MC110 PDI Builder

Release 7.4.1/1.0

Embention Sistemas Inteligentes, S.A.

2025-10-06

Contents

Scope of Changes

- Version 1.0
  - Added:
    - First version issued.

Quick Start

MC110 PDI Builder is the main tool for setting all configurable parameters of the **Veronte Motor Controller MC110**.

Here the user can configure, tune and define the motor, control systems and sensors to be used together with the ESC.

Once MC110 has been detected on Veronte Link, download and install **MC110 PDI Builder**.

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

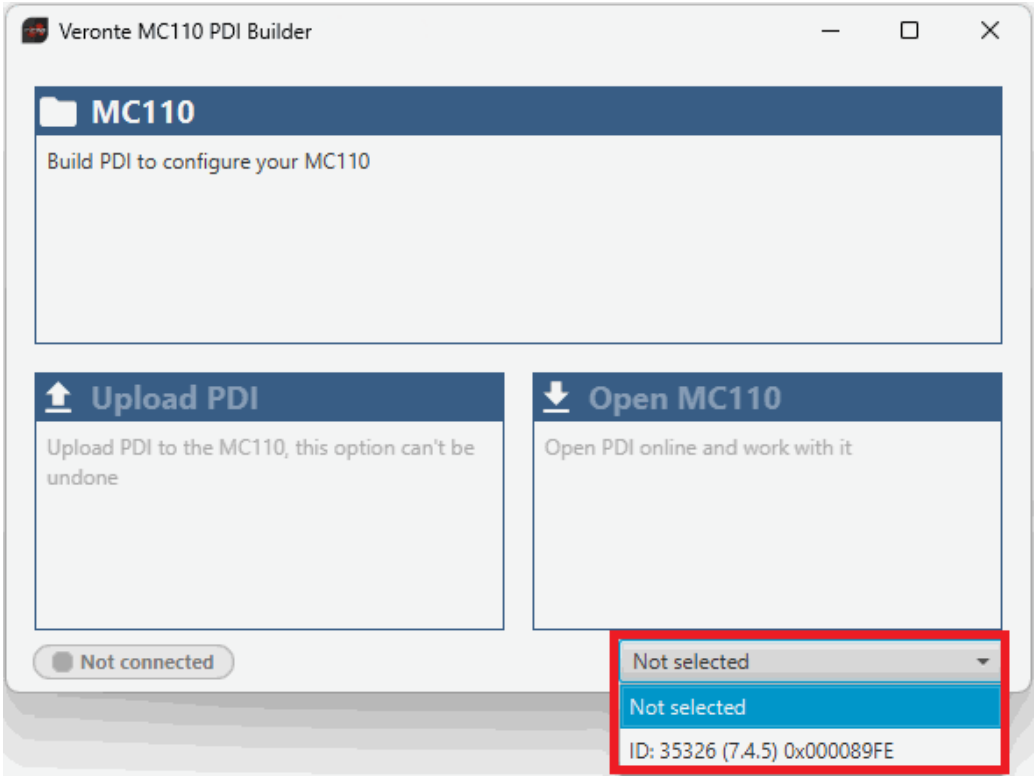
MC110 PDI Builder software is available in the **Veronte Toolbox** platform. From there, users can download and install the application. For more information, please refer to the Veronte Toolbox user manual.

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

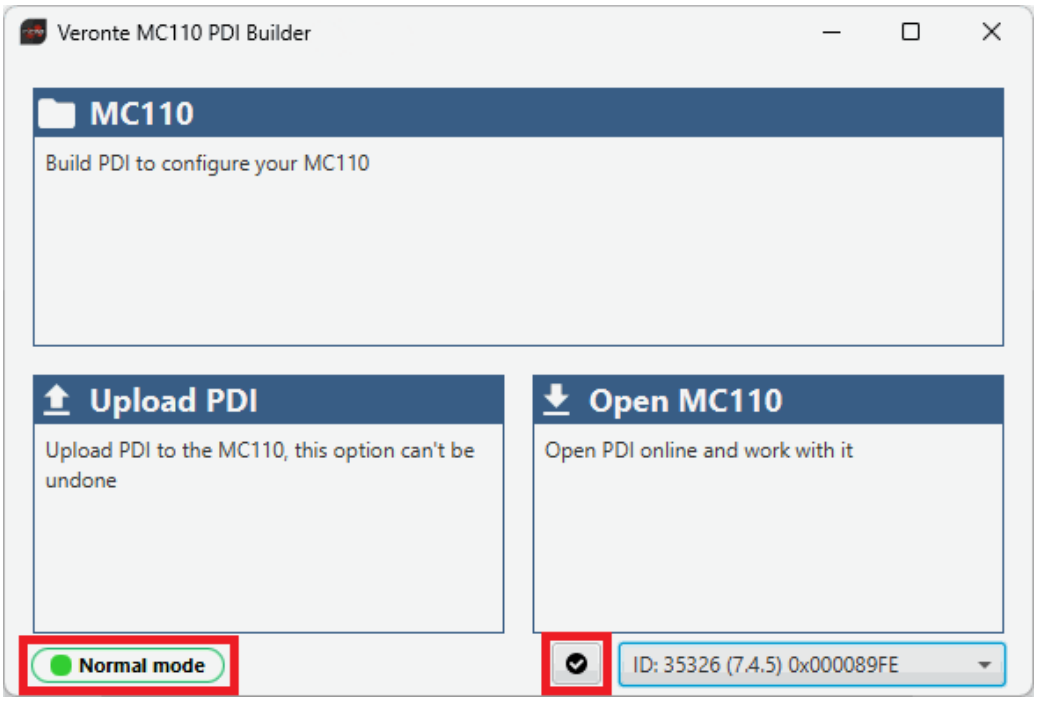
This section explains each option and parameter available in **MC110 PDI Builder**.

Once the installation is finished, open **MC110 PDI Builder** and select the unit.



MC110 ID

If the MC110 is correctly connected, **MC110 PDI Builder** will display the **mode** in which the connected unit is. In addition, a **PDI error button** will appear:

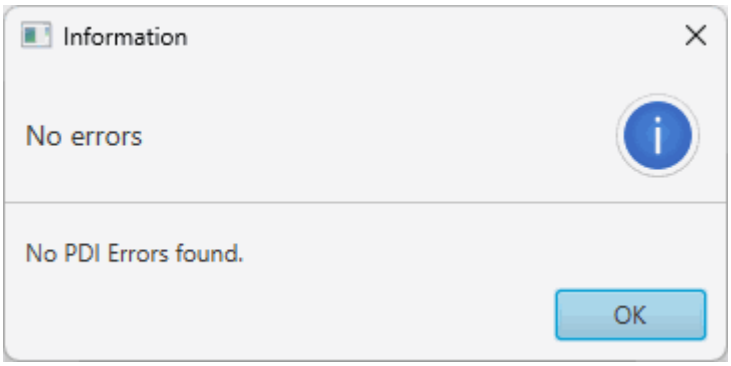


MC110 PDI Builder

- **MC110 mode**: The MC110 unit should appear in **Normal mode**, as shown in the figure above, or **Maintenance mode**.
  It can also appear as **Maintenance mode (loaded with errors)** or **Normal mode - Disconnected**.

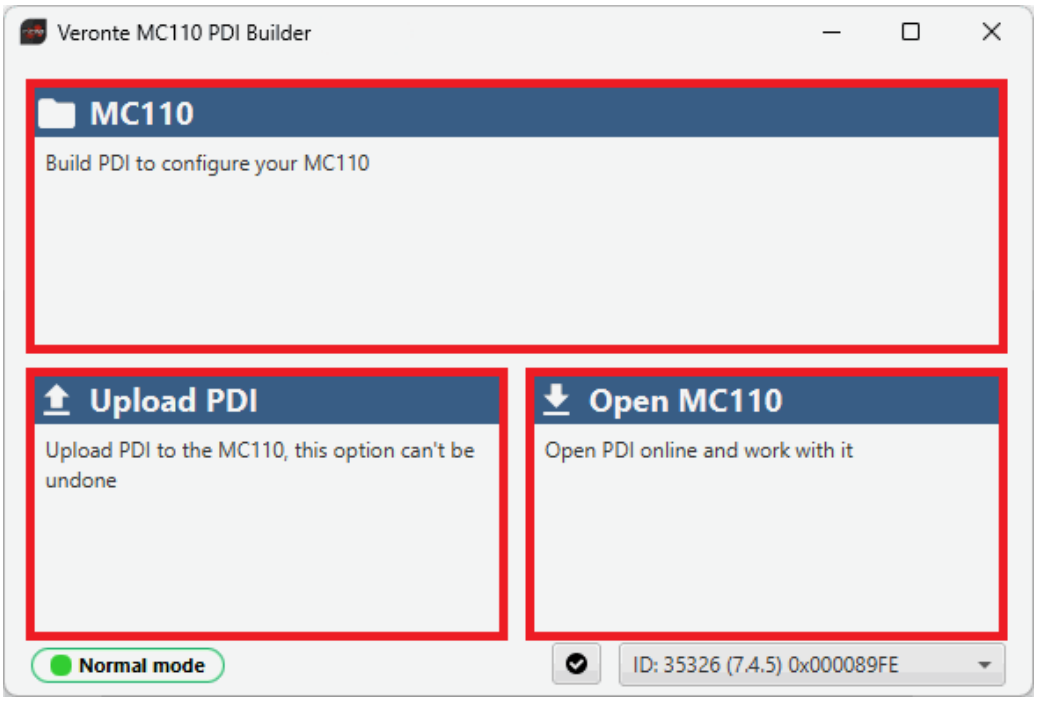
  > **Note**
  > **Maintenance mode (loaded with errors)** appears when something is wrong in the configuration.

- **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:



MC110 PDI Builder - PDI Errors message

The user can access now to 3 configuration options:



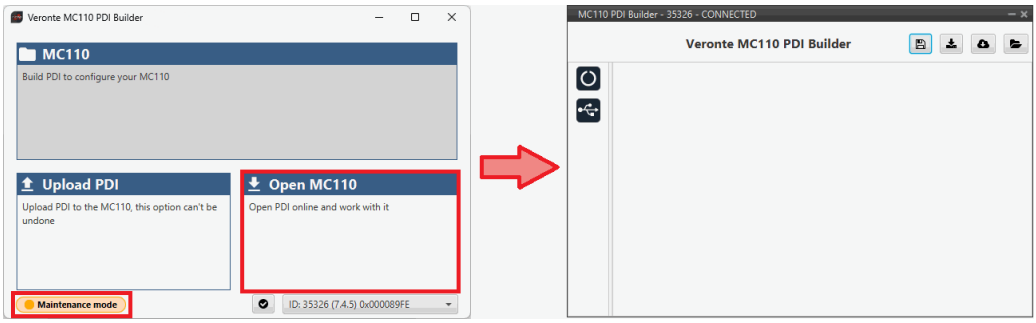
MC110 PDI Builder options

- **MC110**: It allows the user to work with **offline** configurations. A previously exported MC110 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 **MC110 PDI configuration** can be imported to the current **MC110** flash memory.
- **Open MC110**: By clicking on this option, **MC110 PDI Builder** configuration menu opens with the configuration (the PDI files) loaded in the MC110.

  Then, the user can modify it online.

> **Note**
>
> These PDI files are the files that contain the configuration of the MC110. They are used by modular control with improved version management.
>
> These files are located in a single folder, **setup**, which contains several .xml files, where all parameters and the control system are stored.

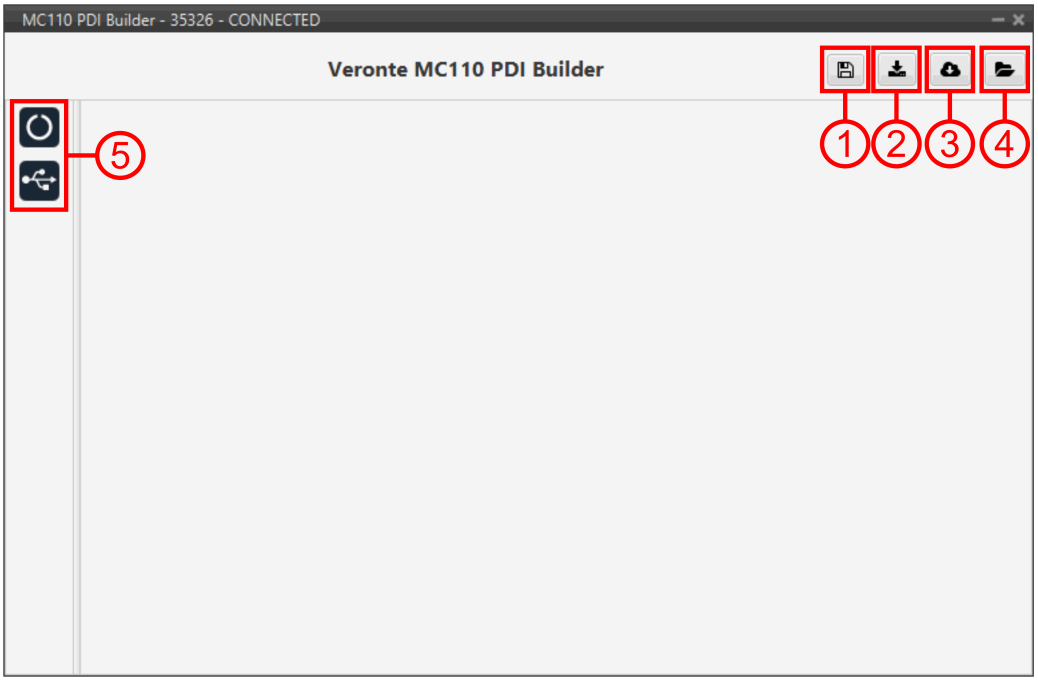
Finally, click on **'Open MC110'** to open the configuration and start editing online.



Open MC110

> **Note**
>
> When MC110 PDI is open, the unit changes to **Maintenance mode**.

The different 'buttons' that can be seen in the initial menu of the **MC110 PDI builder** are explained below.



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 MC110 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 MC110.
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 MC110.
5. These are the different functions of MC110. They are explained in the following menus:
   -  MC
   -  Input/Output

MC

Open Loop Startup

If it is not possible to start-up with the low speed PLL observer, it could be done by the open loop procedure:

Open Loop Startup panel

The user can configure the following parameters:

- **Enable**: Enables the open-loop start-up procedure. This could be used to perform some initial testing without a fine tuning.
- **Maximum Speed**: Maximum speed reached on the start ramp. It is usually approximately **10-15%** of the maximum speed of the mechanical motor.
- **Applied D-voltage**: This field defines the direct-axis voltage that is applied to the motor to generate the initial torque required to start spinning.

Motor

This tab sets all the electrical/mechanical parameters of the motor that will define the controller dynamics, which should be obtained from its official datasheet.

These values are the foundation for the controller's dynamics and are critical for achieving stable and efficient performance.

Motor panel

- **Stator internal resistance**: This is the line-to-neutral resistance of the motor. Here it is necessary to take into account the harness resistance if it is not negligible.
- **Quadrature/Direct Inductance**: Line-to-neutral motor inductance in q-d axes.
- **Pole Pairs**: The number of **pole pairs** shall allow calculation of the mechanical speed of the motor.
- **Maximum RPM**: Maximum desired mechanical speed that the motor will reach in normal operation. It allows to calculate the input rate in terms of desired speed.
- **Maximum Iq**: Maximum permissible quadrature current that the motor can drive. This is directly related with the maximum permissible motor torque.
  In a $I_d = 0$ scheme (no MTPA/FW) and steady-state operation, this is the allowed phase current peak value.
- **Maximum Id**: Maximum allowable direct current that the motor can conduct. Only used in the MTPA/FW scheme.
- **Back-EMF constant**: This is the line-to-neutral Back-EMF constant. Usually known as magnetic flux.
- **Total Moment Of Inertia** (MOI): Total moment of inertia (motor + load). This is only used for speed regulator gain auto-calculation tool.
- **Maximum temperature**: The maximum operating temperature that the motor can safely withstand. If this temperature is reached, a health alert will be raised.
- **Temperature coefficient**: The temperature coefficient of the stator resistance. If the appropriate thermistor is configured, this value allows the

controller to compensate for changes in motor resistance as it heats up, improving control accuracy.

> **Note**
>
> Back-EMF constant is an important parameter for low-speed PLL observer, so a correct characterization of this parameter will allow control in the low-speed range and start-up without open-loop procedure.

Observer

The algorithm is prepared to work with **two observers** (low and high speed observer) working in a predefined speed range.

Moreover, there are **two low-speed observers** that can be selected: **PLL observer** and **Arc-tangent observer** (this should be configured in the FOC Control panel). The first one allows starting without open loop while the second is usually selected when there is an open loop procedure.

The observer mixing limits [low,up] are usually approximately **2%** and **15%** of the maximum mechanical speed.

Observer panel

> **Warning**
>
> No changes are expected in this menu from the default values.

Here the observers parameters must be configured:

- **Sliding Mode Observer (Back-EMF estimator)**: Sensor-less estimator for the Back-EMF.
  - **Sliding Control Gain**: This is the main gain of the observer. To ensure the stability of the algorithm, this value must be greater than the absolute value of Back-EMF on the alpha/beta axes.
  - **Back-EMF feedback gain**: This value is to account for unfiltered Back-EMF. It should be between 0 and 1.

- **Arc-tangent Observer (Low speed)**: It is one of the low speed position observers.
  - **Time filter constant**: This is the time constant of the first order filter of the speed calculation. **The lower value, the lower filtering**.

- **PLL Observer (Low speed)**: It is the second one low speed position observer.
  - **Filter gain**: Filter cutoff frequency.

> ⓘ **Note**
>
> Users can choose one of the two low-speed observers (arc-tangent or PLL) by selecting it in the Low speed observer parameter of the FOC control menu.

- **PLL Observer (High speed)**: This is the high speed position observer. It is controlled by a PI control.
  - **Position gain**: This is the proportional gain for the position estimation loop. It determines how aggressively the observer corrects errors in the estimated rotor angle. A higher value results in a faster response, but may lead to instability if set too high.
  - **Speed gain**: This is the integral gain for the speed estimation loop. It works with the Position gain to eliminate steady-state errors and dampen oscillations, ensuring a stable lock on the motor's speed.

> ⚠ **Warning**
>
> It is recommended to tune this estimator even if an external sensor is used for feedback (sensored control).
>
> As it will be described in the next section, the control will automatically jump in case the primary feedback source fails to a sensorless control.

FOC Control

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 **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.

FOC Control is the main control menu, where all parameters related to FOC control are set.

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

FOC Control panel

Initial alignment

FOC Control - Initial alignment and Current Limiter parameters

> ⓘ **Note**
>
> These default parameters should work correctly.

Start-up initial alignment procedure in order to fix the motor starts from a known position. This is used by the Sin/Cos sensor to calibrate the sensor.

This procedure allows the rotor to always be aligned in the same electrical position.

> **Note**
>
> This part of the configuration is essential if the Sin/Cos sensor is enabled.

- **Applied D-voltage**: This sets the desired direct-axis voltage used to align the rotor.
- **Alignment time**: time used to set previous voltage. Usually about 1 second.

Current Limiter

> **Warning**
>
> Only changes regarding speed are expected in this menu from the default values. This is specially important in a sensor-less very low speed operation.

Limiter block for the desired currents I_q and I_d.

- **Maximum Id value for low speed**: maximum allowed D-current when the speed is lower than **Lower speed limit to start Id control**.
- **Maximum driving Iq for low speed**: maximum driving Q-current when speed is lower than **Maximum speed to consider low-speed**.
- **Maximum regenerative Iq**: maximum allowable regenerative Q-current when a break/reverse operation is required.
- **Maximum regenerative Iq for low speed**: maximum breaking Q-current when speed is lower than **Maximum speed to consider low-speed**.
- **Lower speed limit to start Id control**: lower speed limit to consider adding Id control with a value of **Maximum Id value for low speed**. This about a 2% maximum motor speed.
- **Upper speed limit to remove Id control**: upper speed limit to consider adding Id control with a value of **Maximum Id value for low speed**. This about a 6% maximum motor speed.
- **Maximum speed to consider low-speed**: low speed limit to consider Iq limiting using limits defined at **Maximum regenerative Iq for low speed** and **Maximum driving Iq for low speed**.

Observer Mixing Controller

FOC Control panel - Observer Mixing Controller and MTPA Controller

Definition of the range of low-high speed observers.

- **Low speed observer**: Users should select the low-speed observer to be used: arc-tangent or PLL.
- **Lower limit to mix observers**.
- **Upper limit to mix observers**.

MTPA Controller

Maximum Torque Per Ampere controller. It is used to reach a higher speed than the base speed of the motor. In addition, it can work in a Field Weakening to control with I_d .

- **Minimum speed to consider MTPA or FW**.
- **Minimum speed to active MTPA**.

FOC Controllers

FOC Control panel - FOC Controllers

This section is used to define the three main PI control loops of the FOC algorithm: the **Quadrature current** (torque regulator), **Direct current** (flux regulator), and motor **Speed Controller**.

- **Auto-calculation**: Instead of calculating gains manually, the user can input the desired performance characteristics, and the system will compute the appropriate gains considering the motor characteristics and the desired bandwidth.
  - **Current controller bandwidth**: Defines the desired responsiveness of the current control loops (Direct and Quadrature).
  - **Speed controller bandwidth**: Defines the desired responsiveness of the speed control loop.
  - **Auto-calculate gains button**: After setting the desired bandwidths, the user clicks this button to automatically fill in the gains fields below.

> **Note**
> 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.

Alternatively, the user can enter the gains manually in the sections below. These fields will be fill in automatically when using the auto-calculation feature. The gains are divided into three sections:

- **Quadrature Controller**: This loop controls the quadrature current (I_q), which is directly related to the motor torque. It is controlled by a PI control.
- **Direct Controller**: This loop controls the direct current (I_d), which is related to the magnetic **flux** of the motor. It is controlled by a PI control.
- **Speed**: This loop controls the rotational speed of the motor. It is controlled by a PI control. In addition to the Proportional gain and Inverse integral time constant, the user can also set:
  - **Tracking gain**: An additional gain to improve the controller's ability to track the desired speed reference and avoid controller wind-up.
  - **Maximum speed rate**: Defines the maximum desired acceleration that the motor will perform. This is platform dynamics dependent.

Control Input

FOC control panel - Control Input

Control input allows the user to select the input source and several additional parameters such as:

- **Mode**: Control mode. The available options are **m_PPM**, **m_CAN** and **m_CAN_PPM**.
- **CAN Time Out**: Whenever this timeout is met, the control input will switch to the next option if available. If none of them are available, it shall end up in failsafe mode. A health alert will any of the sources are not available.
- **Value for Fail Safe**: Value between **0 and 1**. To be written in case the previous mode options are not available.
- **Fail Safe**: Activates failsafe mode.

The input flow is the following in case the mode **m_CAN_PPM** is selected:

CAN_PPM mode diagram

Additional options

FOC Control panel - Additional options

In this section, a variety of advanced features and behaviors of the motor controller can be configured.

The available parameters are:

- **Switching frequency**: Defines the switching frequency of the controller's transistors. Higher values can result in smoother motor operation but may increase thermal losses.
- **Regenerative Braking**: Activates regnerative braking feature which is basically consisting of thoughing negative current back to battery whenever the motor is loosing speed (it uses the kinetic energy to charge de battery). If disabled, it will only allow slow down due to losses.

  > ⚠ **Warning**
  > Activating this option is only recommended with batteries (charging them when braking), but not with a power supply, as they are not prepared to receive current (it is necessary to activate the sink mode to work with it).

- **Input deadband**: This is the value of the input rate at which the motor will start to move. For example, it may be desired to be non-zero in case an RC stick outputs about 0.2 duty cycle by default.

  > ⓘ **Note**
  > In case CAN Bus is used to command (see CAN I/O - Input/Output section of this manual), this deadband can be calculated as $\frac{RPM^*}{RPM_{max}}$.

- **Hall speed sensor filter cut-off frequency**: Configures the cut-off frequency for the speed signal filter when using Hall sensors, which helps to smooth the speed reading.

- **Fan Cooling**: This option activates a PID control on a cooling fan to reach normal operating temperature.
- **Invert spin direction**: Reverses the motor's direction of rotation without needing to change the phase wiring.
- **Enable regenerating limiting**: Activates a system to limit the regenerative current, which is useful for protecting the battery from overvoltage during intense braking. The following parameters are available when enabled:
  - **Voltage to start limiting regenerative current**: The battery voltage at which the system begins to limit the regenerative current.
  - **Voltage to stop regenerative current**: The battery voltage at which the system stops limiting regeneration.
- **Limit Output Power**: Activates a system to limit the maximum output power of the motor, based on the battery voltage. When enabled, a multi-step power limit curve can be configured:
  - **First/Second/Third voltage/power step to limit**: These fields allow the user to define a stepped power limit. As the battery voltage is between certain ranges (first field), the maximum output power is reduced to the specified value in the second field.

Finally, the following state machine will be executed during a normal operation (when the open loop procedure is disabled):

State machine diagram

Fault Detection

This tab allows the user to configure the thresholds for various electrical and system faults, as well as identify whether faults are critical or not.

Values definition

This section is used to set the specific numerical limits and timings that will trigger a fault condition. The parameters whose limits can be modified by the user are:

Fault Detection panel

Types of fault detection:

- **Over-voltage**: Over-voltage detected in battery.
- **Under-voltage**: Under-voltage detected in battery.
- **RMS imbalance**: Imbalance detected between phase currents.
- **Open-DC**: Potential open circuit detected in battery.
- **Low-energy**: Low voltage battery detected during a long time.

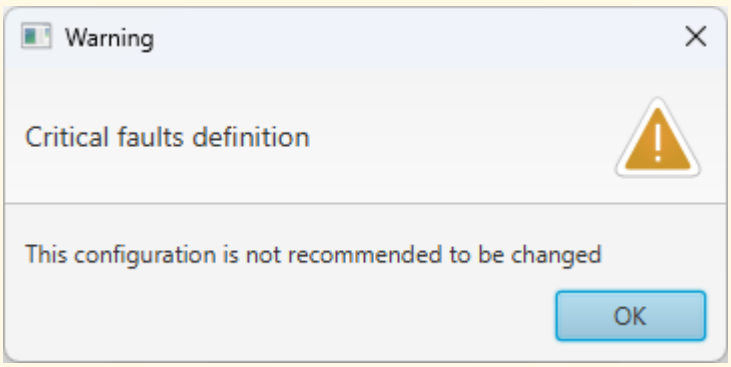
Critical faults definition

Users can mark which faults should be considered critical. The differences between critical and non-critical:

- **Critical fault**: It will be followed by a latched OFF state.
- **Non Critical fault**: It will be only followed by a warning.

Fault Detection panel

> ⚠ **Warning**
>
> Although it is possible to change critical faults to non-critical and vice versa, it is not recommended to modify the default settings. For this reason, the following warning message appears when entering this panel:



Fault Detection panel - Warning

Sin Cos

In case it is necessary to use an external SIN/COS sensor as a source of electrical angle/speed feedback, the main interface for doing so is via ADC inputs.

This menu allows users to calibrate their sensor. Proper calibration corrects analog signal errors, ensuring accurate angle feedback for the motor controller.

Sin Cos panel

The parameters that can be configured by the user are:

- **Enable sensor**: Checkbox to activate or deactivate the external Sin/Cos sensor for feedback.
- **Offset (Cos) / (Sin)**: Corrects DC voltage offsets to center the Cosine and Sine signals at 0V.
- **Amplitude gain (Cos) / (Sin)**: A multiplier that scales the signal to ensure both Sine and Cosine have identical amplitudes, which is critical for precise angle calculation.
- **Orthogonality gain (Cos) / (Sin)**: Corrects phase errors. This gain ensures the Sine and Cosine signals are perfectly orthogonal (90 degrees apart).

> ⓘ **Note**
>
> If the calibration values are not known, this is auto-calculated by the system enabling open-loop procedure and initial alignment.

Thermistor Configuration

This section is for configuring a thermistor to provide accurate temperature readings.

Status panel

Set the following parameters:

- **Variable resistor side**: Sets the thermistor's position in the voltage divider circuit, either **High** or **Low** side.
- **Beta**: The thermistor's Beta coefficient (B-value) from its datasheet.
- **Reference Temperature**: The temperature for the specified nominal resistance (usually 25°C).

- **Reference Resistance**: The thermistor's nominal resistance at the reference temperature.
- **Serial Resistance**: The value of the fixed resistor in the voltage divider.
- **Additional Resistance**: Any extra circuit resistance, such as from wires, to include in calculations.

The variable resistor is calculed using the following equation:

$$R_T = R_0 \cdot e^{\beta(\frac{1}{T}-\frac{1}{T_0})}$$

- For resistor in **high** side:

$$R_T = \frac{R_s}{V} - R_s - R_1$$

- For resistor in **low** side:

$$R_T = \frac{R_s \cdot V}{1 - V}$$

Where,

- R_0 : Reference Resistance parameter
- β : Beta parameter
- T_0 : Reference Temperature parameter
- R_s : Serial Resistance parameter
- R_1 : Additional Resistance parameter

Status

- **Enable VCP Status Message** enables the periodic sending of the status message that Veronte Link uses to recognise the Veronte Motor Controller MC110.
- **Period**: Enter a desired period to send repeatedly the status message.

Status panel

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

Input/Output

I/O Setup

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

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 a bit in a way that allows this connection to be activated/deactivated depending on the status of the selected bit. By default, the 'Always Ok' bit is set to all connections so that they are always active.
- **Add a New Row**: Press the `+` button at the bottom of the connections list to add a new connection. A new empty row will appear with "None" set for both Producer and Consumer.

New I/O Setup connection

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

The following I/O ports are available:

Field	Description
RS232	Serial Port 232 (SCI-B)
RS485	Serial Port 485 (SCI-A)
USB (SCI-C)	USB Port
Commgr port	COM Manager ports send and receive VCP messages. This is the protocol used by Veronte products to communicate. For more information on VCP, read the VCP user manual
Custom Message producer	This allows user to send a serial custom message, see Serial custom messages section
CAN to serial / Serial to CAN	**Serial to CAN** sends serial streams over a CAN Bus / **CAN to serial** undoes the transformation 'Serial to CAN'

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

Serial Custom Messages

> ⚠ **Warning**
>
> - MC110 has a **serial limitation** of **64 vectors** (fieldset).
>
> In addition, there is a limit shared with **CAN Custom Messages**:
> - Maximum number of **vectors** (fieldset): **104**
> - Maximum number of **fields**: **2000**

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

Serial Custom Message

To configure a serial custom message, the user must follow the next steps:

1. Press the **configuration button** (icon) and a pop-up window will be displayed.
   In this window press the ➕ icon to add a custom message, the user can choose between **System variables**, **ADSB Vehicle** or **External Sensor**.
   Serial Custom Message configuration

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

2. When it is already added, the following options are available to configure a custom message:
   Producer Serial Custom Message configuration

   - **Endianness**: Depending on the order in which the device outputs the message, it is possible to select:
     - **Big endian**: Set the value from left to right.
     - **Little endian**: Set the value from right to left.
     - **Mixed endian**: Some devices use this format. If users need to configure it, please contact the support team (create a ticket in the customer's **Joint Collaboration Framework**; for more information, see Tickets section of the **JCF** manual).
   - **Period**: It is the inverse of the send frequency.
   - **Delay**: It is a delay applied before sending the message. This serves to send messages with the same period without overloading the serial bus.

3. To create the structure of the message, click on the **edit message button** (icon) and then press the ➕ icon to add fields to it.

The following type of messages are available to configure a structure: **Variable**, **Checksum**, **Matcher**, **Skip** and **Parse ASCII**.

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

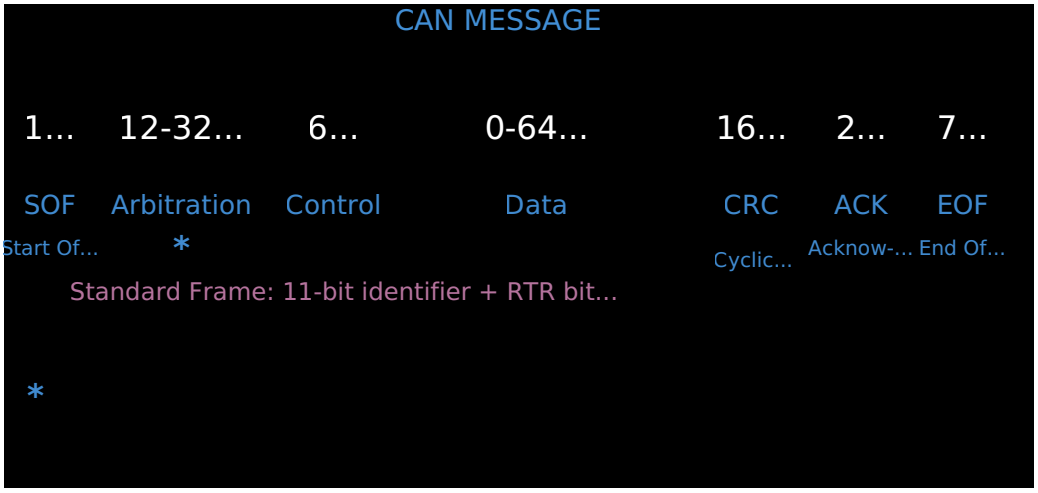
> ⚠ **Warning**
>
> Before configuring any message, user has to know the structure it has to have according to the device that is connected to the port. Each device may have a different message structure when it sends or receives information.

CAN I/O

A CAN (Controller Area Network) Bus is a robust standard communication protocol used for vehicles. MC110 is equipped with two distinct CAN buses that can be configured independently:

- **CAN B (Classic CAN)**: A standard classic CAN bus. The configuration for this port, such as its Baudrate and reception Mailboxes, is managed within this CAN I/O panel.
- **CAN A (CAN FD)**: A CAN bus with Flexible Data-Rate (FD) capability, allowing for higher speeds and larger data frames. The configuration for this port is managed in the CAN-FD A Setup panel.

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.

Configuration

This menu allows the configuration of the CAN inputs and outputs.

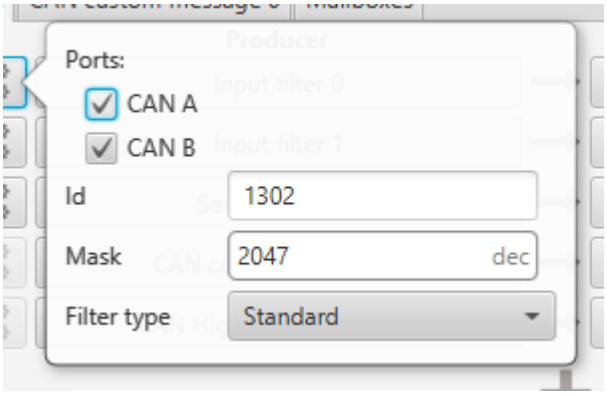
CAN configuration panel

In this menu, the user can find the same 'columns' (**Priority**, **Producer**, **Consumer** and **Bit**), as well as the "Add a New Row" option, as in the I/O Setup panel.

> ⚠ **Warning**
>
> In CAN, in Low state the specified period is not guaranteed but in High state it is.
>
> However, only those **messages that are critical for external devices** should be set as **high priority**, as this may disrupt the proper functioning of Veronte MC110.

MC110 has the following list of **producers**:

- **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 ⚙:



Input Filter configuration

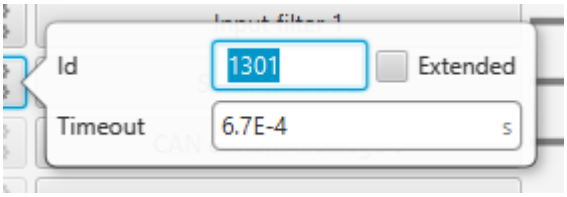
- **Ports**: It is required to configure the CAN bus from which it listens, the user can select between CAN A, CAN B or both at a time.
- **Id**: CAN Id must be set and it is used to identify messages. The value set has to be decimal format.
- **Mask**: A CAN Id mask can be set to filter messages. **The 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.

- **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:

  > ⚠ **Warning**
  > For correct communication, mark it as **High priority** (with the Priority checkbox).



Serial to CAN configuration

- **Id**: CAN Id must be set and it is used to identify messages. The value set has to be decimal format.

    - **Extended**: If it is enabled, the frame format will be 'Extended', i.e. with a **29-bit identifier**. Otherwise, the frame format 'Standard' (11-bit identifier) is set by default.
    - **Time out**: This is the threshold time between receptions to consider that it is not being received correctly.
- **CAN custom message 0**: CAN custom messages transmission. They are configured in the next section: CAN custom message.
- **CAN High speed telemetry**: Telemetry messages sent via CAN faster than normal. They are configured in the next section: CAN High Speed Telemetry.
- **IPC Diagnostics data sender**: Transmits a predefined CAN message containing internal diagnostic data. It is used for advanced monitoring and debugging purposes.
- **Motor Performance Status**: Transmits a predefined CAN message that summarizes the motor's performance status. It is designed to provide a high-level overview of the motor's state.

The **consumers** are explained in the following list:

- **Output Filter**: CAN output filters. The user can select between **CAN A**, **CAN B** or both at a time in .

Output Filter configuration

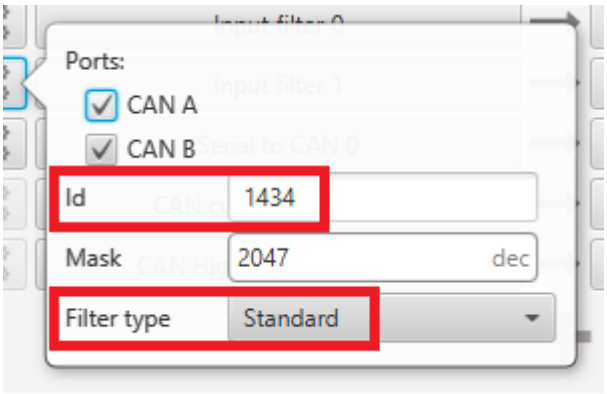
- **CAN to Serial**: This undoes the 'Serial to CAN' action, it has to be connected to I/O Setup **producer**.

> ⚠ **Warning**
>
> For correct communication, mark it as **High priority** (with the Priority checkbox).

- **CAN Cmd**: This is the input ESC expects to command RPM to the motor. It has to be connected to an **Input Filter**.

> ⚠ **Warning**
>
> This configuration is already done by default. If the user changes this ID, make sure that the ID of the mailboxes also matches.

CAN configuration - CAN Cmd



CAN Cmd - Input filter configuration

CAN custom message

In the CAN custom message tab, the user chooses the variables to be sent 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 MC110 boots up). They can be used to initialize some devices.
- **TX**: Used 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.
> - MC110 has a **CAN limitation** of **40 TX** messages and **40 TX Ini** messages.
>
> In addition, there is a limit shared with **Serial Custom Messages**:
> - Maximum number of **vectors** (fieldset): **104**
> - Maximum number of **fields**: **2000**

CAN custom message panel

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

Mailboxes

> ⓘ **Note**
>
> This section is used to configure the classic CAN bus (Port B). For the configuration of Port A, which supports CAN FD, please refer to the CAN-FD A Setup section below.

Main screen to configure baudrate and reception mailboxes of the CAN B.

Mailboxes panel

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

> ⓘ **Note**
>
> Furthermore, in the previous image it can be seen that there is a mailbox already configured with ID **1434**. Remember that this is the mailbox associated with the **CAN Cmd message**, explained above, and that is **configured by default**.

Mailboxes - CAN Cmd

CAN-FD A Setup

This section is used to configure the CAN FD (Flexible Data-Rate) bus for Port A. CAN FD is an evolution of classic CAN that allows for higher speeds and larger data frames.

The configuration is split into two main parts:

Mailboxes

Mailboxes Panel

This screen is used to configure the reception mailboxes for the CAN FD bus. The interface and function are similar to the Classic CAN mailboxes, allowing for the setup of hardware filters for incoming messages.

For more detailed information on configuring mailboxes, please see the Mailboxes - Input/Output section of the **1x PDI Builder** user manual.

Note that unlike the classic CAN configuration, the Baudrate settings for CAN FD are located in a separate tab.

Baudrate

Baudrate for the CAN FD bus

This tab is essential for configuring the main feature of CAN FD: its ability to operate at two different speeds.

Arbitration Rate (Base Speed)

This part sets the standard communication speed for the bus.

- **Arbitration Baud rate**: This dropdown sets the base speed (e.g., 1MHz) used for bus negotiation. This rate must be the same for all devices on the network to ensure reliable communication.
- **Arbitration Timings Configuration**: These optional parameters can be configured by checking the Customize box. These are advanced settings for fine-tuning the bus timing. They typically do not need to be modified.

Data Rate (High Speed)

This part enables the high-speed data transfer feature of CAN FD.

- **Enable CAN-FD Mode**: This checkbox must be activated to use CAN FD features, such as larger 64-byte data frames.
- **Enable Bit Rate Switch**: Activates the "dual speed" function, allowing the bus to accelerate during the data phase of the message.

By activating the checkbox, the following parameters will be available:

- **Data Baud Rate**: This dropdown sets the higher speed that will be used for the fast data transmission phase.
- **Data Timings Configuration**: These optional parameters can be configured by checking the Customize box. They are advanced settings for the high-speed data phase.

Serial

MC110 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.

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

SCI panel

- **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.

CAN High Speed Telemetry

MC110 is equipped with special CAN telemetry, which is faster than the regular one. This is intended to **monitor all RPM-dependent variables that can be crucial for tuning the user's ESC during the initial stages**.

For instance, seeing the electrical angle can be extremely difficult with a low sampling rate at high RPMs and as a consequence, tuning the observer gain can be difficult. Similarly, monitoring the current being measured in each phase can present the same problem and make PI tuning very time consuming.

CAN High Speed Telemetry panel

Only two parameters need to be configured here:

- **Base ID**: This is the CAN Id associated with the first variable in the list to be used. Subsequent IDs will be reserved according to the variable list.
  - **Extended**: If 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.
- **Ddecimation** (frequency): The number of clock steps the ESC skips before sending a High Speed CAN telemetry packet. The clock step is running at 2kHz.

> ⓘ **Note**
>
> A separate tool is offered to see MC telemetry ⇒ **CAN Telemetry**, please contact sales@embention.com.

Tuning

To achieve precise and efficient motor control, the MC110 uses a strategy known as Field-Oriented Control (FOC). The following block diagram illustrates this workflow.

Tuning - FOC workflow diagram

As seen in the block diagram, there is a switch (that is optional) that defines two positions and that will determine the open or closed loop of the speed control. This is done because at startup, the estimator block (sometimes) is not capable of correctly calculating the position/speed of the motor at low speed, so there is a special startup configuration, in which a known speed ramp is commanded and the angle is calculated.

Looking at the block diagram, the blocks to be tuned are:

1. Motor characterization → Motor panel.
2. Open loop start-up (if it is active) → Open Loop Startup panel.
3. Angle estimator (sensor, or failing that, observer). → Observer panel.
4. Quadrature current *PI* → Quadrature Current Controller.
5. Direct current *PI* → Direct Controller.
6. Speed *PI* → Speed Controller.

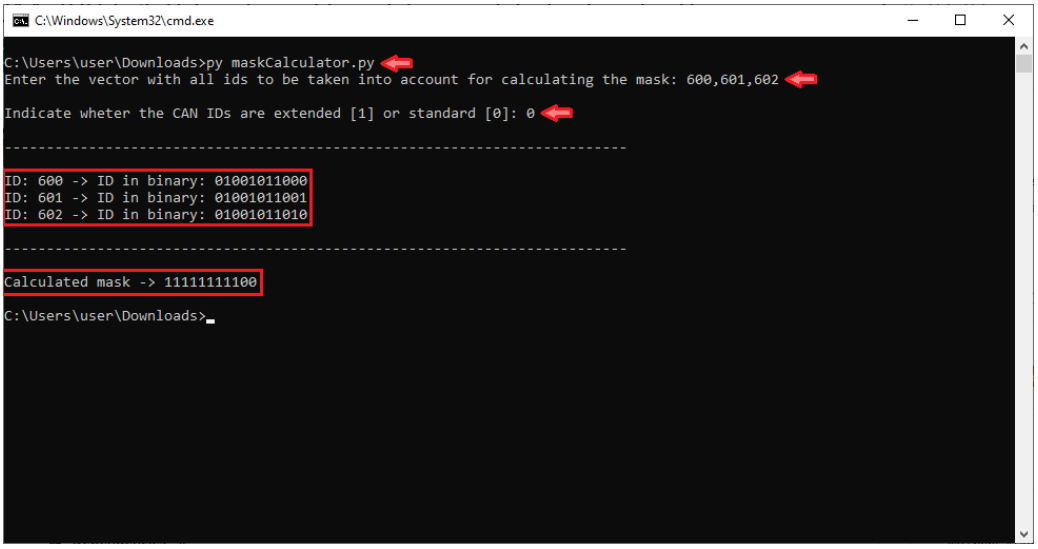
FAQ

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:



Example of maskCalculator program

Software Changelog

This section presents the changes between the previous software version (**v. 6.12.35**) and the current one (**v.7.4**).

Added

- A new CAN-FD A Setup panel has been added to the Input/Output menu, allowing for the configuration of the high-speed CAN FD bus, including its separate Mailboxes and Baudrate settings.
- A Thermistor Configuration panel has been introduced in the MC menu to allow for the setup and calibration of a thermistor for temperature readings.
- An auto-calculation feature has been added to the FOC Controllers panel, which computes the PI gains automatically based on desired controller bandwidths.
- New parameters for Maximum temperature and Temperature coefficient have been added to the Motor configuration panel.
- The Fault Detection panel now includes a much more extensive list of configurable fault values and timings.
- The Additional options tab belonging to the FOC Control panel has been significantly expanded with the following new parameters: Switching frequency, Invert spin direction, Enable regenerating limiting, and Limit Output Power.

Improved

- The Sin Cos panel has been completely redesigned for sensor calibration. It now uses parameters like Offset, Amplitude gain, and Orthogonality gain instead of the previous ADC channel and voltage settings.
- The process for adding connections in I/O Setup and CAN I/O is now dynamic. The user must press a + button to add new rows, replacing the previous static list of connections.
- The main CAN I/O panel has been updated to clarify the new architecture: it now manages the low-level configuration for the Classic CAN (Port B) and the logical routing for both buses.

- The Fault Detection panel is now better organized, splitting the configuration into Values Definition and Critical Faults Definition.

Changed

- In the Open Loop Startup configuration, the Start-up Iq parameter has been replaced with Applied D-voltage.
- In the Initial alignment panel, the Initial alignment flux current parameter has been replaced with Applied D-voltage.
- The tabs for configuring the Quadrature, Direct, and Speed controllers are now grouped under the main FOC Controllers panel.

Removed

- The Idle Speed and Idle Speed Hold Time parameters have been removed from the Additional options tab.