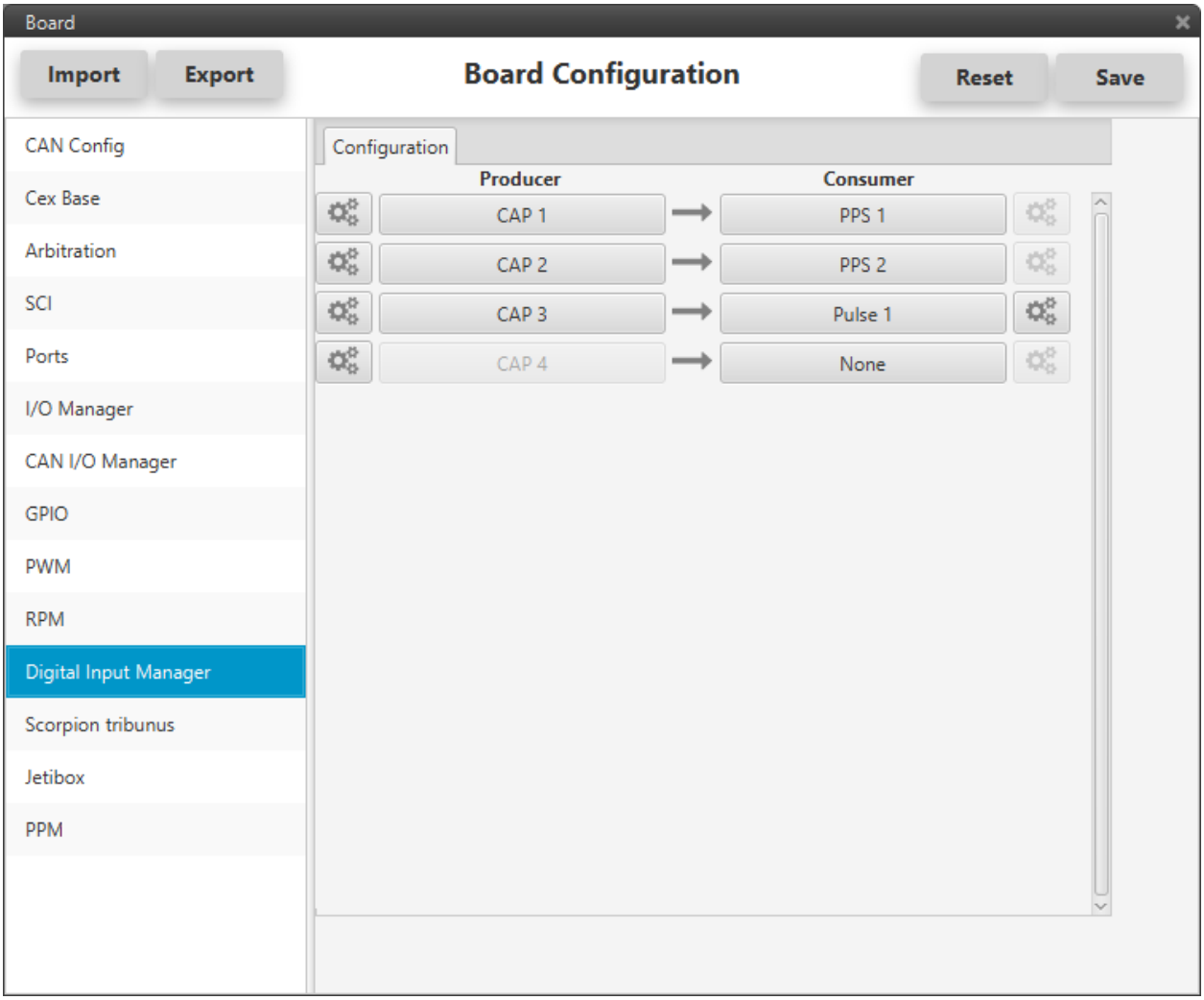
4.7 IO Manager

In this tab we can configure how serial-based devices and ports are connected:



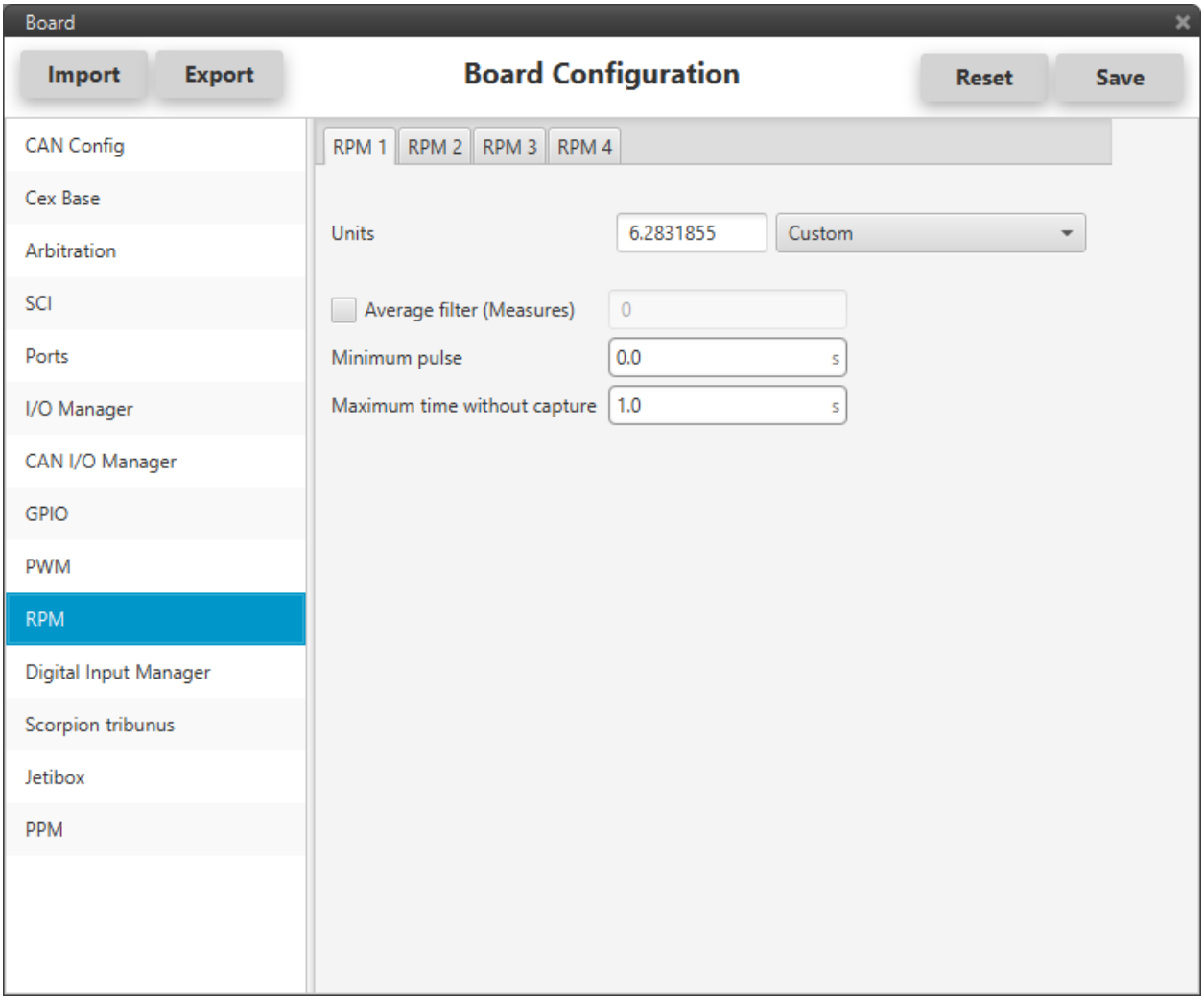
4.8 Digital IO

CEX digital inputs can be used to measure pulse count, pulse widths and PPM signals from a RC radio. We shall connect each source to the desired consumer to allow measuremets.



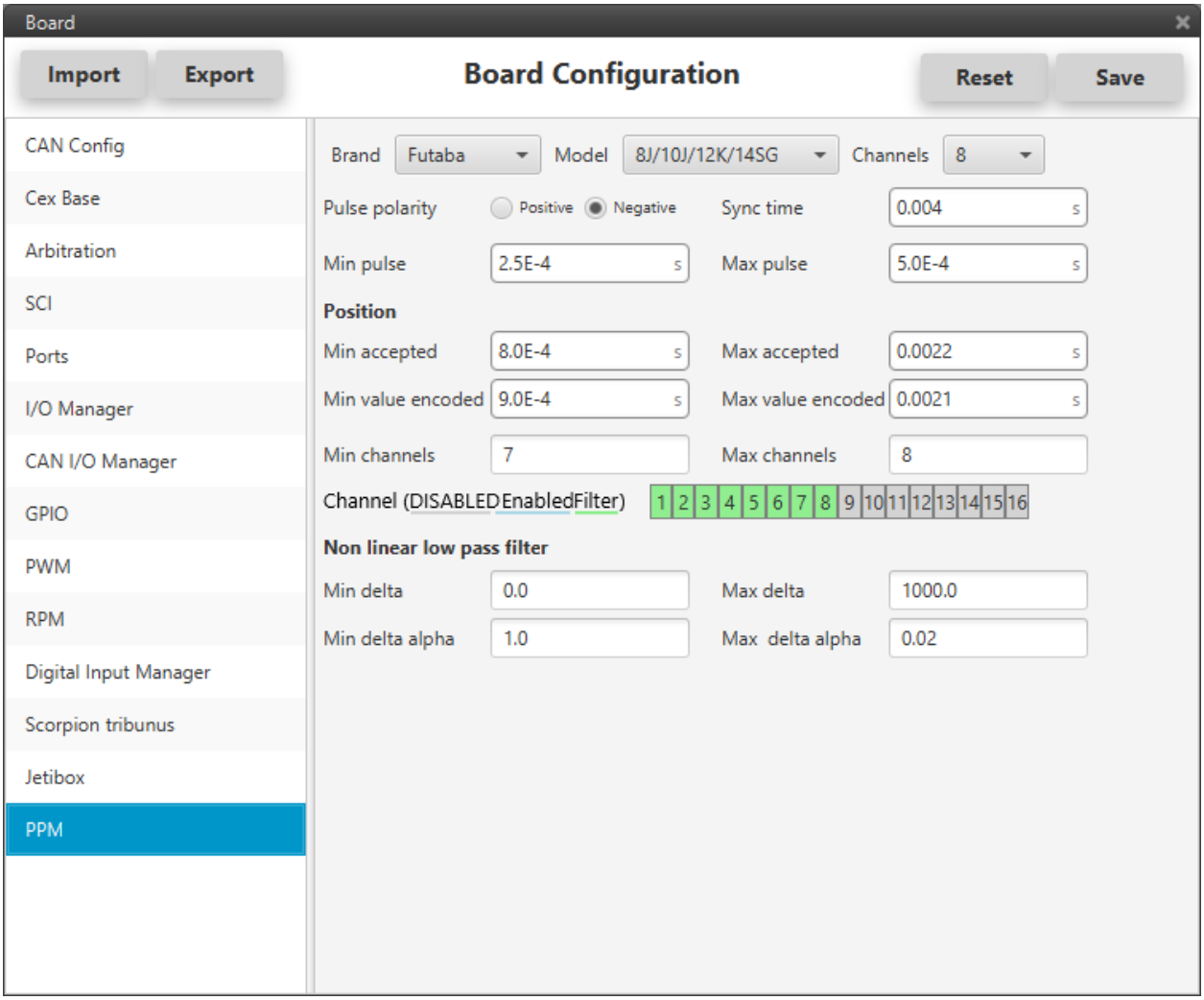
4.9 RPMs

CEX can measure RPMs by measuring from up to four inputs sources:



4.10 PPM

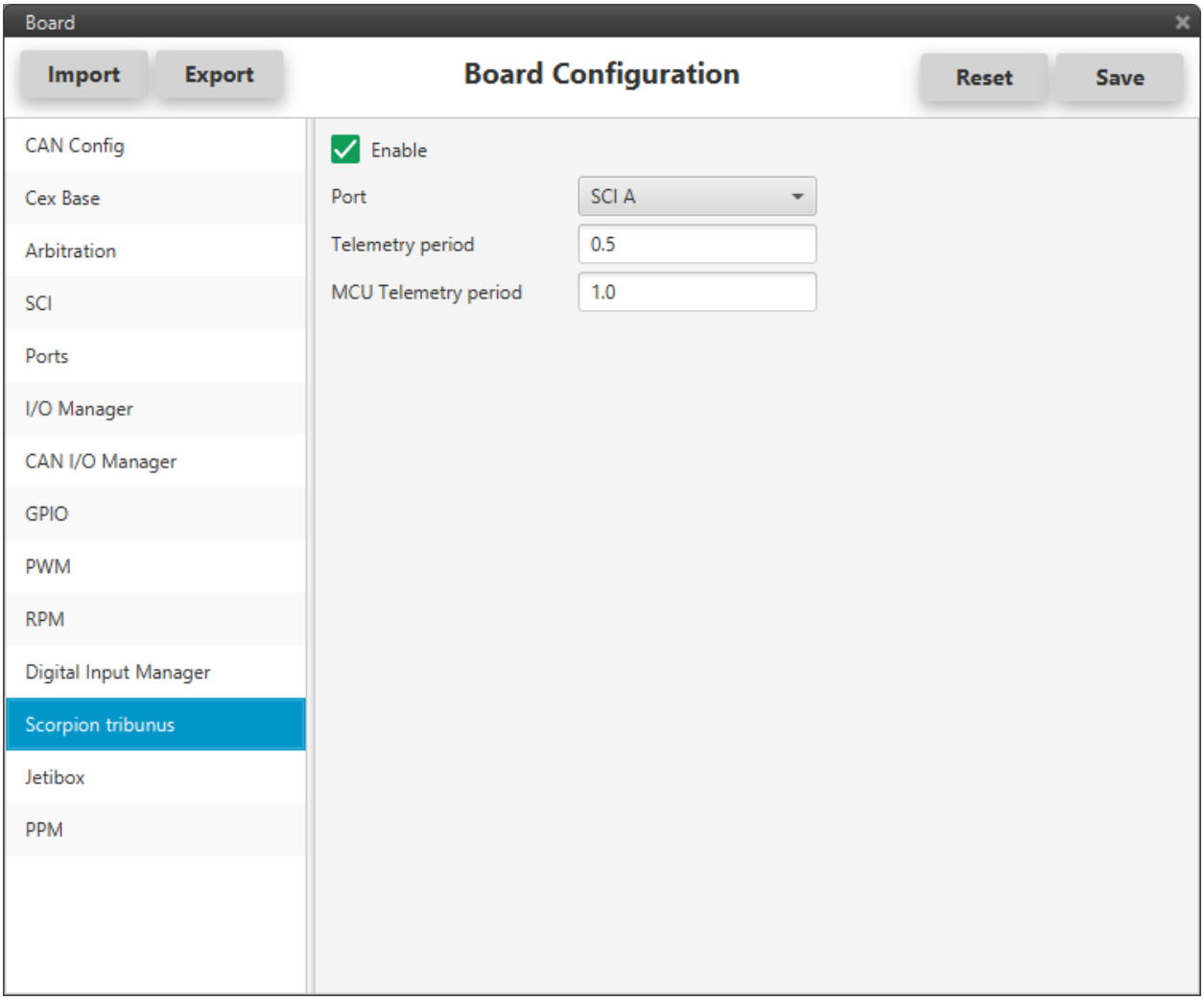
CEX can be configured to read PPM using a wide range of possibilities:



4.11 Scorpion Tribunus Telemetry

CEX can read telemetry from Tribunus ESCs by connecting it to one of its serial ports. Note that the serial port will be totally reserved for this, so it will not be usable to other things and the IO Manager affecting it wil be ignored.

We can configure the serial port, the telemetry period and a special MCU telemetry period for a propietary device (MCU). Telemetry data will be send using the configured CEX CAN Transmission ID.

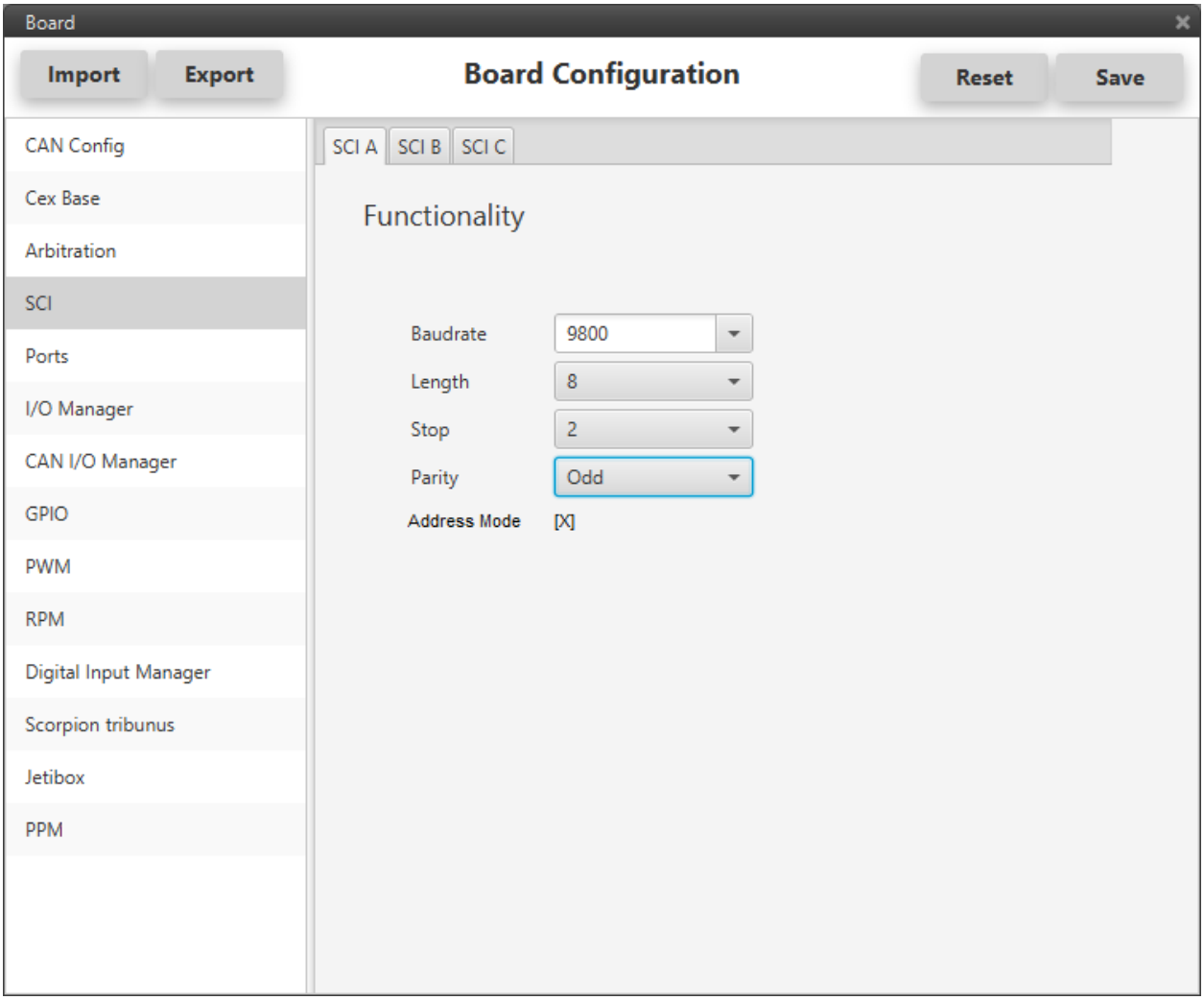


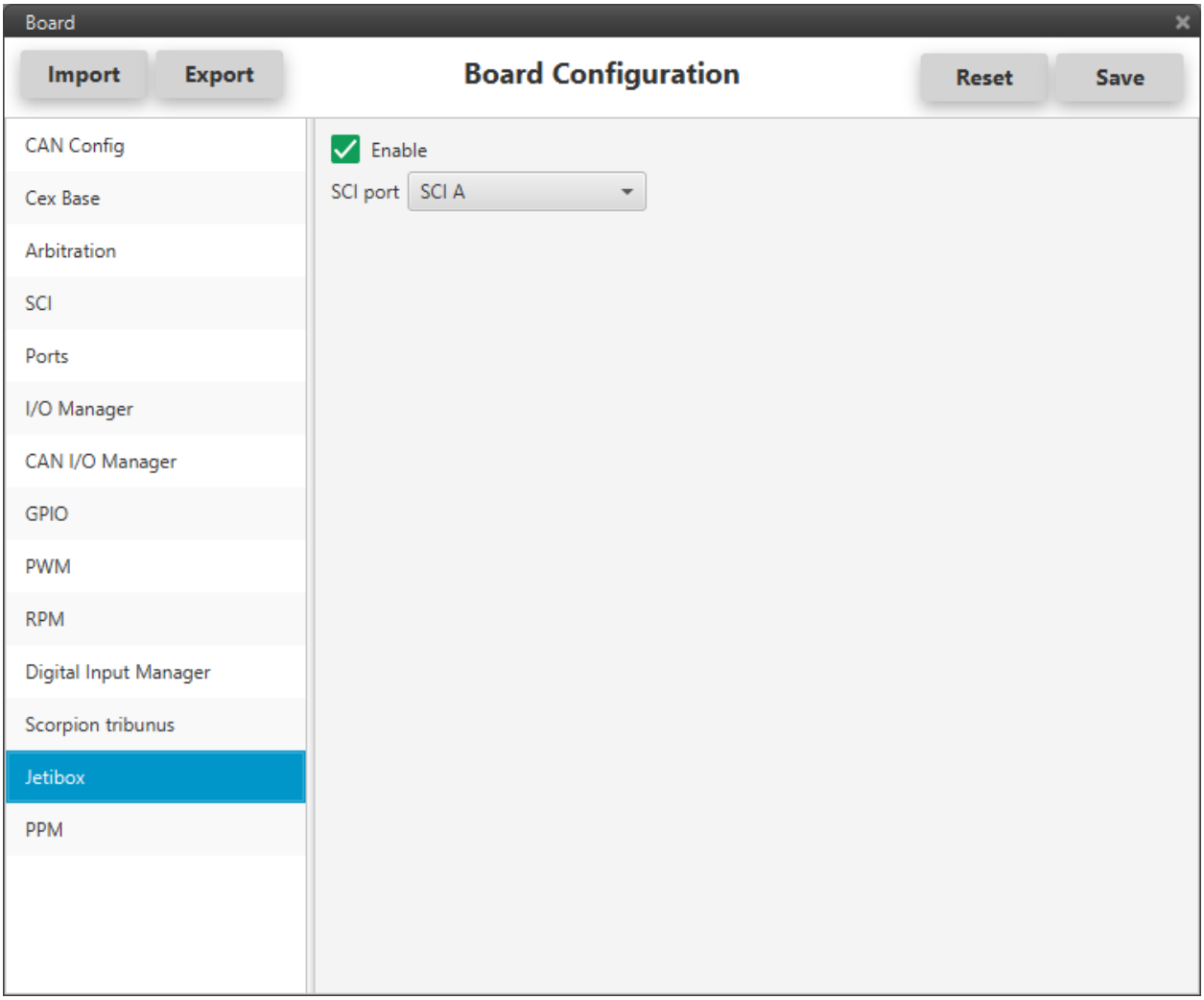
4.12 Jetibox Telemetry

CEX can simulate a Jetibox to read telemetry from legacy Jeti devices.

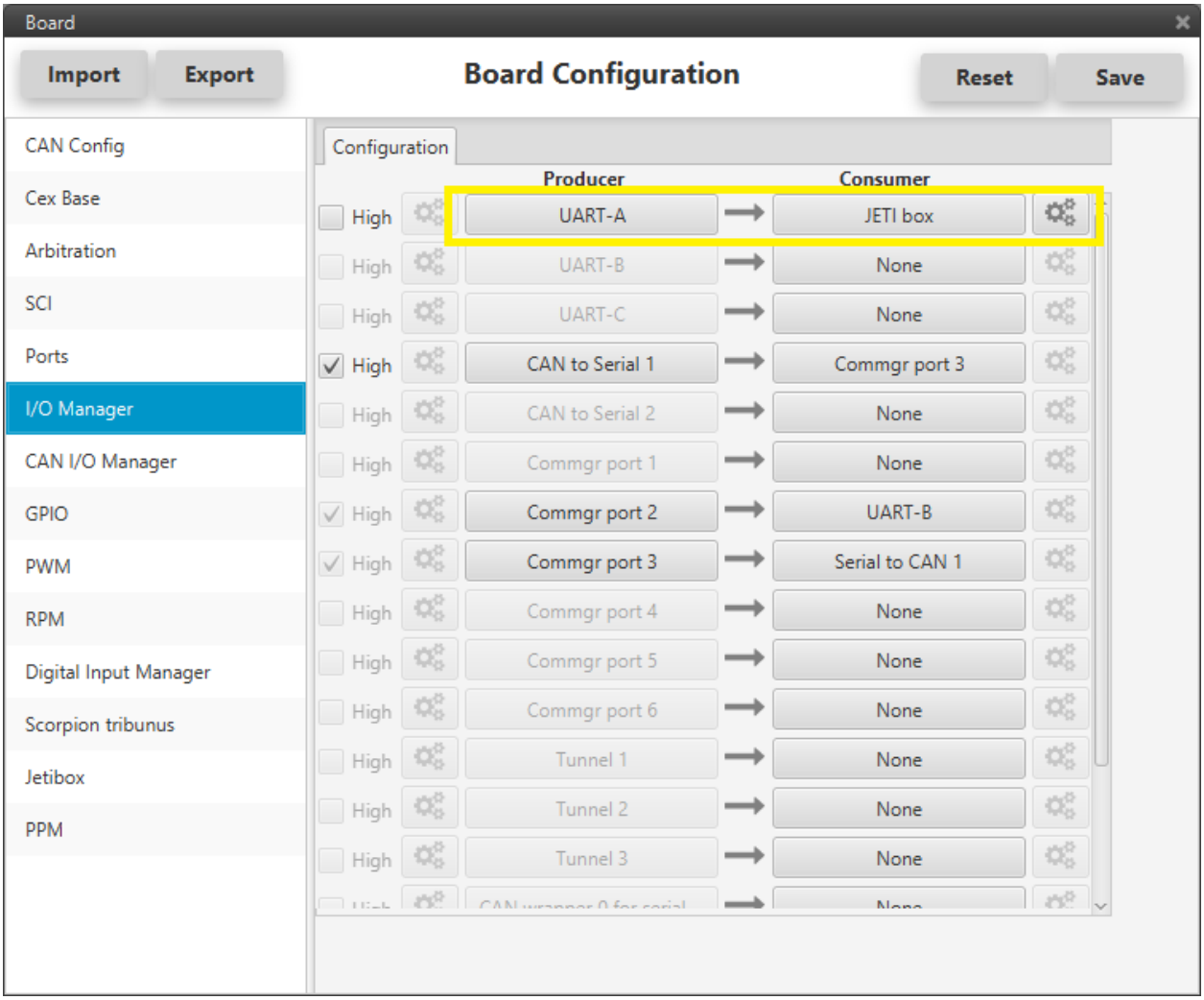
Note that the serial port will be totally reserved for this, so it will not be usable to other things and the IO Manager affecting it wil be ignored.

We shall configure the serial port with: 9800, 2 stop bits, Odd parity and address mode ON

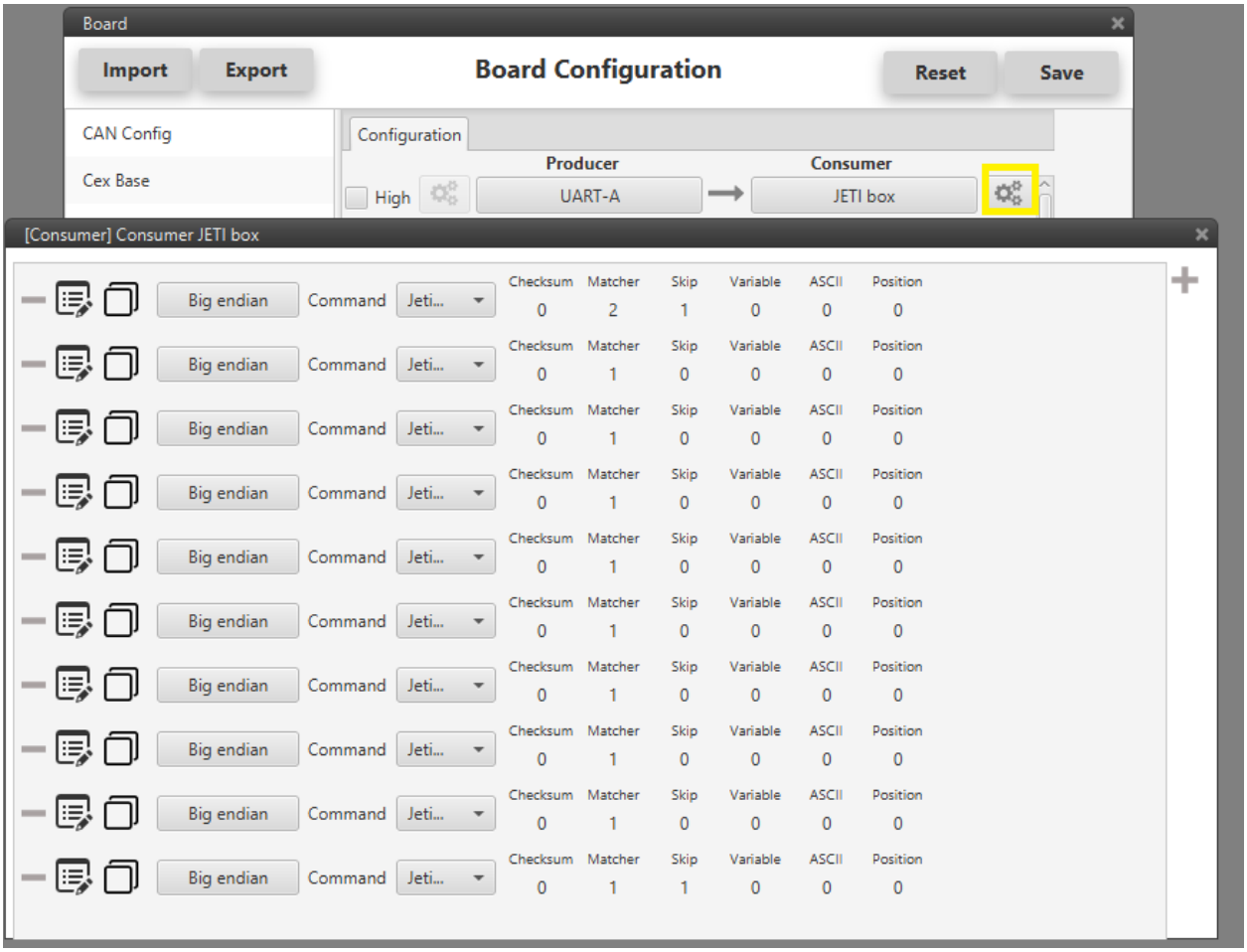

Board
Import
Export
Board Configuration
Reset
Save
CAN Config
Cex Base
Arbitration
SCI
Ports
I/O Manager
CAN I/O Manager
GPIO
PWM
RPM
Digital Input Manager
Scorpion tribunus
Jetibox
PPM
SCI A
SCI B
SCI C
Functionality
Baudrate
9800
Length
8
Stop
2
Parity
Odd
Address Mode
[X]



and link the specific Jetibox IO consumer to that port:



Also the sequence to retrieve the data shall be configured in the Jetibox consumer:



For example, to read the Actual Voltage from a Jeti MasterSpin 220 we should configure the Consumer with (use big endian in all messages):

1. Expected text: “CONTROLLER TYPE MasterSpin 220~”

   Action: Down

   - Matcher(32) “CONT” 0x434F4E54 (1129270868)
   - Skip(24*8) 192
   - Matcher(32) “220~” 0x3232307E (842150014)

2. Expected text: “MeasureOrSetting MEASURE ~”

   Action: Down

   - Matcher(32) “Meas” 0x4D656173 (1298489715)

3. Expected text: “Max Temperature”…

   Action: Down

   - Matcher(32) “Max ” 0x4D617820 (1298233376)

4. Expected text: “Min Temperature”…

   Action: Down

   - Matcher(32) “Min ” 0x4D696E20 (1298755104)

5. Expected text: “Actual Temperatu”…

   Action: Down

   - Matcher(32) “Actu” 0x41637475 (1097036917)

6. Expected text: “MaxCurrent”…

   Action: Down

   - Matcher(32) “MaxC” 0x4D617843 (1298233411)

7. Expected text: “MinCurrent”…

   Action: Down

   - Matcher(32) “MinC” 0x4D696E43 (1298755139)

8. Expected text: “Max Voltage”…

   Action: Down

   - Matcher(32) “Max ” 0x4D617820 (1298233376)

9. Expected text: “Min Voltage”…

   Action: Down

   - Matcher(32) “Min ” 0x4D696E20 (1298755104)

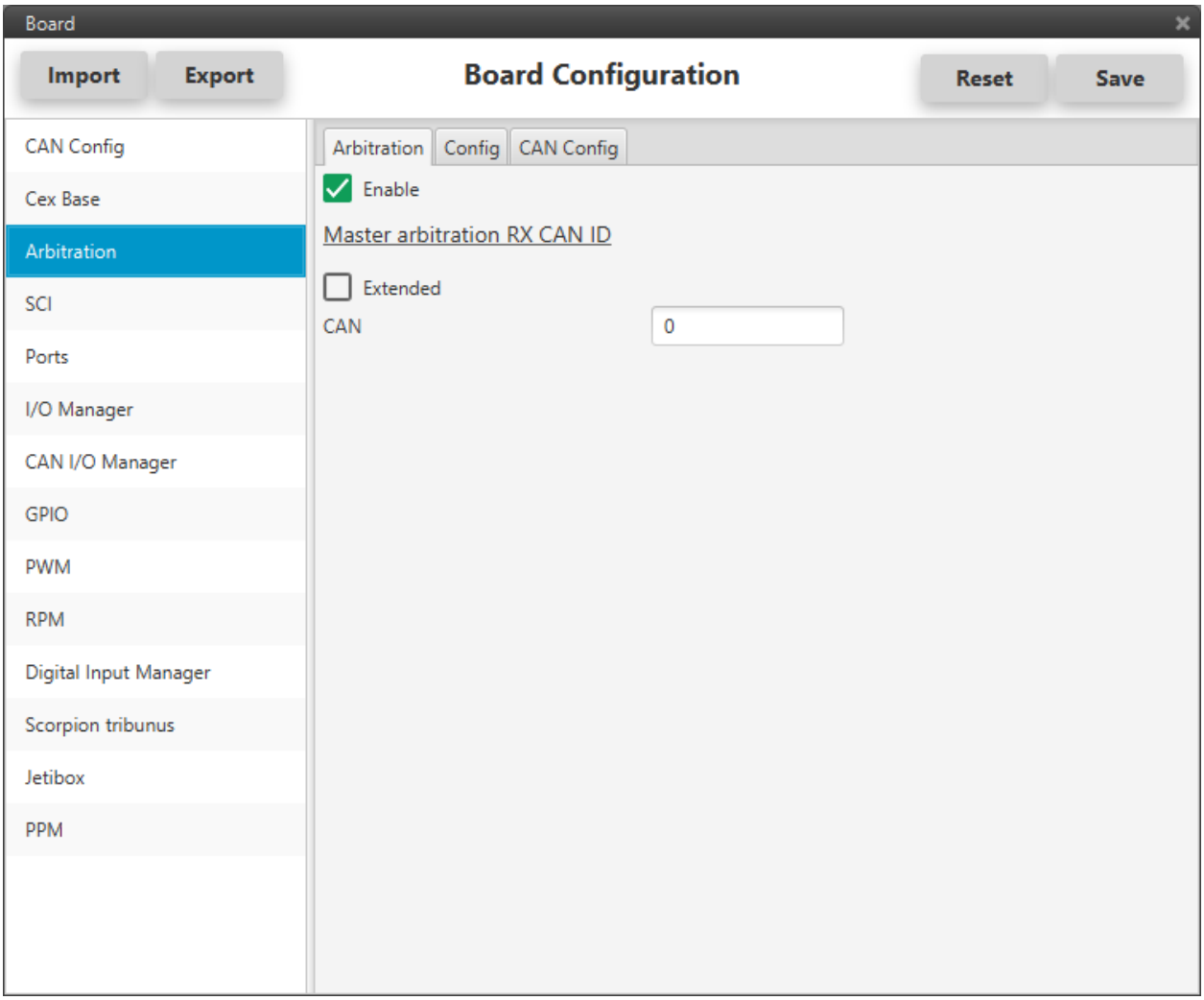
10. Expected text: “Actual Voltage 11,86 V “

    Action: Nop

    - Matcher(32) “Actu” 0x41637475 (1097036917)
    - Skip(12*8) 96
    - Ascii int(2), separartor(‘,’), decimal(2)

4.13 Arbitration

CEX is able to output PWMs using arbitration in the same way 4xVeronte does. This functionality has to be enabled as follows:



Master arbitration RX CAN Id is exclusive for CEX and specifies if we want be in synchronization with a 4xVeronte arbitration.

If enabled, when an arbitration message is received from the 4xVeronte, the selected Autopilot will be updated from the data received.

The “Config” and “CAN Config” sections work the same as in the 4xVeronte and are explained in depth in its manual.

CEX is a sophisticated tool that allows multiple ways to communicate various systems.

From version 5.42.x, it can be configured without the need of reflashing it.

CHAPTER

FIVE

VERONTE PIPE CONFIGURATION

5.1 Direct Connection

By default CEX can stablish VCP communications over its SCI-A and SCI-B ports (RS232 nd RS485 on the MC version). Using any of these connection will be possible to connect it to a PC and establish communication with PIPE software.

5.2 Connection via Veronte's CAN

It is usual to have a CEX in a system that does not allow to directly connect CEX to a PC. In that situation, we can configure a Veronte that is connected over CAN with CEX, to be able to stablish a connection between Pipe and CEX.

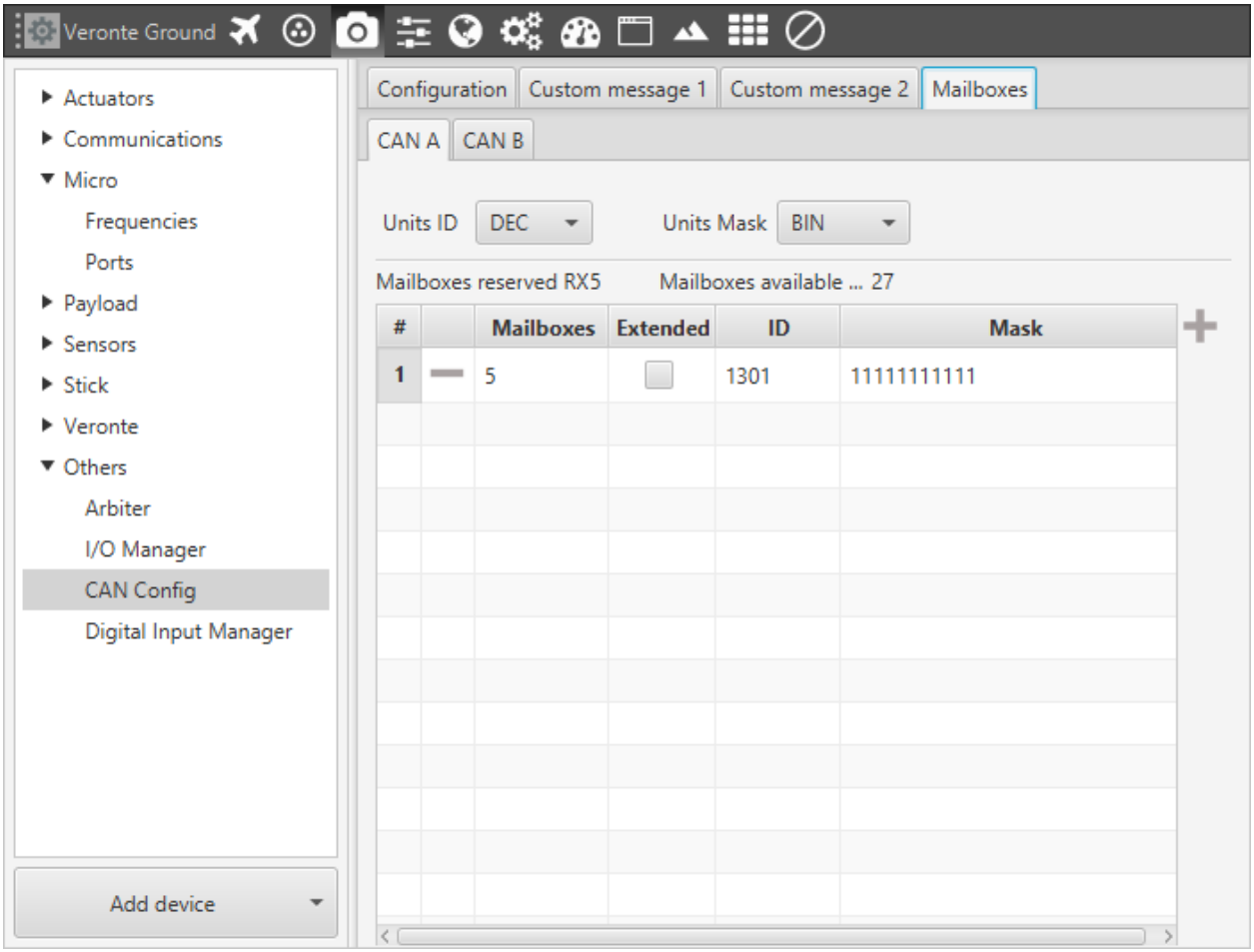
By default CEX as a Serial-Over-CAN connection configure, using **Standard** CAN ids:

- Tx CAN Id: 1301
- Rx CAN Id: 1302

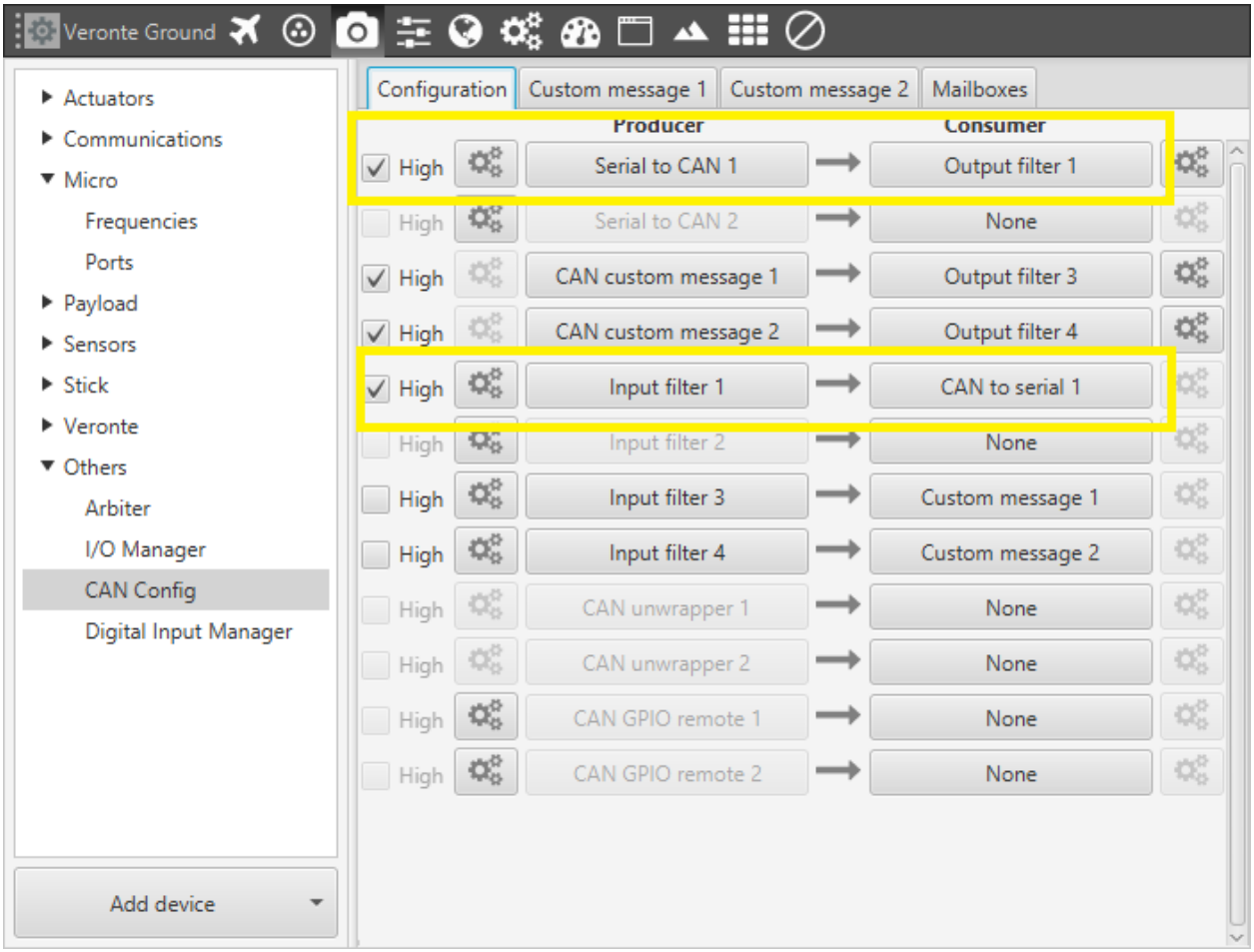
Veronte shall be configured as follows:

- One Serial-Over-CAN having:
  - Rx CAN Id: 1301 (With at least 5 mailboxes reserved)
  - Tx CAN Id: 1302

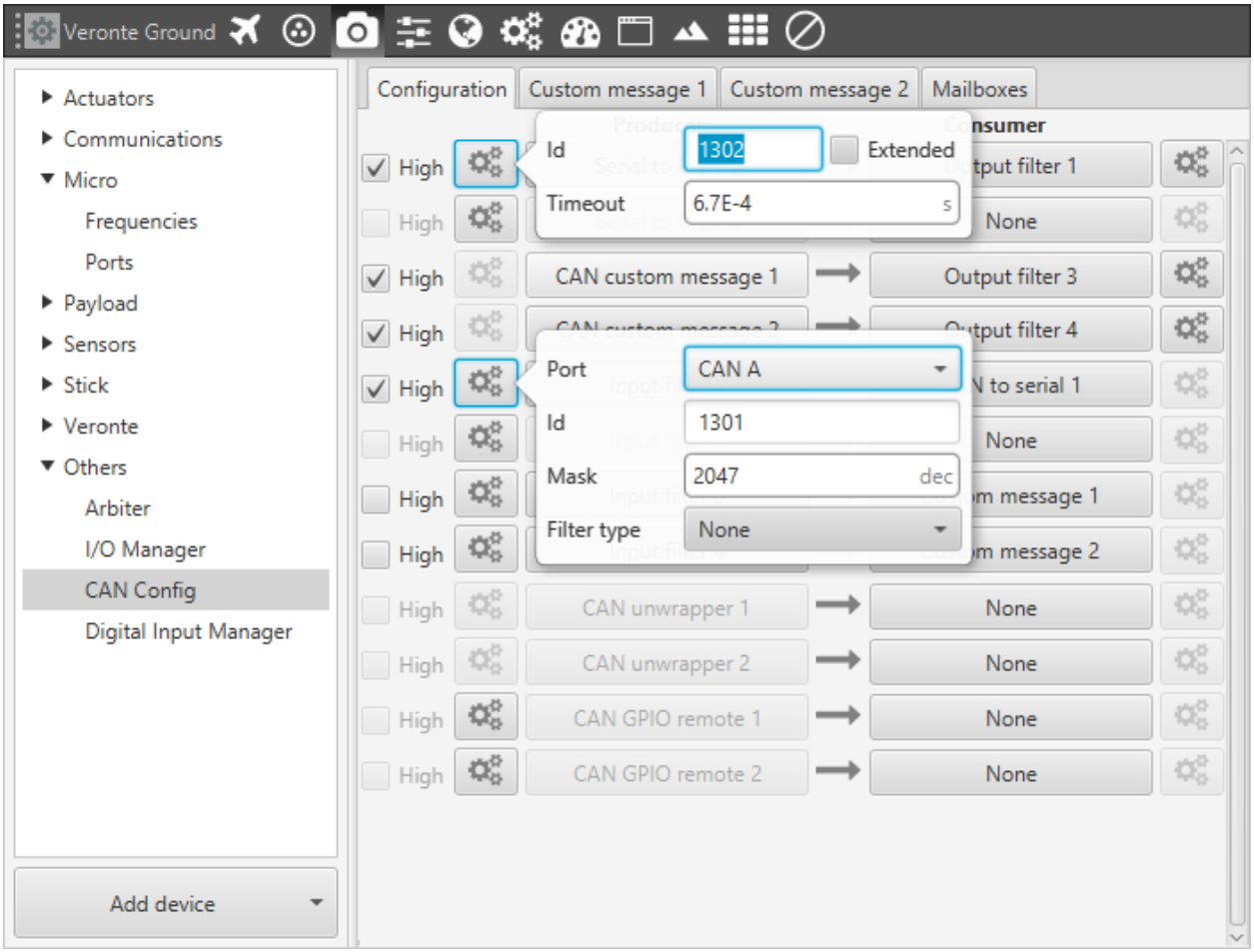
5.2.1 CAN Config Section



SerialCAN Mailboxes

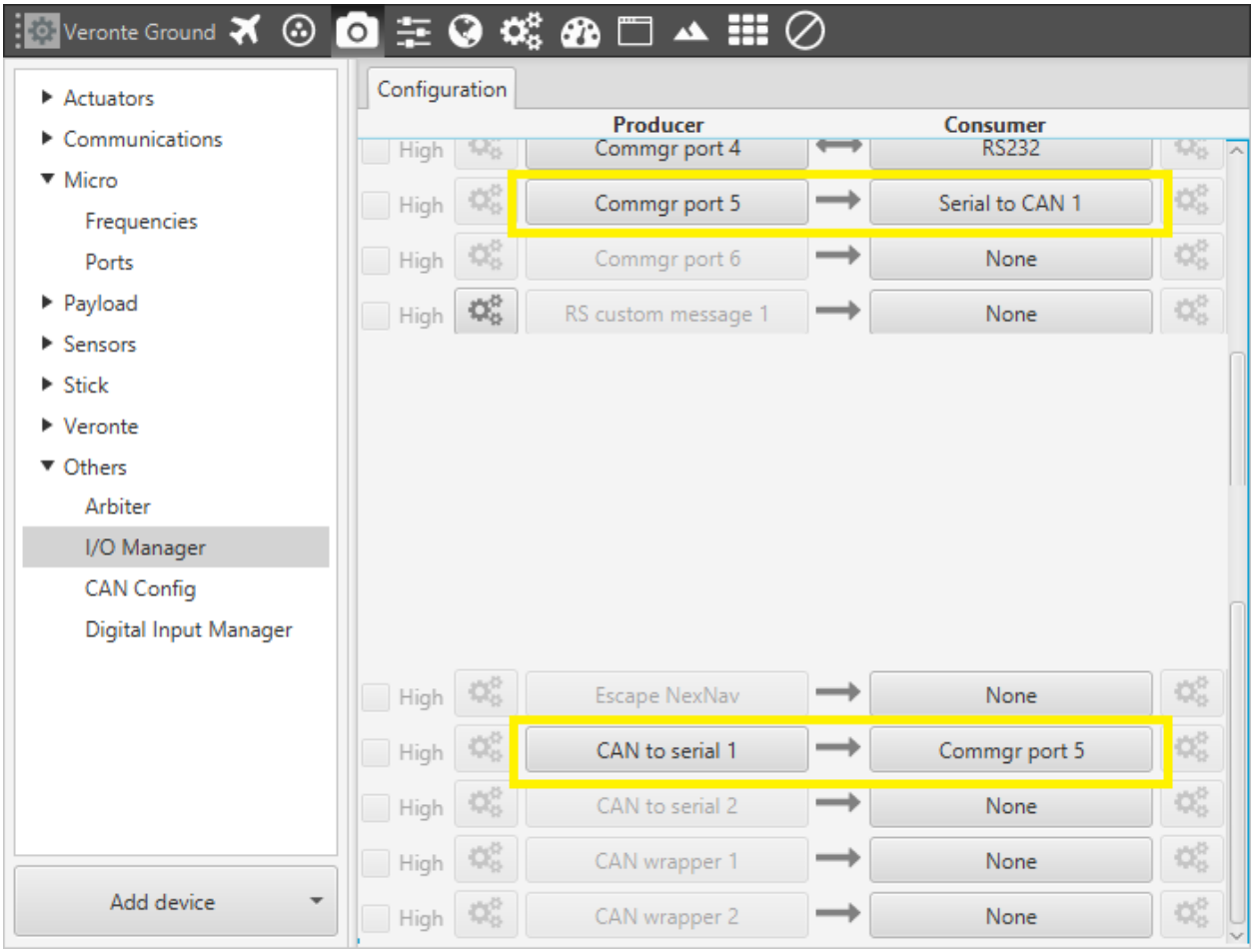


SerialCAN CAN Connections



SerialCAN CAN Configuration

5.2.2 IO Manager Section

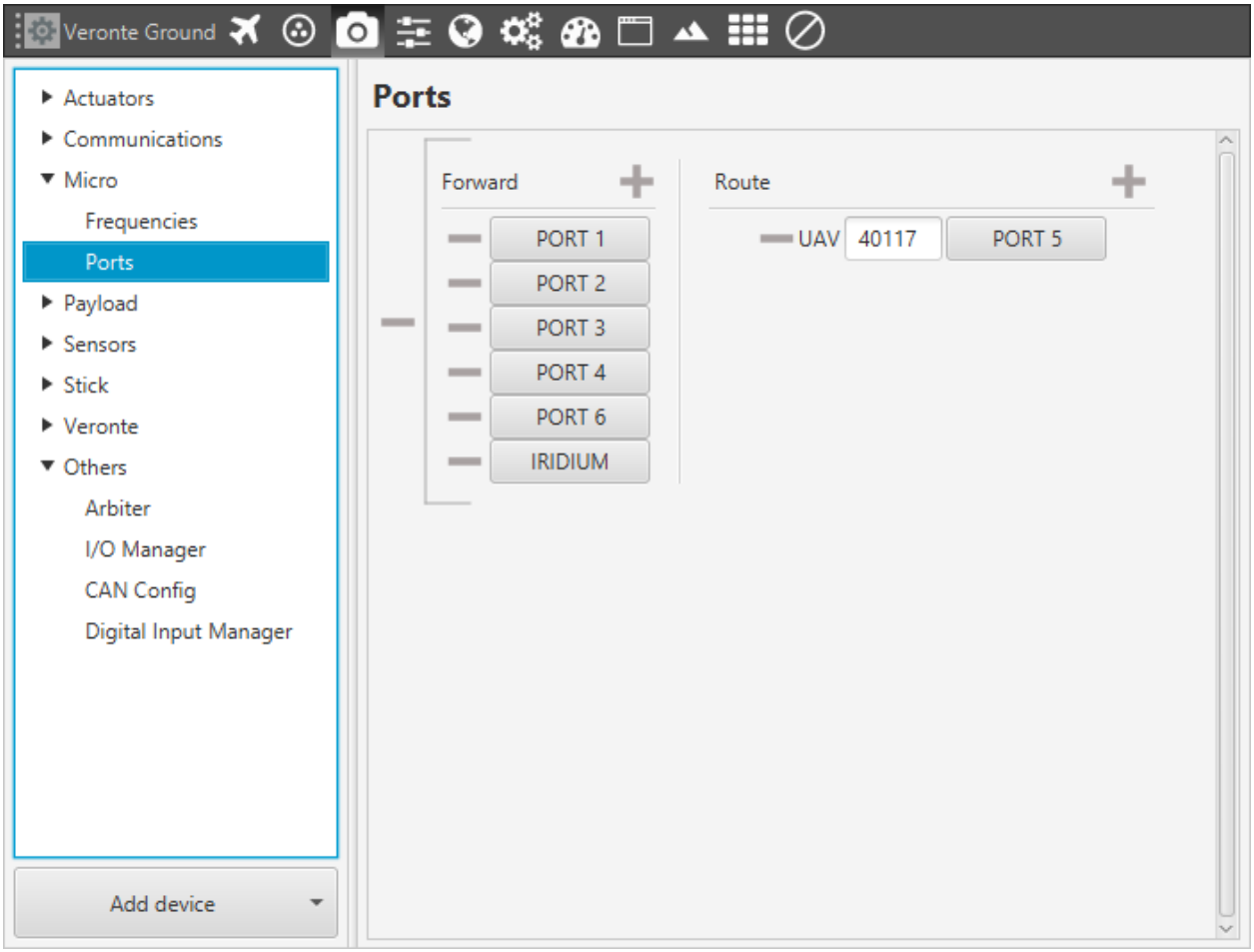


IO SerialCAN Connections

5.2.3 Ports

This step is not always necessary, but will improve the communications. Add it if there are lags or the communication does not work.

The address of target CEX shall be set correctly in the route destination UAV (40117 in this example). If the theorical address does not work, 999 (unknown) can be used as sometimes the address has not been set in CEX.



Ports Configuration

This section describes how to configure Pipe and Veronte to be able to communicate with CEX.

CHAPTER

SIX

EXAMPLES

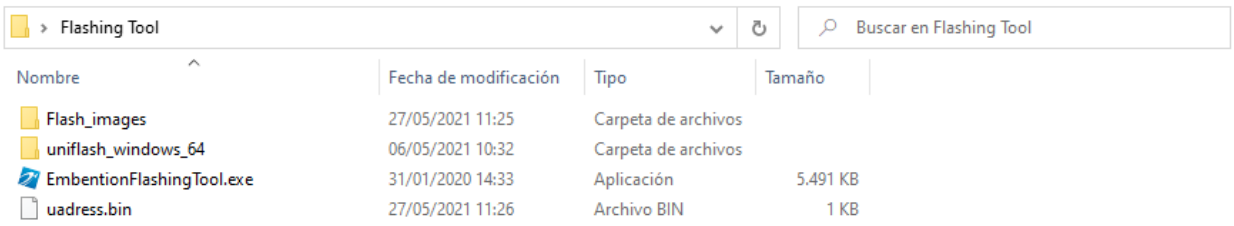
6.1 CAN Expander Update

Required items:

- Veronte CEX or CEM.
- Embention Flashing Tool.
- CAN Expander Update file.
- CAN Expander Programming tool (JTAG).

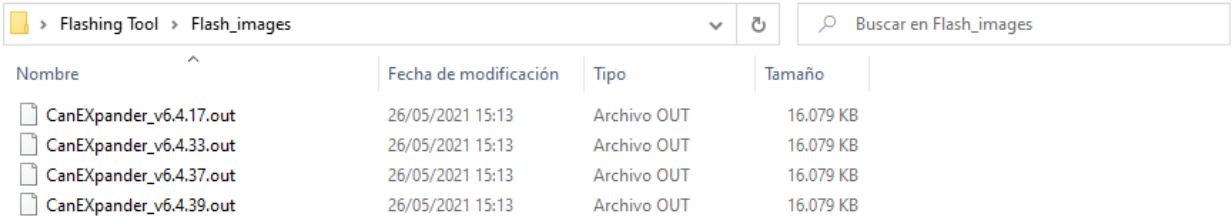
6.1.1 Embention Flashing Tool

1. Uncompress the Flashing_Tool.zip file received by Embention.
2. Connect the CAN Expander Programming tool (JTAG) to CEX.
3. Be sure to keep the following files structure:



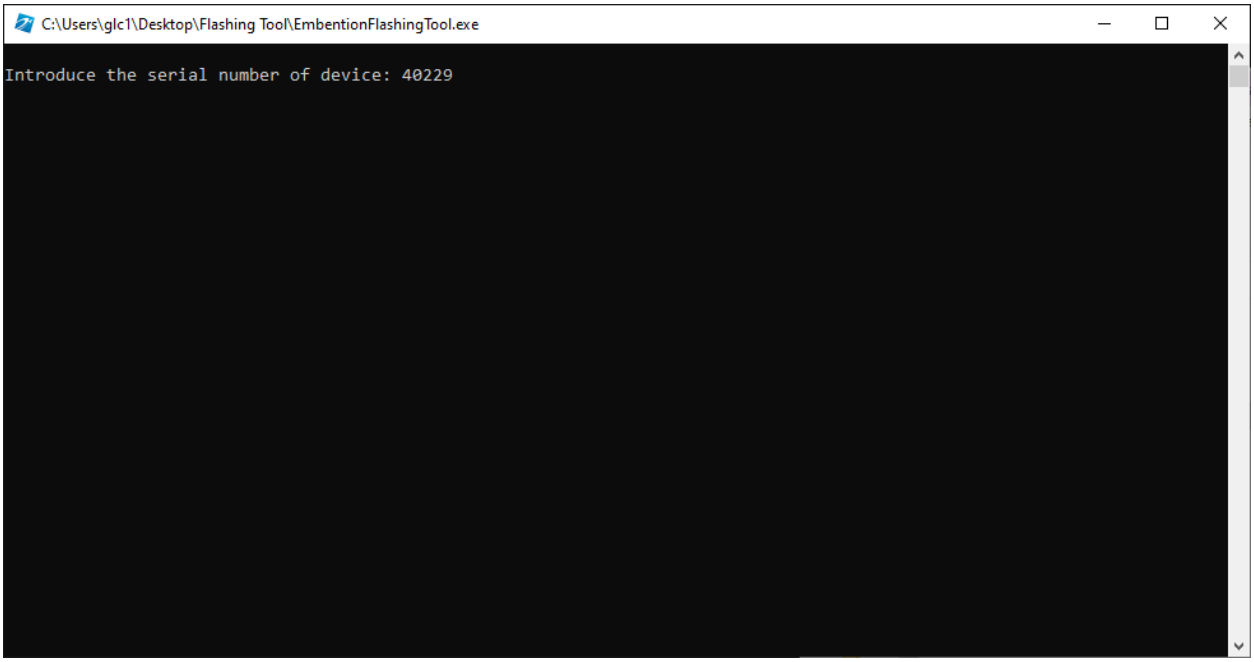
Flashing Tool folders

4. Copy the Update file (CanEXpander_v6.4.X.out, where X refers to the CEX firmware version) inside the Flash_images folder.



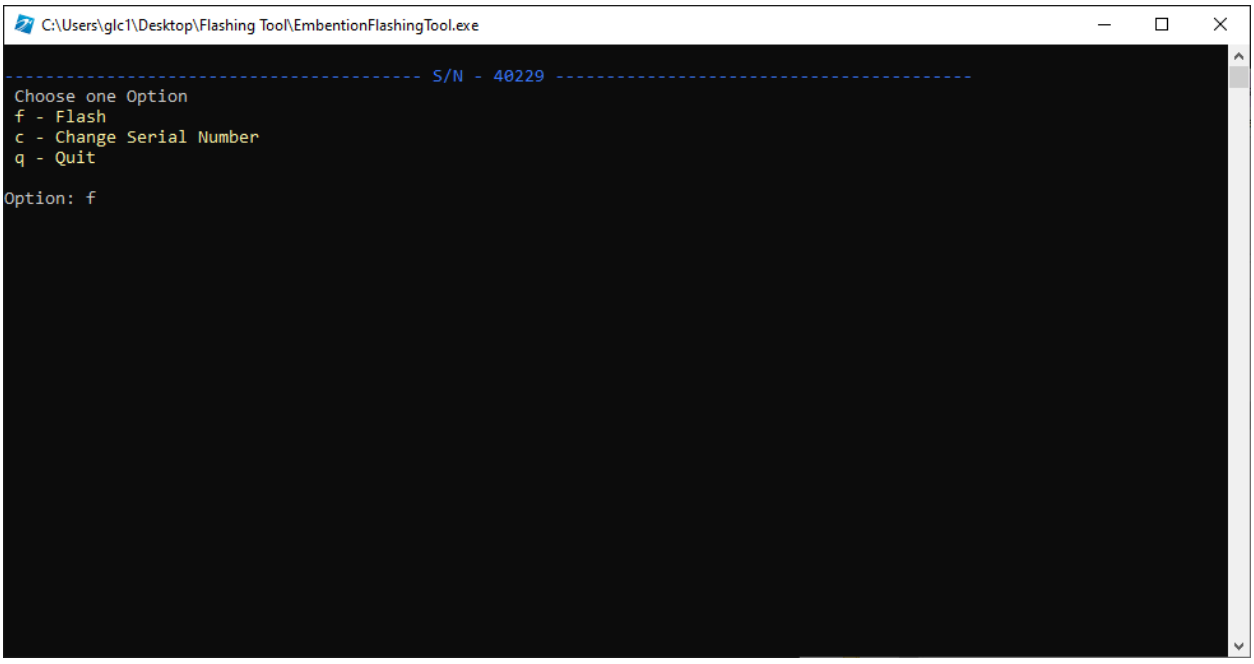
Flash images

1. Open the EmbentionFlashingTool.exe file in order to launch the program. The following window will show up:



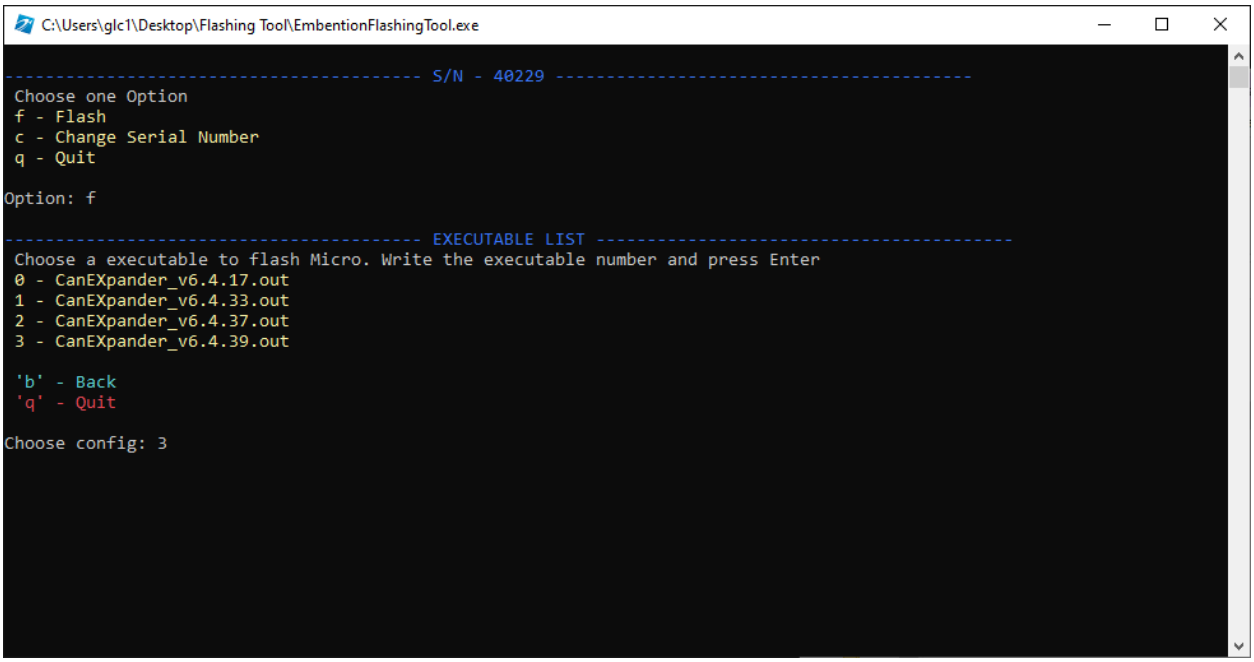
Flashing Tool 1

6. Set the CAN Expander ID (remember that it is composed by **40000 + CEX SN**). Tap Enter.
7. Choose between Flash, Change serial number and Quit options and tap Enter.



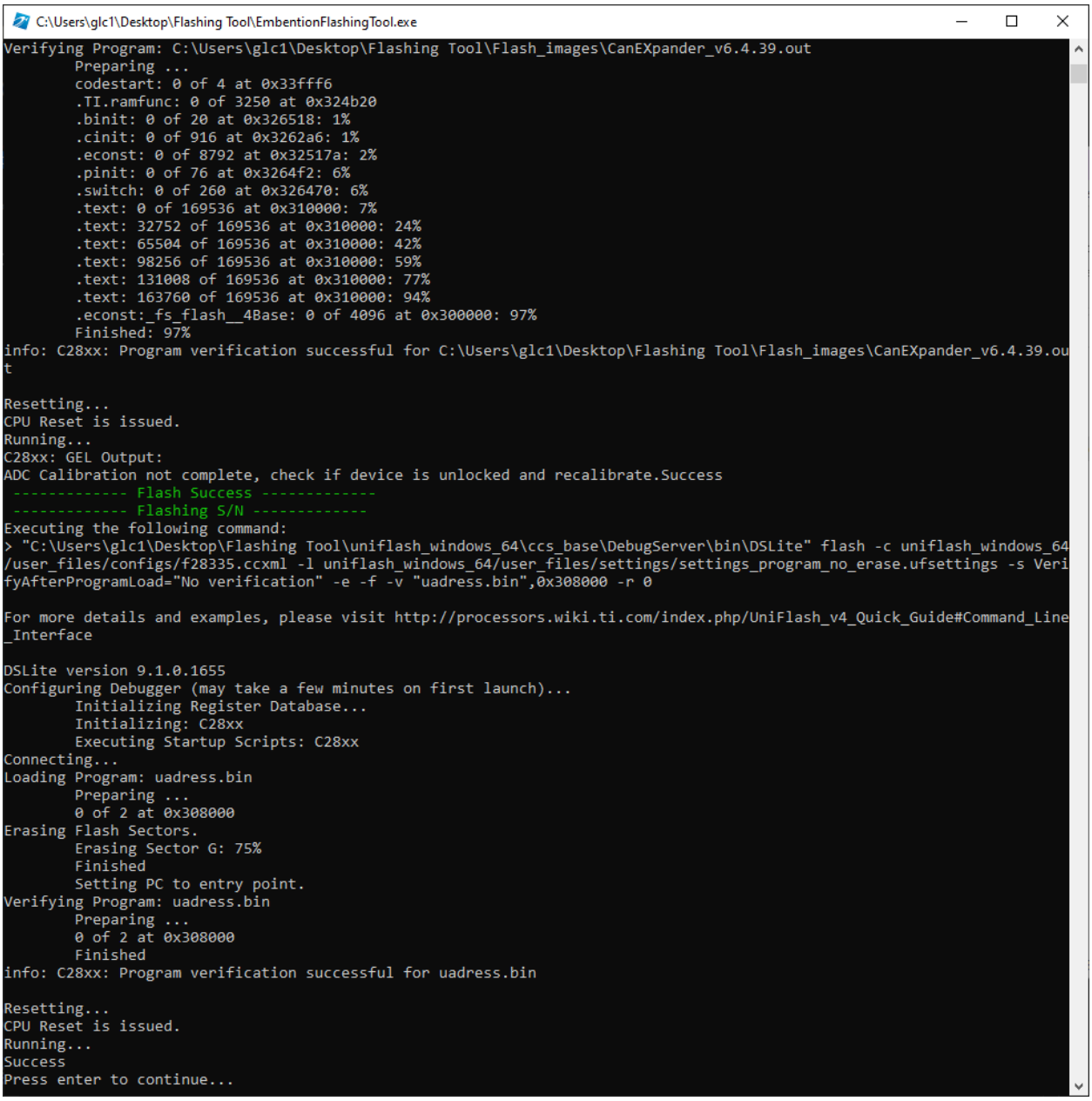
Flashing Tool 2

8. Once the Flash option is selected, the user must select the XXX.out file that refers to the desired firmware version, from the executable list. This list can be modified in the Flash_images folder.



Flashing Tool 3

1. Wait until the end of the flashing process. A “Success” message will show up if the .



Flashing Tool 4

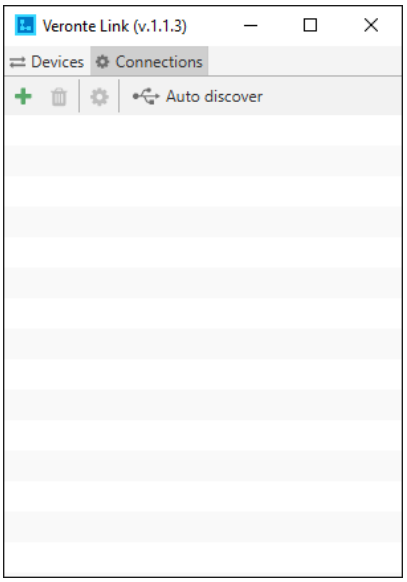
6.2 CEX configuration over CAN Bus

Required items:

- Veronte Link (v1.1.3 or higher).
- CEX PDI Builder (v6.4.47 or higher).
- Veronte CEX or CEM flashed with v6.4.35 or higher.
- Veronte Autopilot (v6.4.22 or higher) configured as described in the previous section.

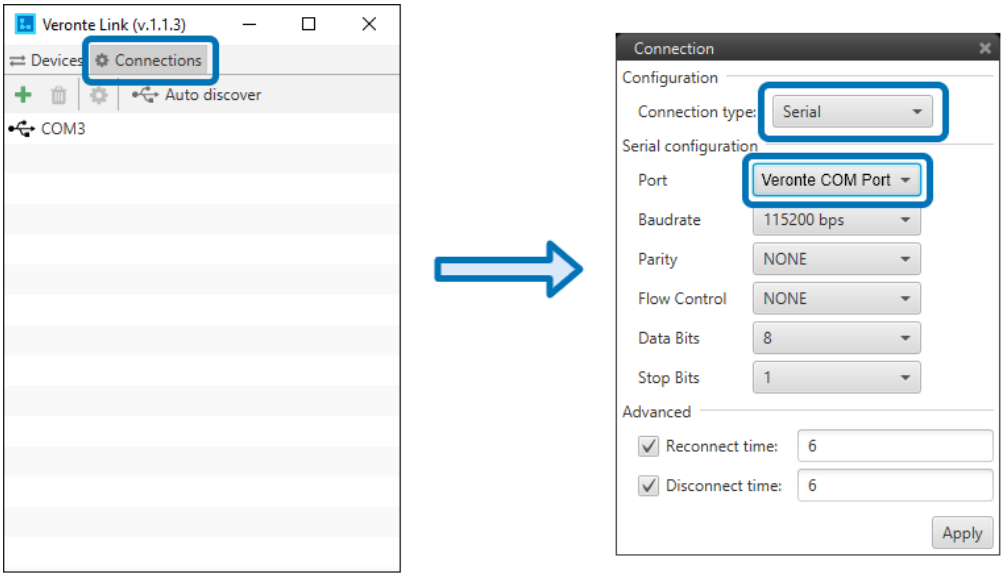
6.2.1 Veronte Link setup

1. Install and open Veronte Link. Close all the Veronte Pipe open windows or disable tghe “Autodiscovery” option and remove all the COM Ports configured in its Connections tab.



Veronte Link

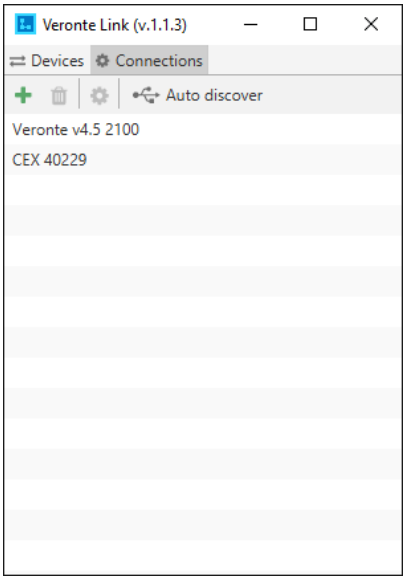
2. Click on Connections, then select “Serial” and the COM Port that Veronte Autopilot is using. CLick on Apply to save these settings.



Veronte Link Connections

3. Open the Devices tab and confirm that Veronte has shown up. The Refresh button can be used in case the devices list is not automatically updated.

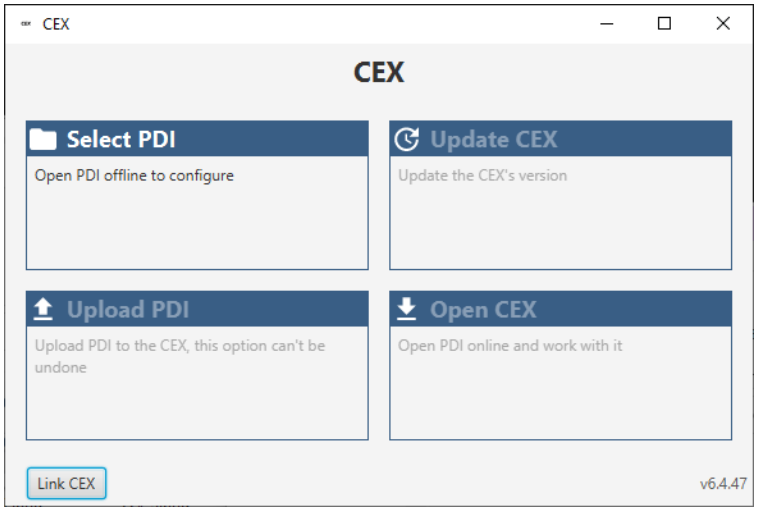
4. Click then on the "Find Device" button and input the CAN Expander ID. Remember that CEX or CEM ID is built as **40000 + CEX Board serial number (SN)** (for instance, board 229 will use ID 40229).
5. Once found, CAN Expander will show up in the Devices list together with Veronte.



Veronte Link Devices

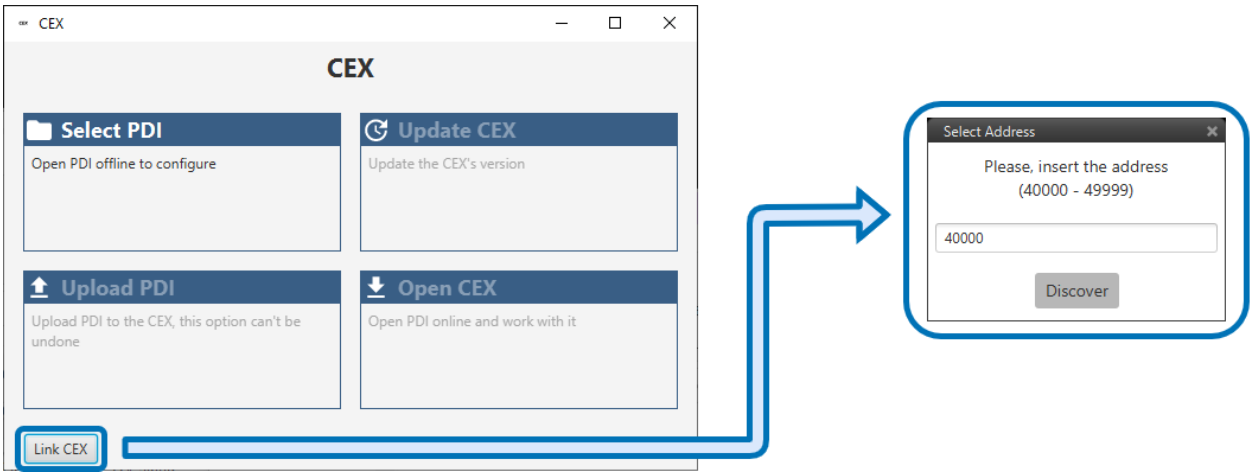
6.2.2 CEX PDI Builder

1. Once Veronte and CEX have been detected on Veronte Link, install and open CEX PDI Builder.



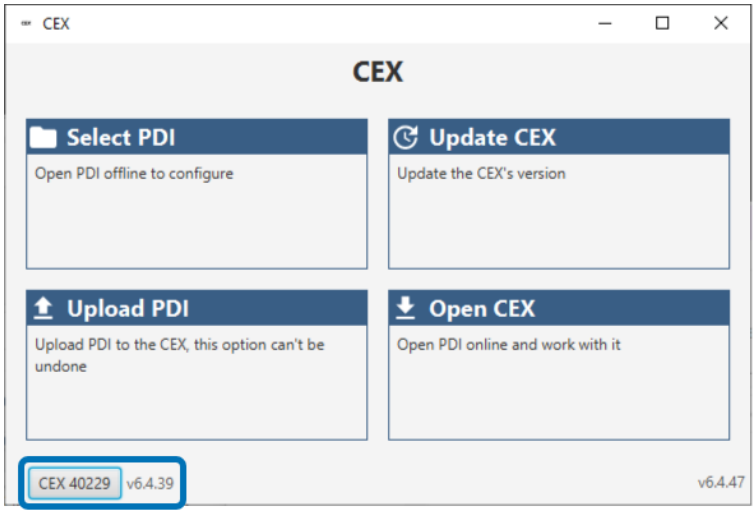
CEX PDI Builder

2. Click on "Link CEX" and enter the CAN Expander's ID.



CEX PDI Builder - Address

Once connected, CEX/CEM firmware version and its ID are shown in the lower part of the same window.



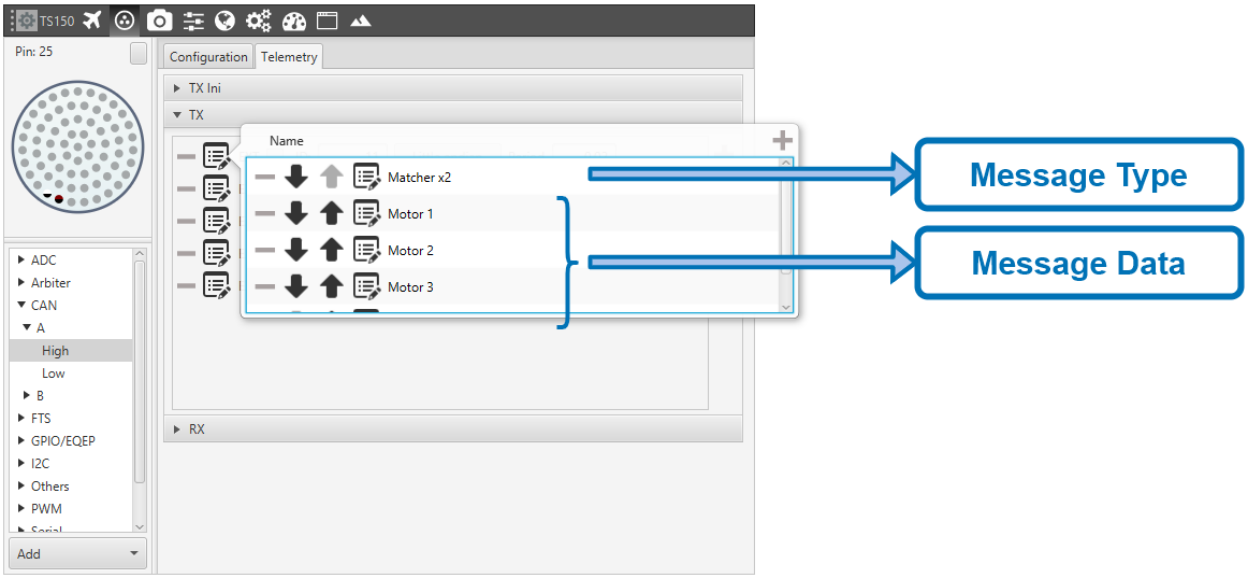
CEX PDI Builder linked

The user can access now to four configuration options:

- **Select PDI:** A previously exported CAN Expander PDI can be opened and modified offline.
- **Update CEX:** not in use, under development.
- **Upload PDI:** A previously exported CAN Expander PDI can be imported to the linked CAN Expander.
- **Open CEX:** CAN Expander PDI are downloaded from the linked CEX and they can be modified online.

6.3 Sending PWMs

The PWM message format is described in the section *Command PWMs*

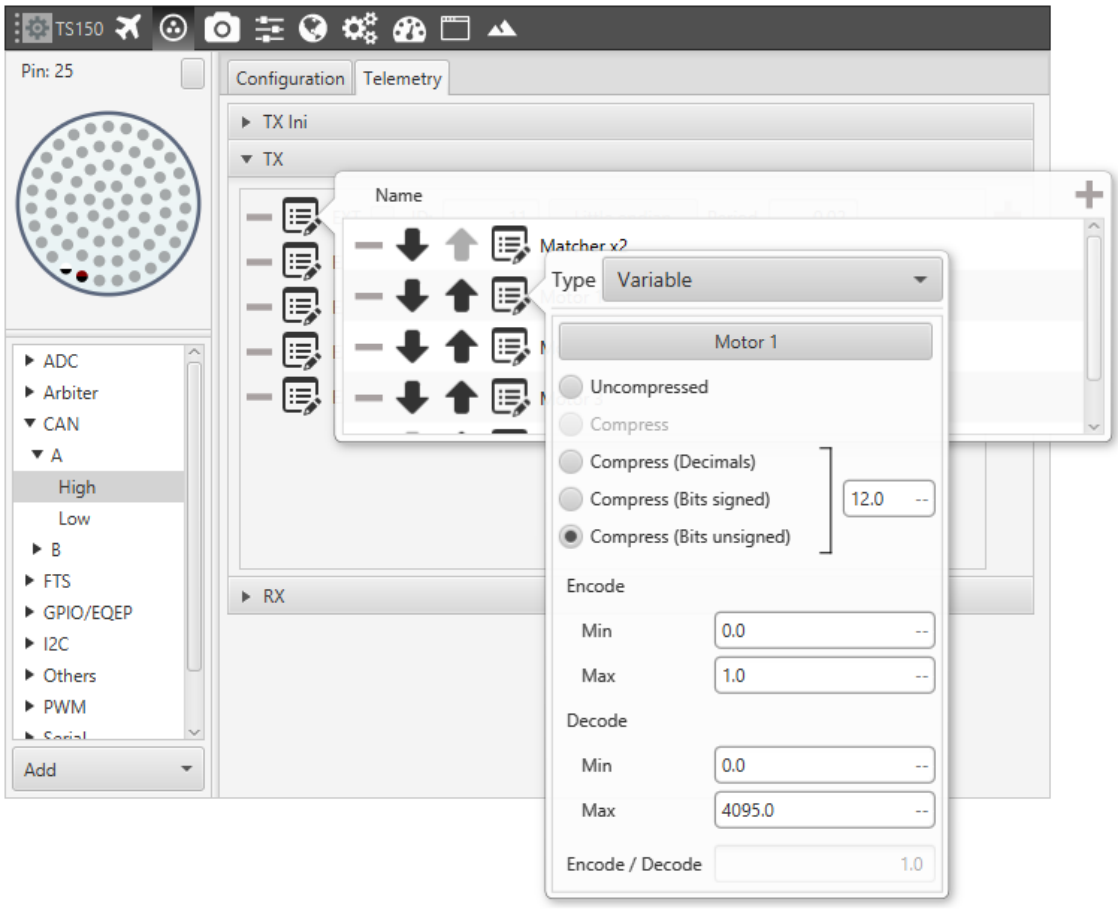


PWM Message

Inside the Veronte configuration, each PWM has to be set using the following format:

- **Compress bits unsigned:** 12
- **Encode:**: Min=0, Max=1
- **Decode:**: Min=0, Max=4095

Conversion to the current width is made taking into account the configuration.



PWM Output

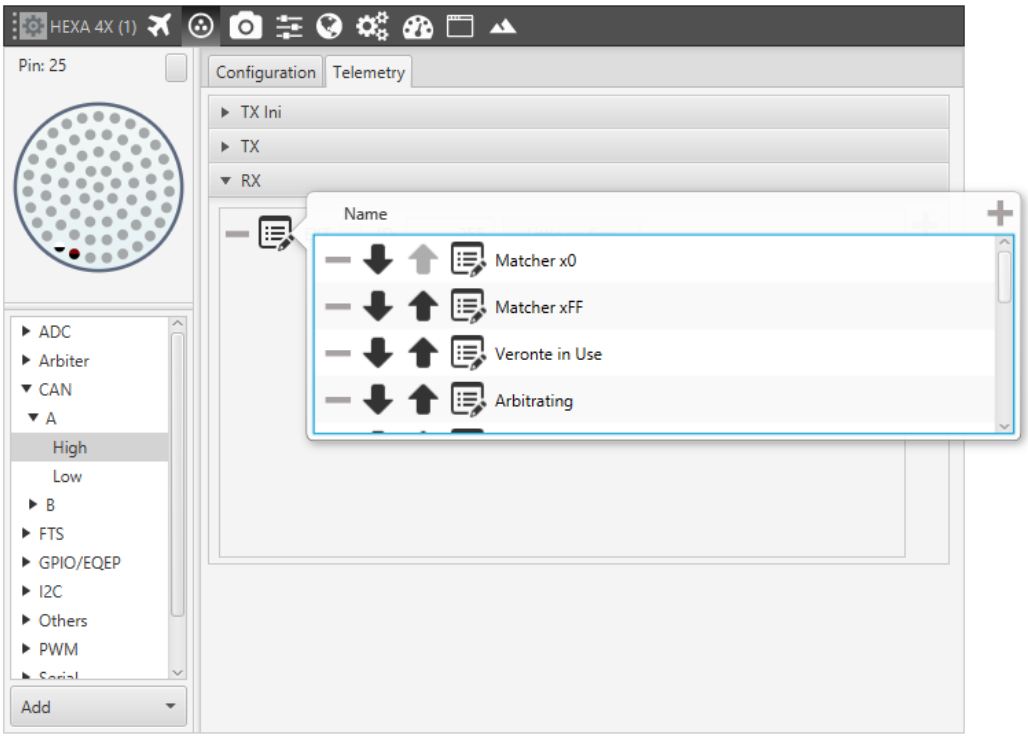
6.4 Reading Arbitration Messages

Note that those messages are generated only if arbitration and the messages are enabled in CEX.

Arbiter will send its telemetry in little endian format, using its CAN-TX ID

- Variable value message: (same as the 4XV)
  - Byte0 0 : t_arbitration message (CANstdp)
  - Byte1 N : Autopilot [0, 3]
  - Byte2-5 VVVV : Autopilot score as Float (32 bits)
- Status message: (same as the 4XV. However, 4XV sends more data)
  - Byte0 0 : t_arbitration message (CANstdp)
  - Byte1 0xFF : Status flag
  - Byte2
    - bits6-0 : Chief autopilot (current, selected)
    - bit7 : Arbitrating 0:false, 1:true

- Byte3:
  - bit0 : AP0 Alive
  - bit1 : AP1 Alive
  - bit2 : AP2 Alive
  - bit3 : AP3 Alive (external)
  - bit4 : AP0 Ready
  - bit5 : AP1 Ready
  - bit6 : AP2 Ready
  - bit7 : AP3 Ready (external)

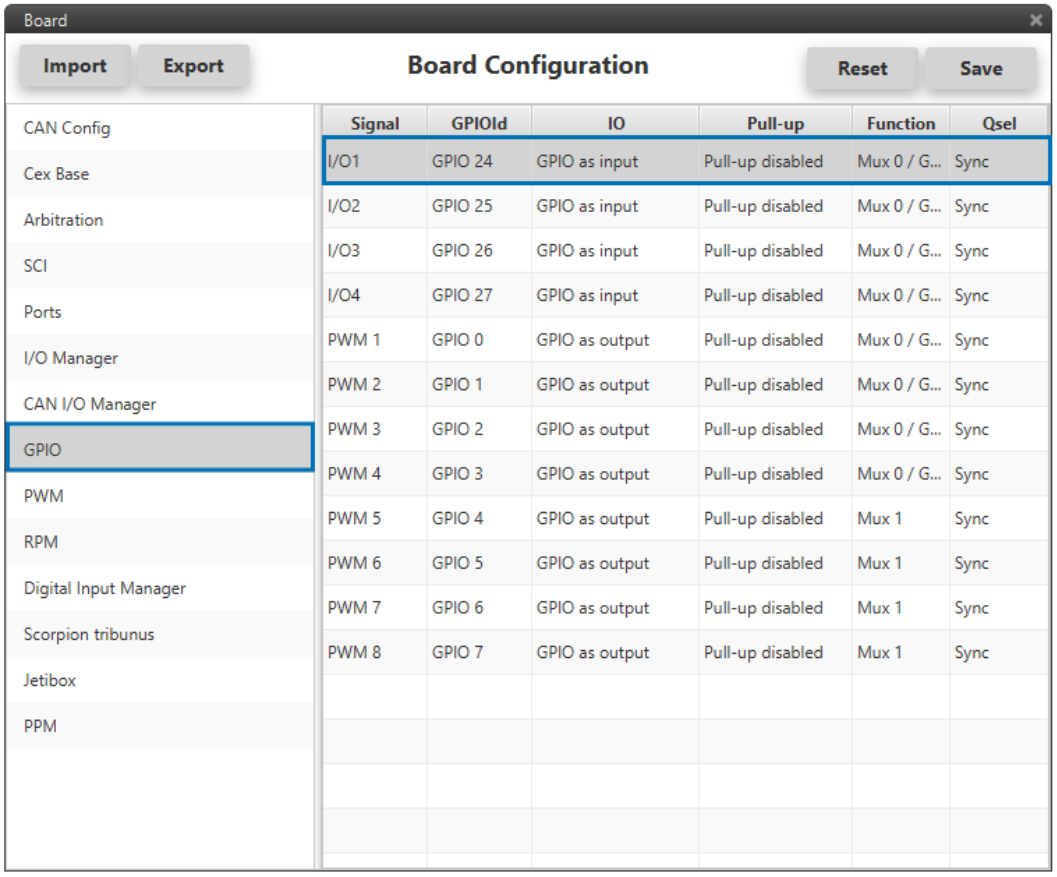


Arbiter telemetry reading

6.5 Reading RPMs

6.5.1 GPIO Configuration

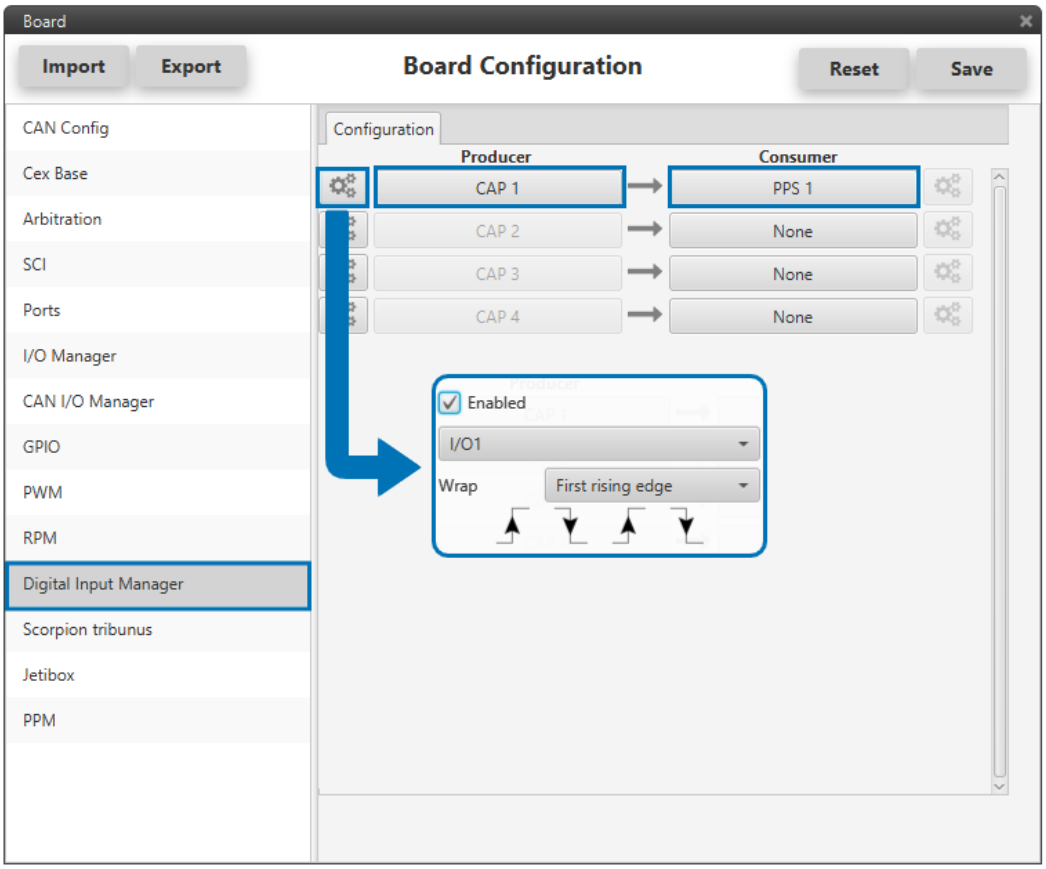
RPM can be read on the available digital inputs I/O 1-4. The chosen pin needs to be configured as “GPIO as input”. In the example shown here, I/O1 is chosen (pin 9 on OEM version; pin 10 on MC version).



GPIO Configuration

6.5.2 Digital Input Manager

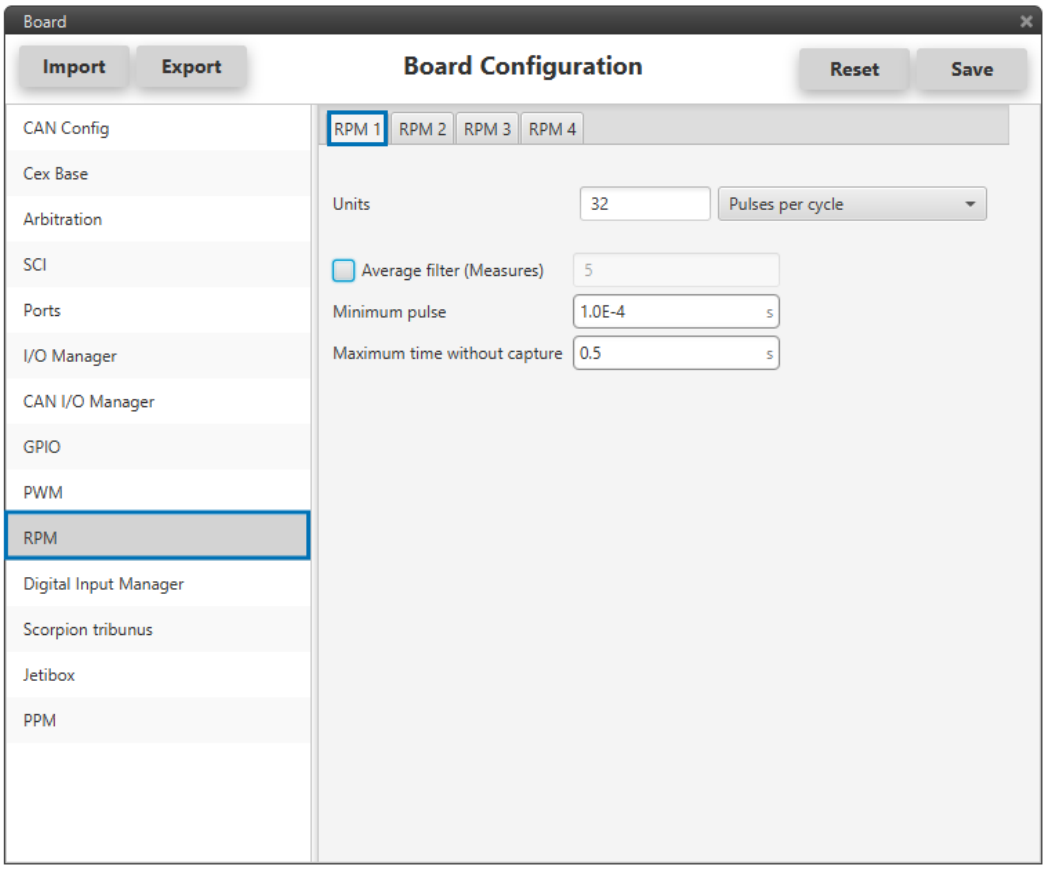
There are 4 possible producers: CAP 1 - 4. One needs to be chosen and linked to one of the RPMs consumers, PPS 1 - 4 (Pulse Per Second). Then, each chosen producer needs to be pointing to the right I/O (in this example, I/O1). Lastly, the expected pulse needs to be wrapped: for RPMs the desired options are “first rising edge” or “first falling edge”.



Digital Input Manager Configuration

6.5.3 RPM Configuration Menu

Here the expected pulse needs to be defined. There are 4 tabs (RPM 1 - 4). The user needs to select the tab according to the chosen consumer in the **Digital Input Manager**. In the example, PPS1 was chosen, therefore RPM1 will be edited.



RPM: Pulse Data

The data available is:

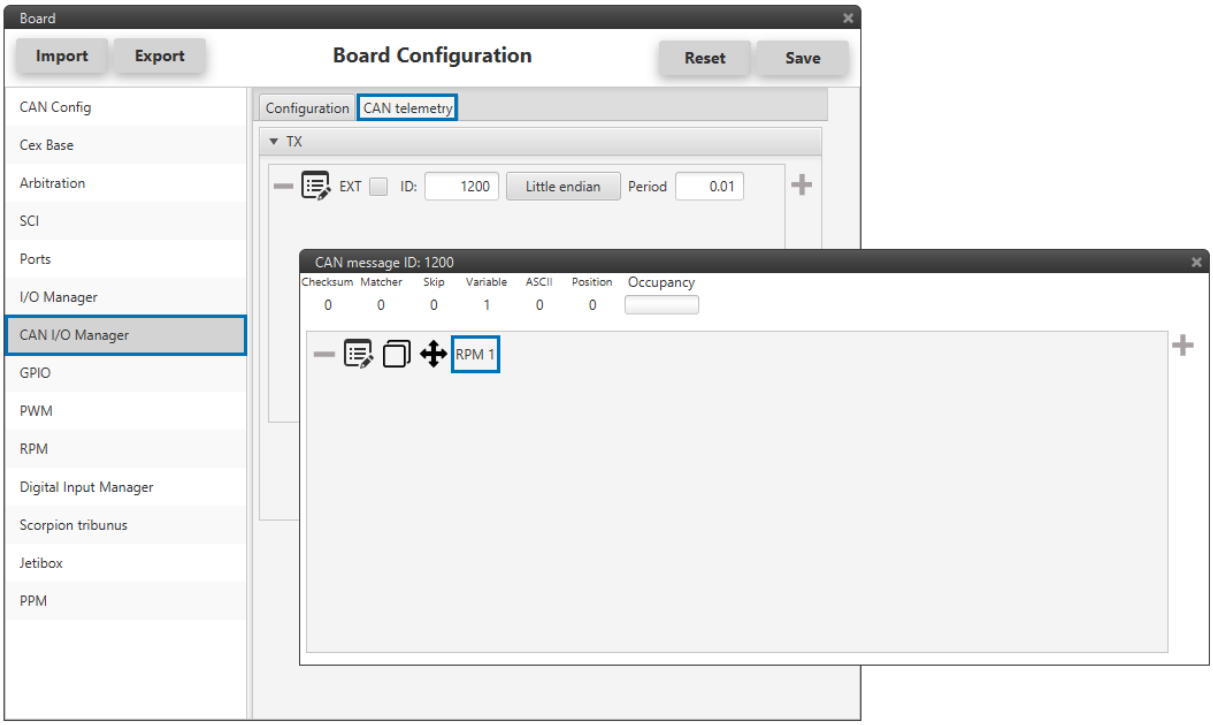
- **Units**: avaialble options are *pulses per cycle*, *radians per pulse* or *custom*.
- **Average filter**: the readout of the pulse can be filtered for the output to be an average. The amount of measurements to do the average needs to be specified.
- **Minimum pulse**: here the minimum expected pulse period needs to be specified. This will discard spurius pulses (e.g. induced by EMI) which are smaller than this minimum pulse.
- **Maximum time without capture**: if no incoming pulse is received for more than this time, the output RPMs will be 0.

6.5.4 CAN Telemetry

Last, the RPMs sensed need to be sent to the autopilot. In the **CAN I/O Manager** a new telemetry message needs to be created with its correspondent ID, endianness and period. In the example below:

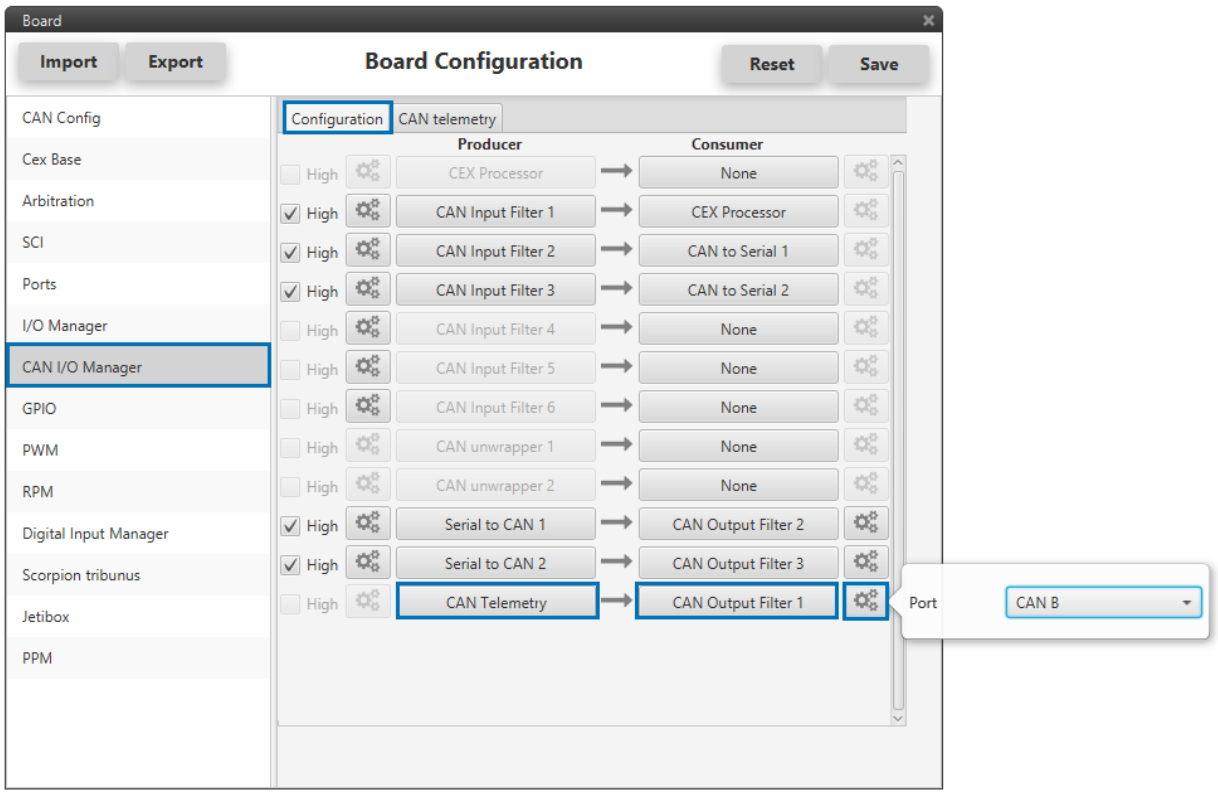
- **ID**: 1200.
- **Endianness**: little.
- **Period**: 0.01 s.

In the telemetry message one of CEX's variables needs to be selected. As we have chosen PPS1 as our consumer in the **Digital Input Manager**, the variable we need to send is **RPM1**.



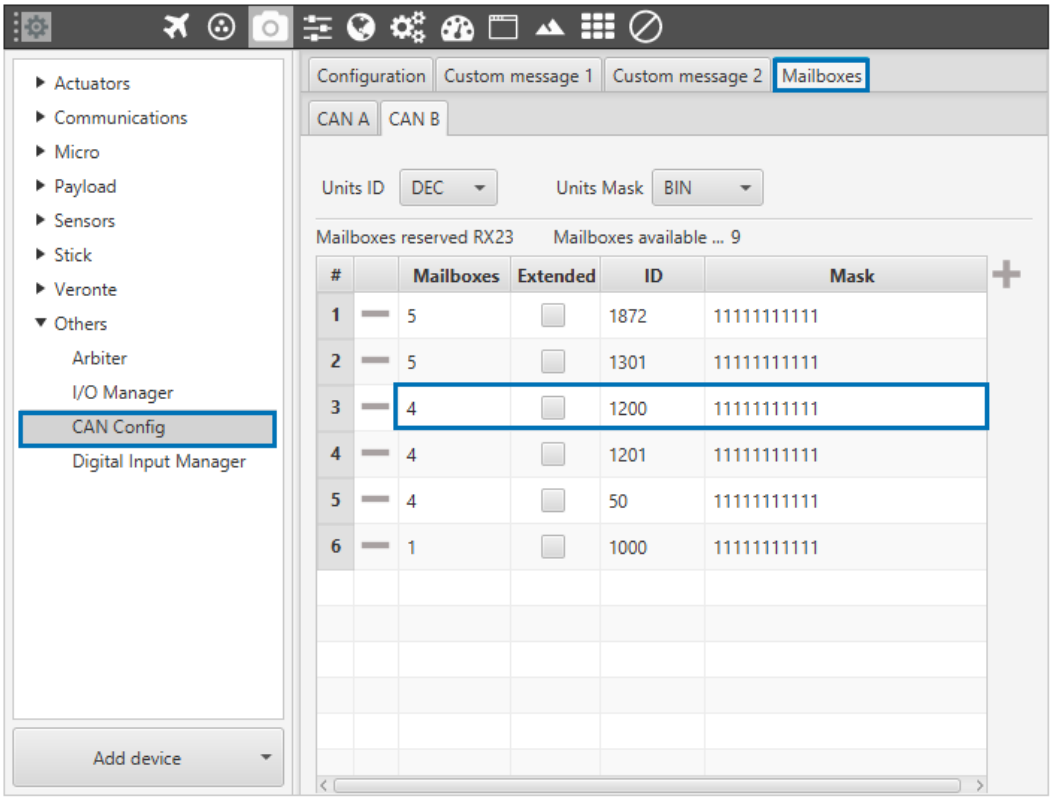
RPM: CAN Telemetry

The telemetry (producer) needs to be send over one of the avialable CAN Output Filters (consumer). In the example below, the **RPM1** variable is sent over CAN B bus of the CEX.



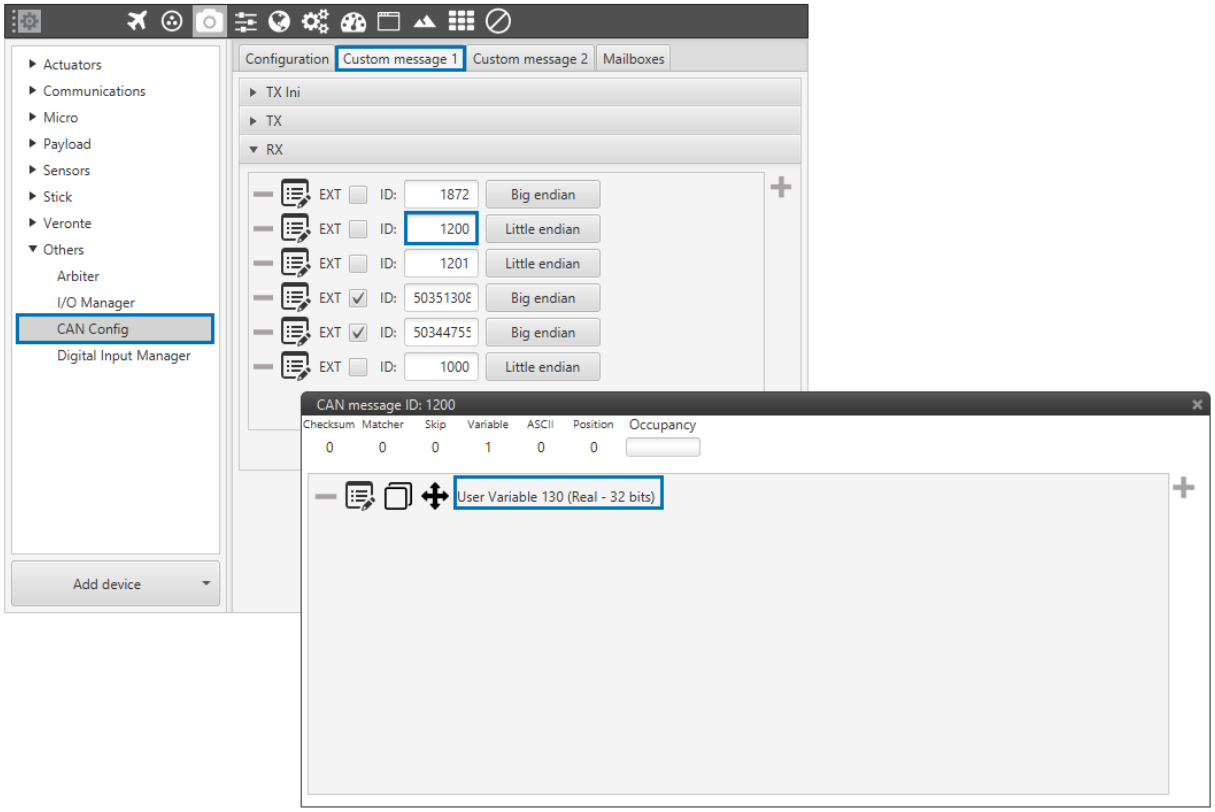
RPM: CAN Telemetry I/O Configuration

On the autopilot side, some mailboxes with ID 1200 will have to be created on whichever chosen reception CAN bus.



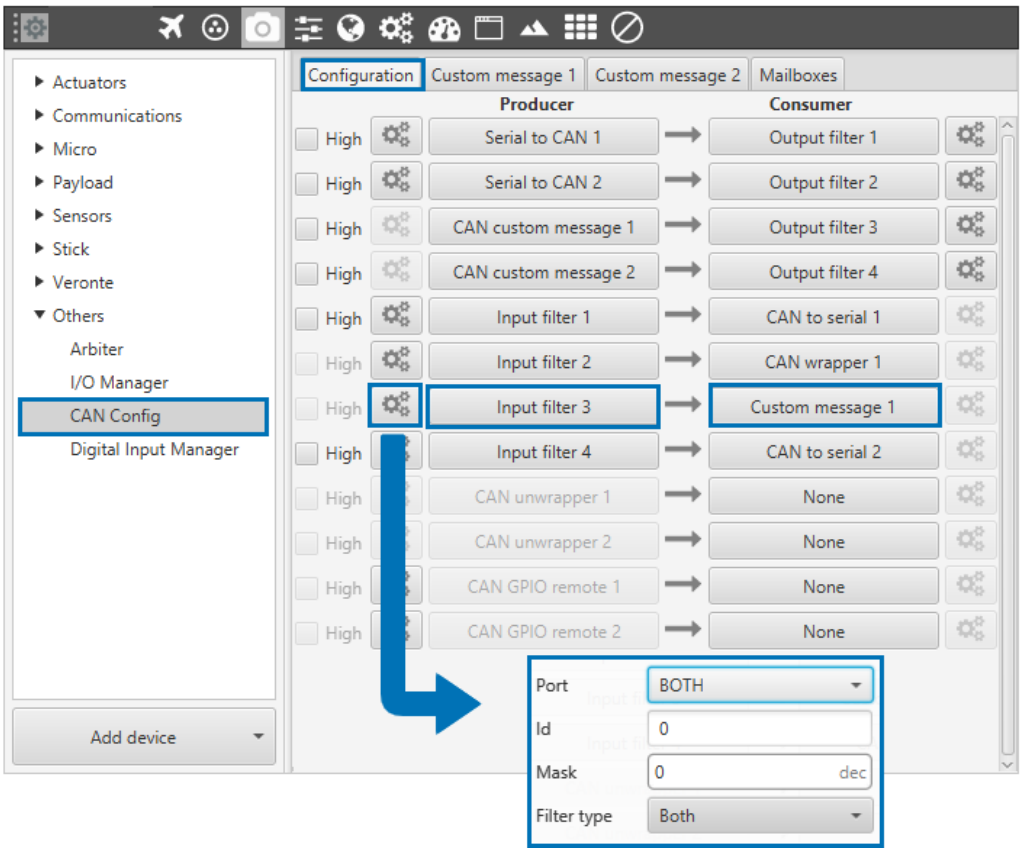
RPM: Mailboxes Assigned on the Autopilot

Accordingly, the 4 bytes information contained in **RPM1** will have to be stored in one of the available 300 real user variables (32 bits).



RPM: CAN Telemetry on the Autopilot

An input filter is used (producer) and the information is being received on the Custom Message 1 (producer). Both CAN buses of the autopilot can be used, as well as normal IDs and extended IDs.



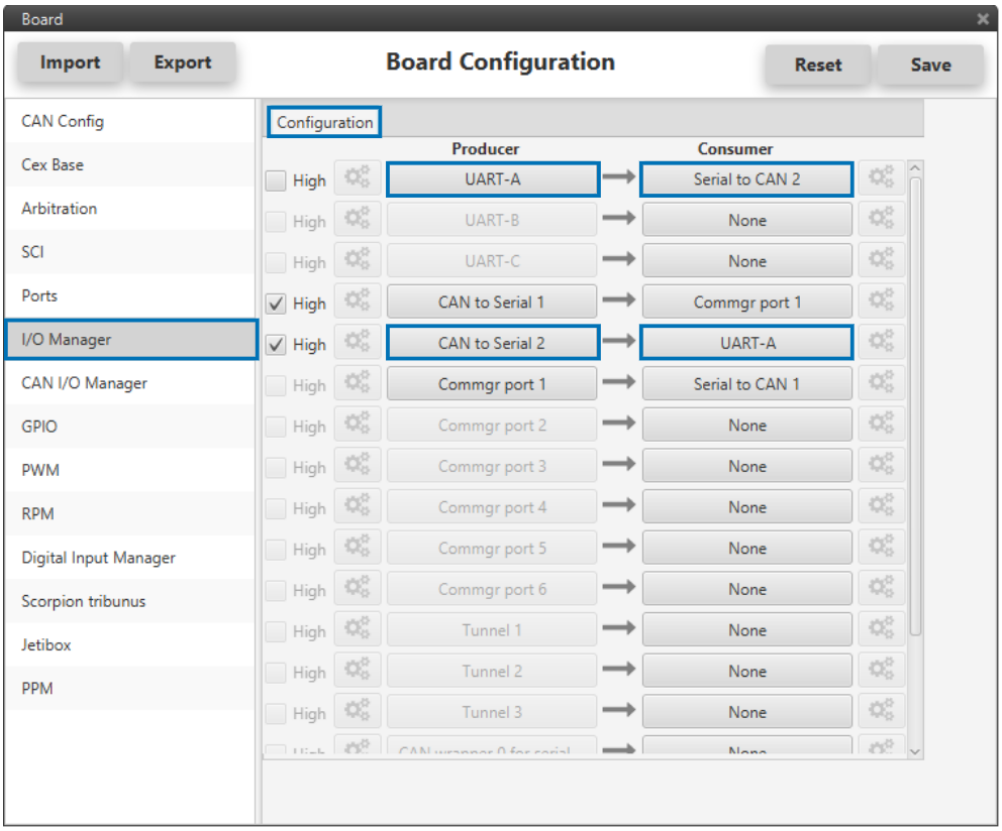
RPM: CAN Telemetry I/O Configuration on the Autopilot

6.6 UART

6.6.1 I/O Manager Configuration

Reception of serial information on UART-A (producer) is stored in Serial to CAN 2 (consumer). Transmission of serial information is sent using the CAN to Serial 2 (producer) over UART-A (consumer).

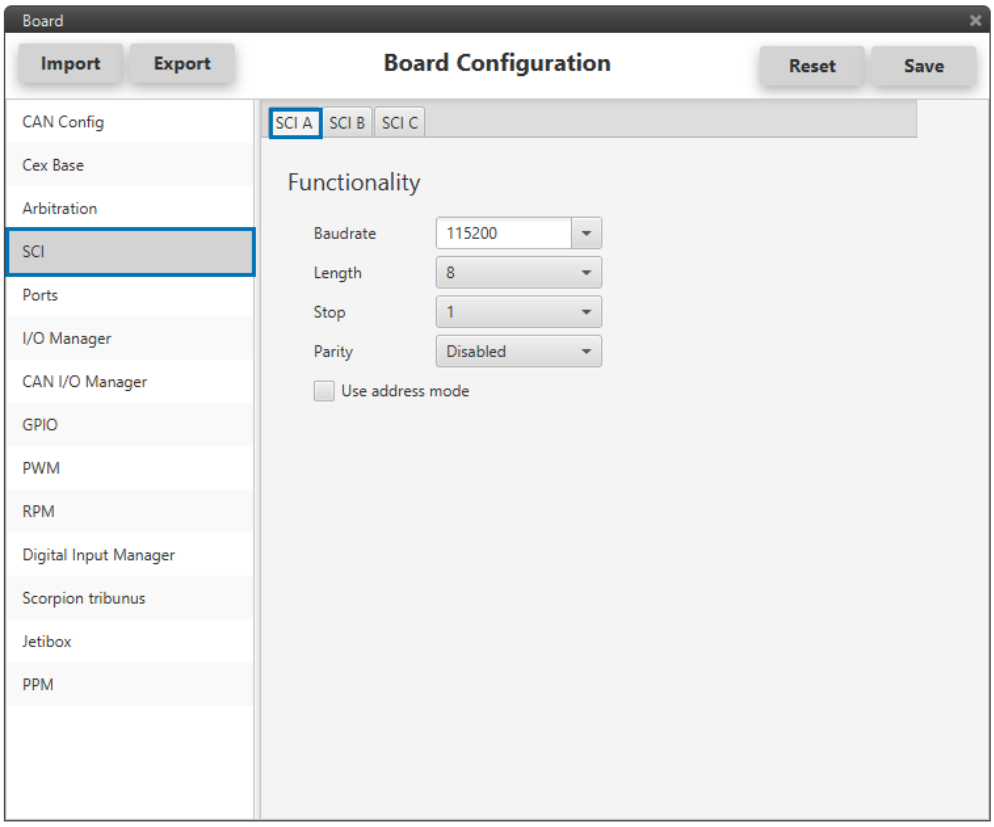
'Serial to CAN' and 'CAN to Serial' configuration is explained in the CAN I/O Manager menu section below.



IO Manager Configuration

6.6.2 SCI

Serial ports A, B and C parameters can be edited in this menu to fit the serial protocol requirements. Ports A and B will be different depending on the CAN Expander version (2xUART in OEM version; 1x RS232 and 1xRS485 in MC version).

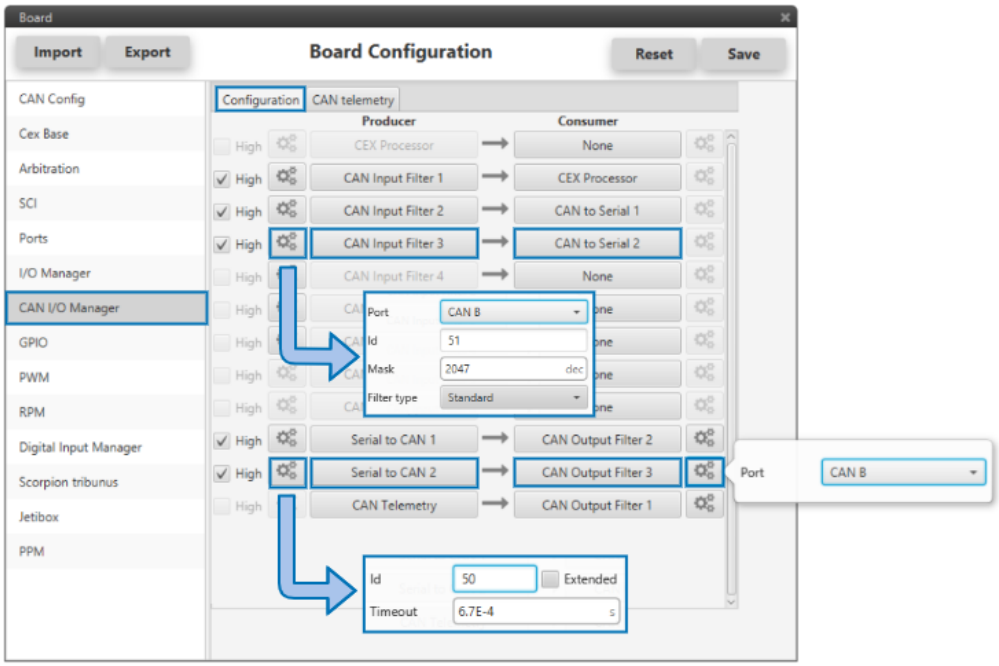


SCI-A Configuration

6.6.3 CAN I/O Manager Configuration

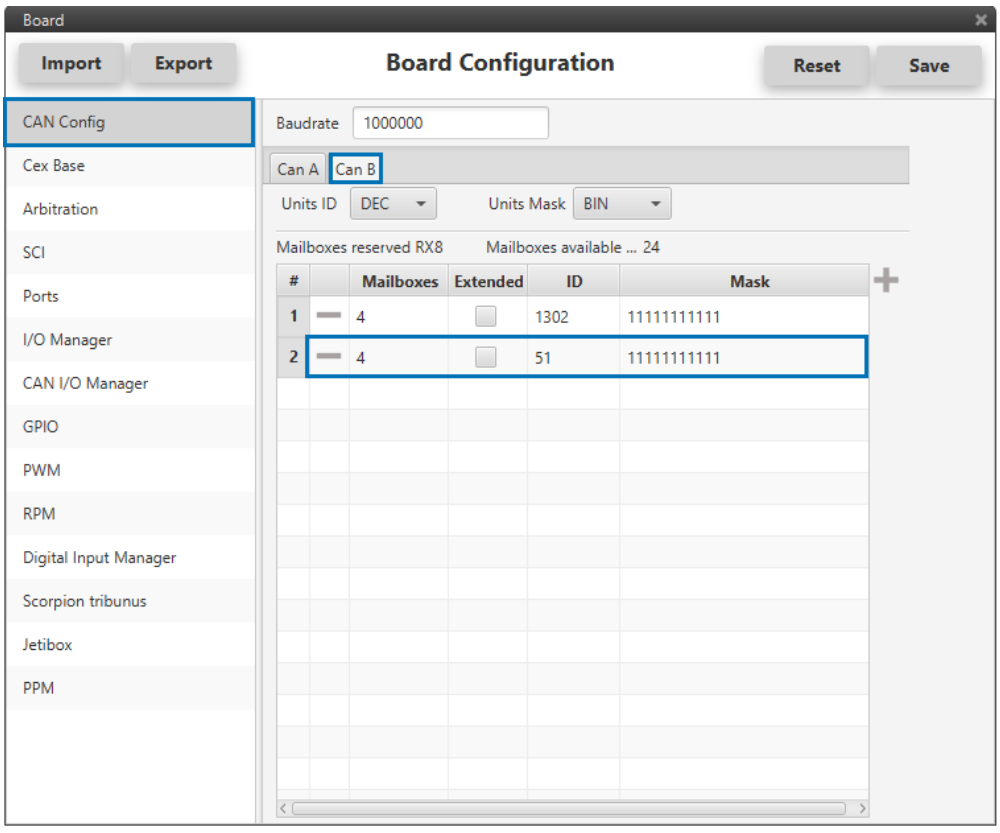
The information that will be sent over serial port UART-A is going to be received on the CAN Expander board over its CAN B port. A mask id of 51 is added to the Input filter. The incoming information (from Veronte autopilo) is processed in 'CAN to Serial 2'.

The information coming from port UART-A and processed in the board as 'Serial to CAN 2' is going to be linked to an Output filter. The information of 'Serial to CAN 2' is going to be sent over CAN B of the board with a mask ID of 50 (to be read by Veronte autopilot).



CAN I/O Manager Configuration

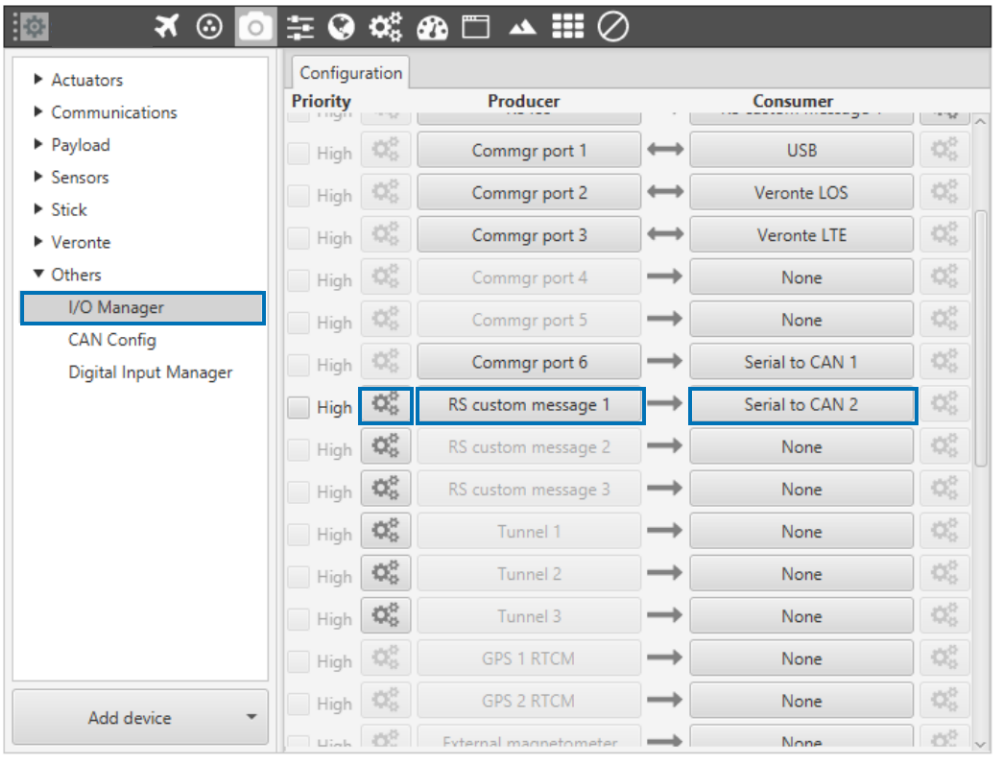
On the CAN Expander board, as in with Veronte autopilot, mailboxes need to be defined for the reception of CAN messages. In the example above, mailboxes for ID 51 need to be added on CAN B port of the board.



CAN B Mailboxes

6.6.4 Veronte Autopilot Side

On the **I/O Manager**, link an 'RS Custom Message' to a 'Serial to CAN' with the serial data that the autopilot is going to send to the CAN Expander board. Then, link another 'RS Custom Message' to a 'CAN to Serial' with the expected serial messages that the CAN Expander board will receive in the selected serial port.



Veronte I/O Manager Configuration

As for the **CAN I/O Manager**, the same IDs employed in the CAN Expander board for the Input and Output filters are going to be employed on Veronte's side, but they need to 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. Some mailboxes with ID 50 will have to be created on whichever chosen reception CAN bus.

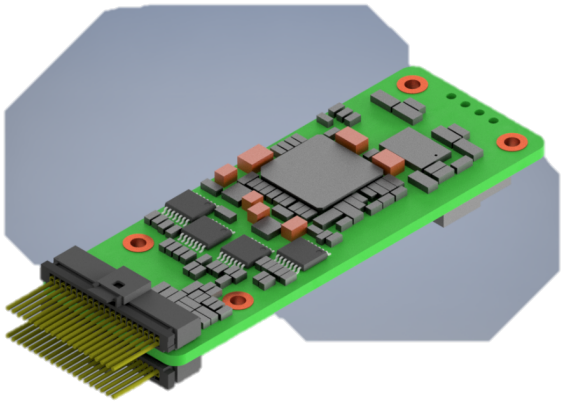
Some configuration examples are explained in this section.

CHAPTER

SEVEN

GENERAL DESCRIPTION

Veronte CAN Expander stands as a powerful peripheral to ease the reduction of wire in autonomous vehicles at the time it permits to increase the number of devices in the system. It makes possible to relocate and to group sensors, actuators, payloads, motor controllers… enhancing the I/O connectivity in the Veronte Autopilot. With its easy integration, Veronte CAN Expander becomes a quick solution for increasing connectivity capacity and allowing wiring optimization, especially in large systems.



Veronte CEX

Veronte CEM

7.1 Wiring Optimization

It is especially in large vehicles, where wire optimization plays a critical role permitting a significant weight reduction. This upgrade is achieved thanks to the reduction of cable length and because of the added flexibility so the right device can be installed in the right location. Another advantage of the use of Veronte CAN Expander is the robustness of the CAN Bus, being resistant to electromagnetic interferences and permitting the installation of long cables with no signal loss. Furthermore, it includes redundancy with CAN bus isolation, making it fail operational even in case of a CAN bus line break.

7.2 Enhanced I/O

With the use of Veronte CAN Expander, the data capacity for input and output in Veronte Autopilots is increased in a great manner. The advanced design makes possible to control several peripherals (PWM, UART, Digital Output, I2C, Analog Inputs…) through the CAN Bus. It can be used for both, expanding the I/O capacity in Veronte Autopilot, or for controlling peripherals with a robust communications protocol. In case it is needed, several CAN Expander boards can be installed in the same network for increasing the number of I/O ports or because of system architecture needs.

7.3 Applications

In aviation, a field where weight means such an important agent in design, struggling with wiring is one of the most common issues faced during the vehicle design. With the use of Veronte CAN Expander, not only this issue would be reduced, but a bunch of opportunities for different sensors and payload could arise:

- By adding more I/O interfaces, a more complex payload control can be achieved, improving connectivity.
- Advanced control of actuators and peripherals becomes feasible, being possible to condensate the connection of control, feedback, sensors… in a single board.

- Devices can be installed at long distances from the autopilot with no signal degradation thanks to the robustness of the CAN Bus.