AVR® MCU Motor Control Library

AVR MCU Motor Control Library

www.microchip.com Product Pages: AVR16EB14, AVR16EB20, AVR16EB28, AVR16EB32, AVR32EB14, AVR32EB20, AVR32EB28, AVR32EB32

Library Scope

The AVR® MCU Motor Control Library is a library for the AVR EB family of devices with support for spinning a Brushless Direct Current (BLDC) motor and Permanent Magnet Synchronous Motors (PMSMs) using either a sensored or sensorless setup. The library interface offers motor-specific and power board customizations, microcontroller (MCU) settings, and pinout functionality. The drive algorithm takes advantage of the MCU peripherals, ensuring the CPU does not have a big overhead and optimizes memory usage and resource consumption. User layer APIs are generated for a simple run-time configuration and control.

The generated code works on the user hardware or the Microchip available demonstration kit comprising a Power Board, an adapter for the microcontroller, and the Curiosity Nano.

- Multi-Phase Power Board (EV35Z86A)
- AVR16EB32-CNANO MPPB Adapter (EV88N31A)
- AVR16EB32 Curiosity Nano (EV73J36A)

This user guide will go through all the capabilities the interface and generated code offer to the user.

Table of Contents

1. Operating Environment

- MPLAB® X IDE v6.20 or newer
- MPLAB® XC8 Compiler v2.50 or newer
- MPLAB® Code Configurator (MCC) v5.5.1 or newer
- MCC Core v5.7.1 or newer
- Melody Core v2.7.1 or newer
- MPLAB® Data Visualizer plug-in v1.3.1665 or newer
- DFP - Microchip AVR-Ex Series Device Support v2.10.205 or newer

2. Supported MCUs

This library supports the following 8-bit AVR devices:

- AVR16EB20
- AVR16EB28
- AVR16EB32
- AVR32EB20
- AVR32EB28
- AVR32EB32

3. Current Version Capabilities (v1.2.0)

- Motor specification in the interface and configuration file (BLDC)
- Trapezoidal drive
- Sinusoidal drive (Space Vector Pulse-Width Modulation (SVPWM) profile, Saddle profile)
- Hall sensor feedback support
- Back-Electromotive Force (BEMF) sensing (only for Trapezoidal mode)
- Motor–Drive (Rotor-Stator) synchronization (Open loop)
- Phase Advance selection
- Start and stop ramps
- MPLAB Data Visualizer Run Time (DVRT) communication support for plotting parameters in real time
- Proportional-Integral (PI) algorithm with fixed parameters for Closed Loop control over speed regulation
- Fault support for Hardware Peak Overcurrent Protection (Peak OCP) and Software Average Overcurrent Protection (Avg OCP), Overvoltage Protection (OVP), Undervoltage Protection (UVP), Overtemperature Protection (OTEMP), and Stall Detection and Hall sensor disconnection
- Current, Voltage Bus, Temperature and Potentiometer analog measurements at run time
- Variable switching frequency from 15 kHz to 45 kHz
- Low-speed sensored control (5% of the motor's nominal speed)
- Hall sensors auto-detection algorithm at start-up

4. Motor Control Library Specifications

- Motor Type support: 3-phase BLDC or PMSM
- Feedback type: Sensorless with BEMF measurement or HALL sensor
- PWM Drive frequency: Configurable from 15 kHz to 45 kHz
- Drive Modulation capabilities: Block Commutation (Trapezoidal), Sinusoidal (Sinus, SVPWM, Saddle Mode)
- Drive to Feedback synchronization method: Phase-Locked Loop (PLL) approximation
- Electrical RPM modulated frequency: 20-2000 Hz
- Synchronization lost for speeds lower than: BEMF ~15% Motor RPM; Hall ~20 Hz
- Ramp APIs: Single Ramp-up and Ramp-down with only two points (no multipoint ramp)
- Minimum to maximum speed acceleration time: 1.6384s (one bit increment from 16-bit speed)
- Current measurement capabilities: Comparator peak trigger and Analog-to-Digital (ADC) averaged accumulation
- ADC resolution: 12-bit single measurement/16-bit average accumulation
- Dead-Time control: 8-bit control with 50 ns steps, 0-12750 ns
- Open-loop response time on step change: 300-500 ms
- Closed-loop response time on step change: 500-750 ms
- Fault capabilities: Over-Voltage Protection (OVP), Under-Voltage Protection (UVP), Temperature monitoring, Over-Current Protection (OCP) – peak and average
- Fault response time (Peak current using comparator): Minimum 100-150 ns (see Figure 4-1)
- Fault response time (using ADC): 30-35 ms with analog filter enabled (software filtering to get smoother values) or 1.5-2 ms with analog filter disabled (see Figure 4-2)
- Feedback response time: One drive PWM period (50 us for 20 kHz drive)
- Total Interrupt window occupation in Sinusoidal mode: ~23 us (see Figure 4-3)
- Total Interrupt window occupation in Trapezoidal mode: ~19 µs between sectors or 30 µs at sector switch (see Figure 4-4)
- Delay safe time between stop and restart: Ramp-down duration [ms] divided by two is the safe time between stop and restart after the ramp-down finish
- Phase advance control: Fixed 0-90 degrees
- Hall sensor detection latency: One drive PWM period for detection and from 30 to 120 electrical degrees for adjustment
- BEMF detection latency: One drive PWM period for detection and from 30 to 120 electrical degrees for adjustment
- Sinusoidal distortion capabilities: No
- Regen breaking capabilities: No
- Start while the motor is spinning capabilities (windmilling): No
- Field weakening capabilities: No

Figure 4-1. Peak Current Trigger Fault Protection Response Time

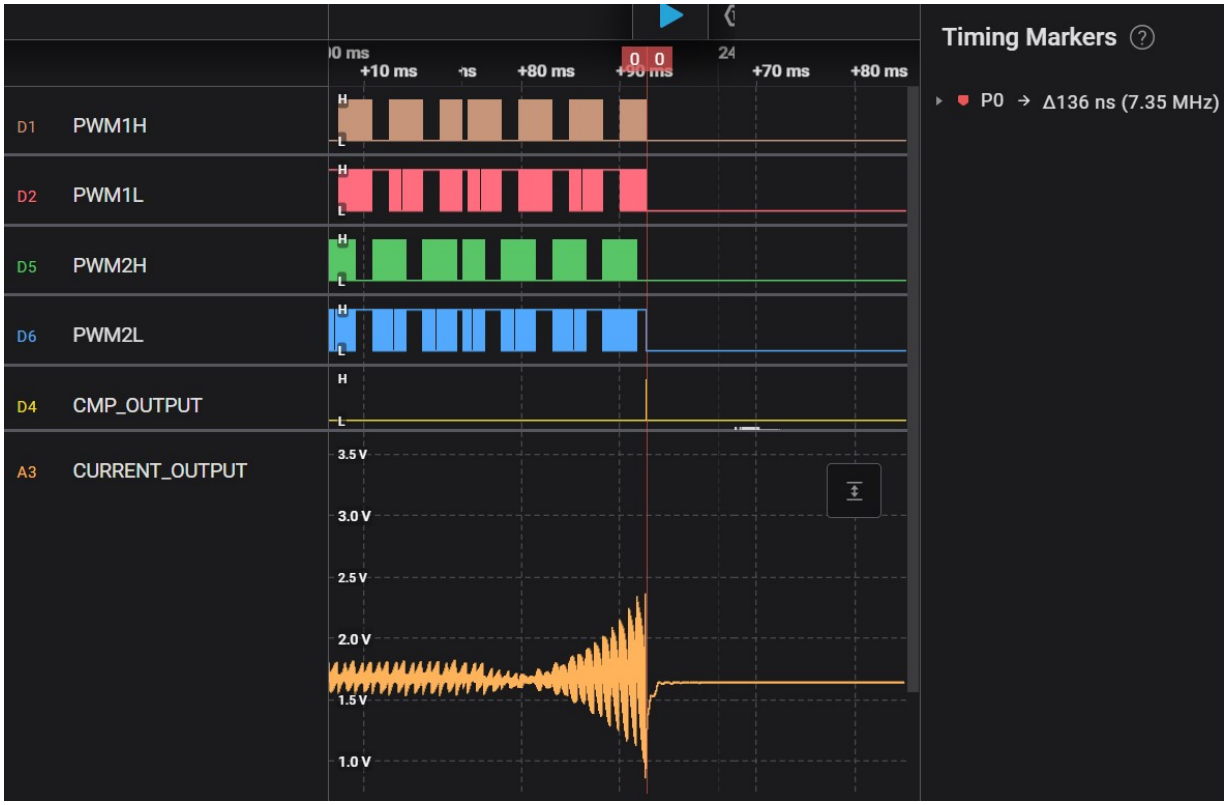


Figure 4-2. Overvoltage Fault Protection Response Time

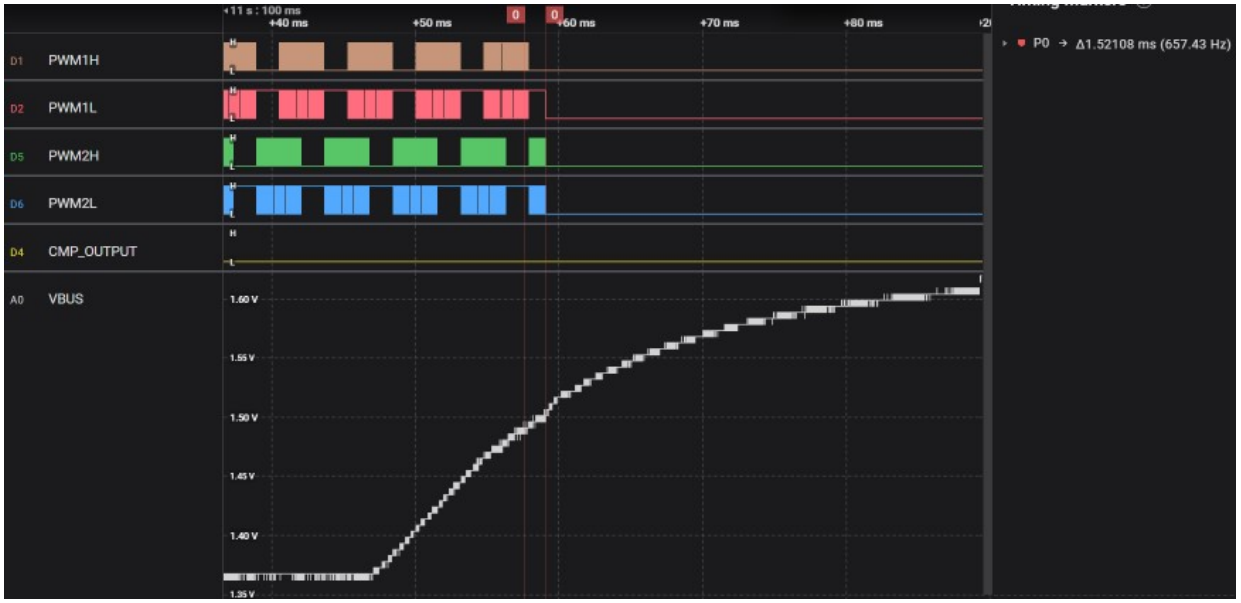


Figure 4-3. Interrupt Window Occupation in Sinusoidal Drive Mode

Figure 4-4. Interrupt Window Occupation in Trapezoidal Drive Mode

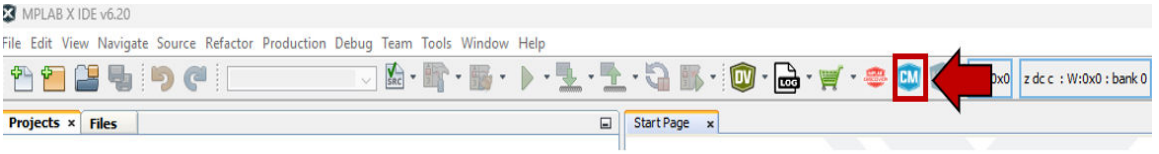
Based on the MOSFET switching frequency, the Interrupt window is split into two consecutive Interrupt Service Routines (ISRs) or executed in a single ISR: When the PWM frequency is above 20 kHz, the computation is done during two ISRs, when under 20 kHz, only one ISR. Examples for 20 kHz and 40 kHz, see the figure below:

Figure 4-5. Motor Interrupt Window Time Diagram

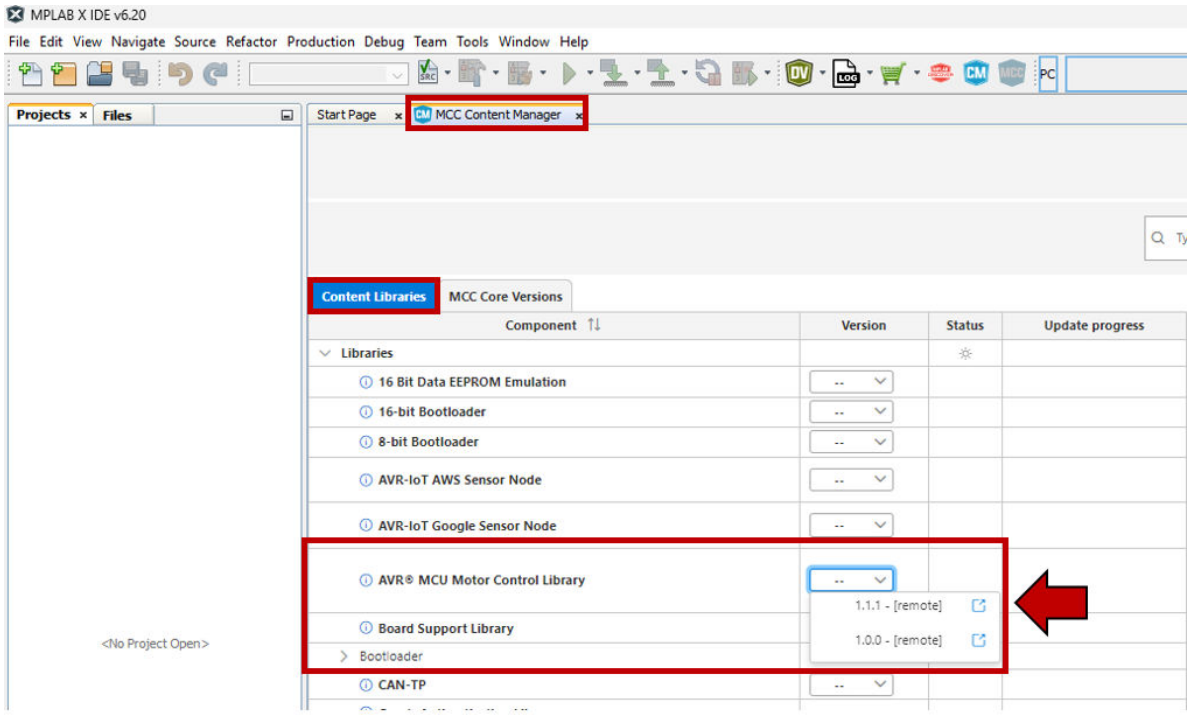
5. Installation

This section presents the installation steps for the MPLAB Code Configurator (MCC) version, how to use the embedded version, and how to complete the first programming and debugging.

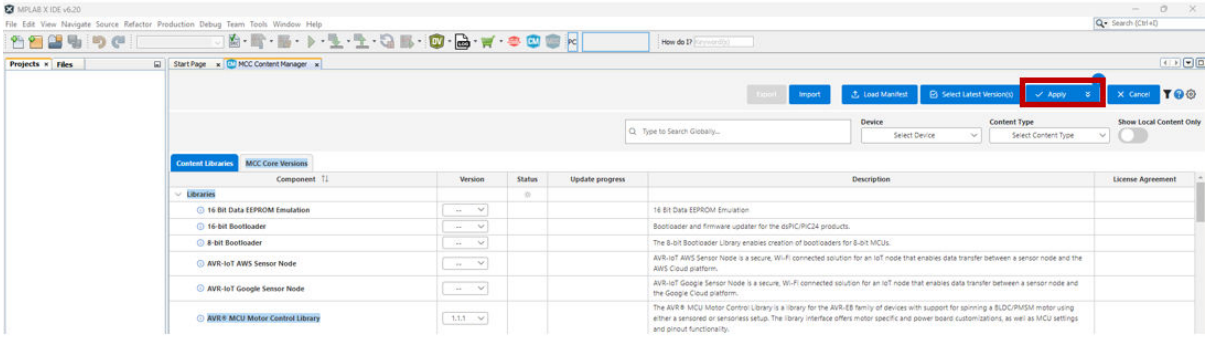
1. Open MPLAB X IDE version 6.20 or newer.
2. Open Content Manager.



3. In the **MCC Content Manager** tab, find “AVR MCU Motor Control Library” under **Content Libraries** and select the latest version.

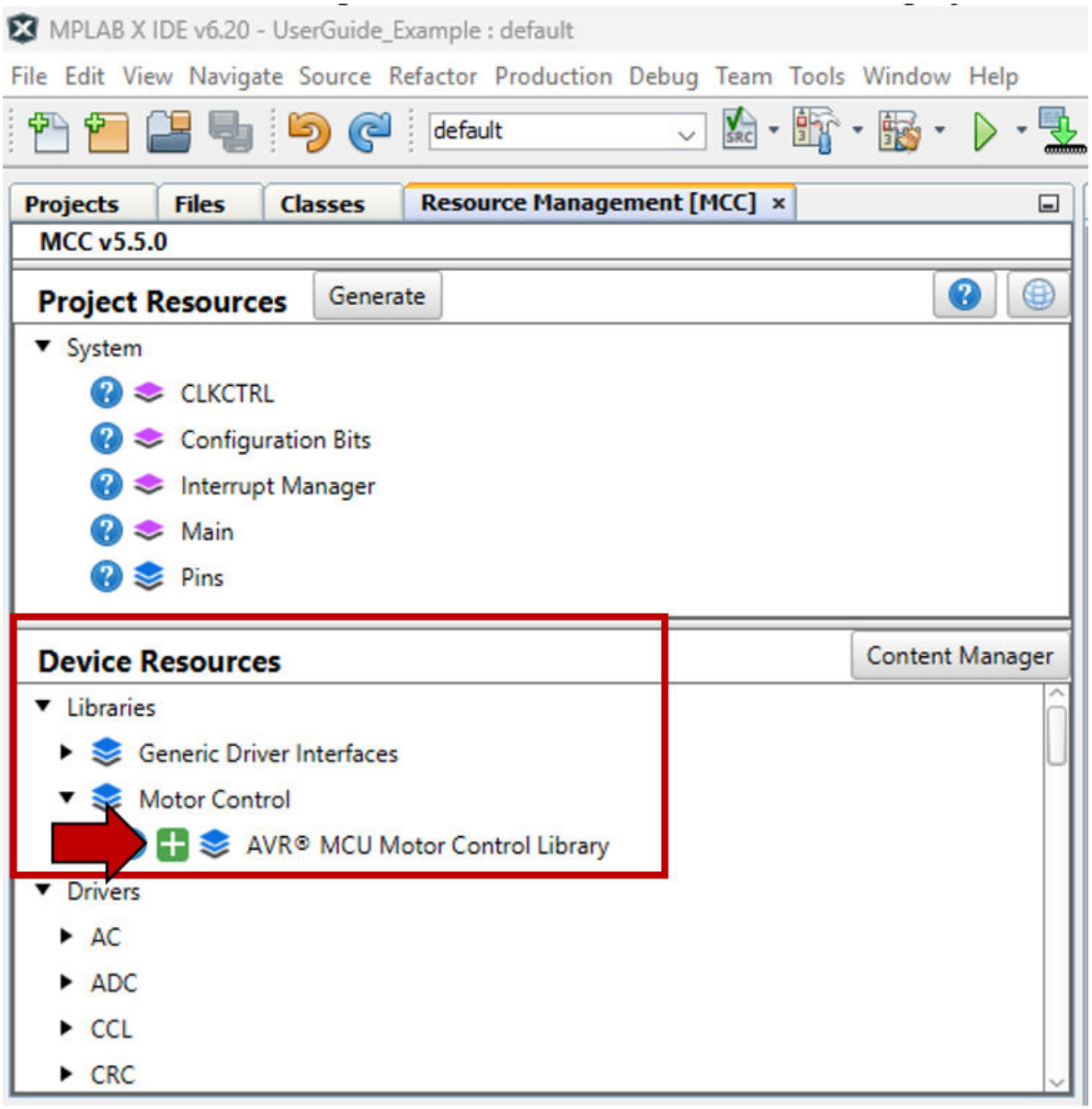


4. Click the **Apply** button to install the library and dependencies in MCC.



5. The installation is now complete.

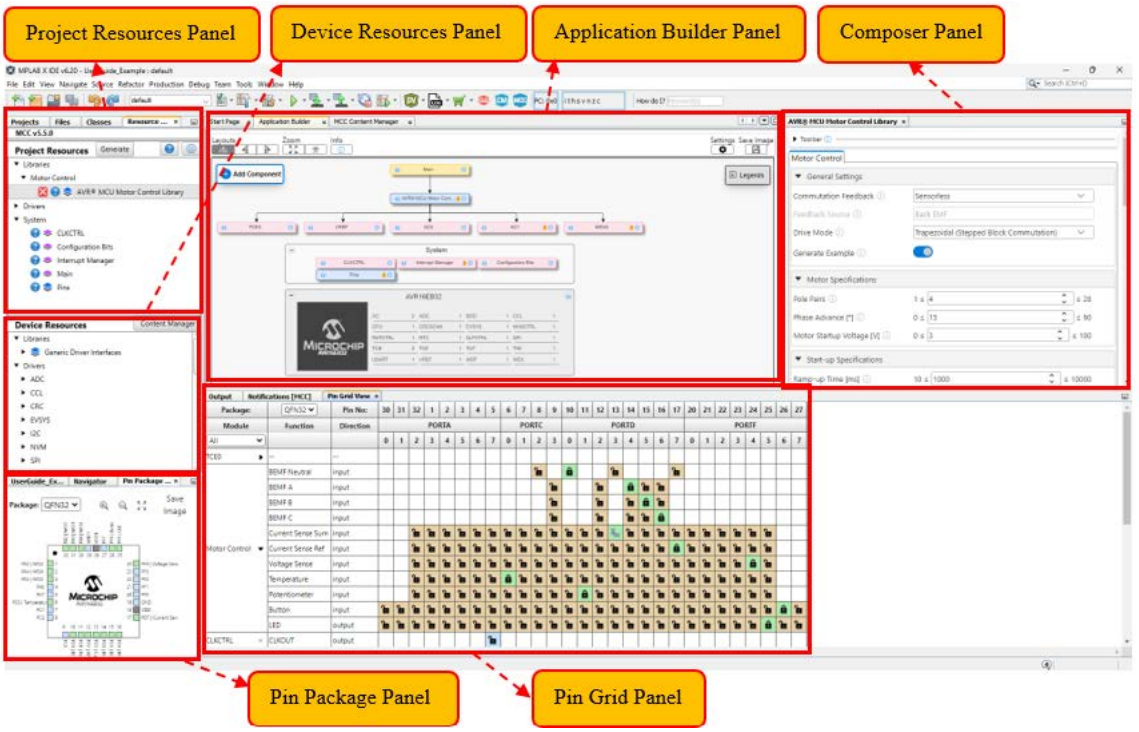
After creating a new project using the supported MCUs and opening MCC, the library is available and can be added from the Device Resources window. The demonstration chapter will describe how to create a new project and get to this step.



6. Configurations

The picture below presents the first interface the user encounters after adding the library to the project. The interface is comprised of the following elements:

- **Resource Management Panel:** The Resource Management Area has two views: The tree and flat views. Both provide access to the complete list of software/peripheral components and the selected components for the current project configuration.
- **Application Builder Panel:** A structured relationship manager with the used resources, providing a clear visualization of a component's related dependencies and context in the project
- **Composer Panel:** When a peripheral, library or other external component is selected from the Project Resources area, its corresponding configuration GUI is displayed in the Composer area
- **Pins Package Panel:** Displays pins in graphical view and provides a zooming feature enabled by the scrolling mouse wheel when hovering with the cursor over the package view
- **Pin Grid Panel:** Displays pins in a structured table format, with lock and unlock features



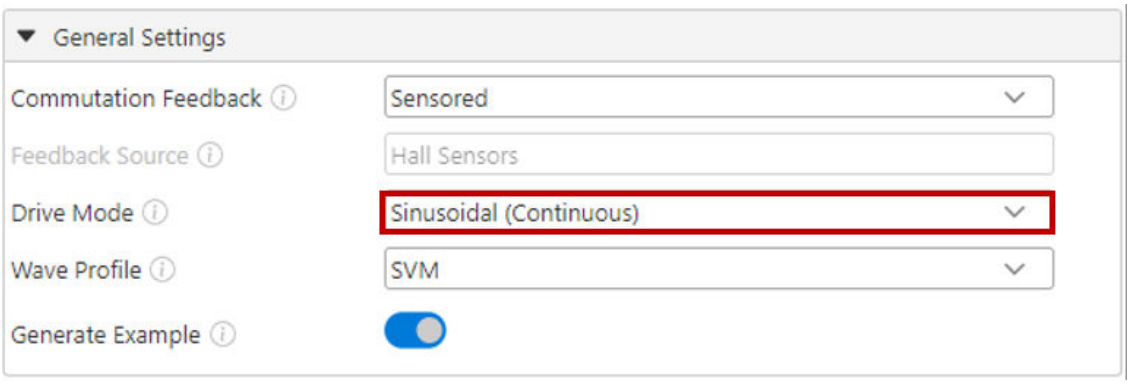
6.1 Composer Panel

The Composer Panel includes all the configurations related to the library functionality, while the Pin Grid Panel contains the exact pin functionality and position.

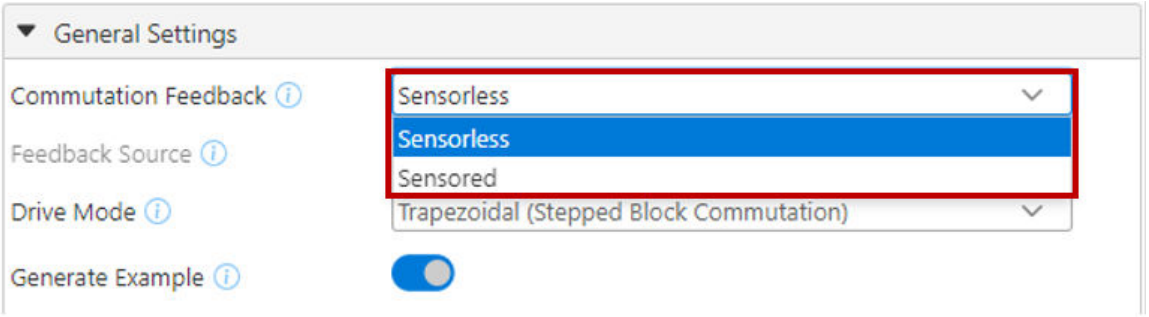
Taken individually, each sector in the Motor Control library has the following options:

1. **General Settings**

   The general settings keep the control settings for drive and feedback modes while allowing the user to generate a demonstration example.



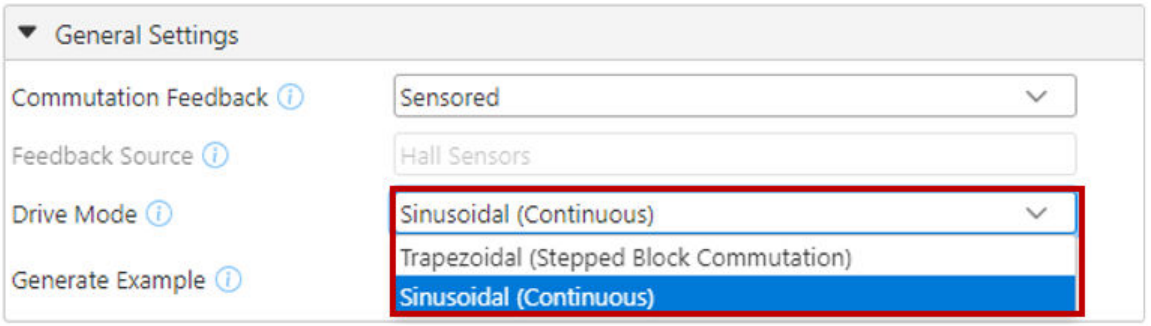
- Commutation Feedback: A choice for motor synchronization source between sensors or sensorless feedback algorithm



- Feedback Source: Allows to change the source of the feedback itself

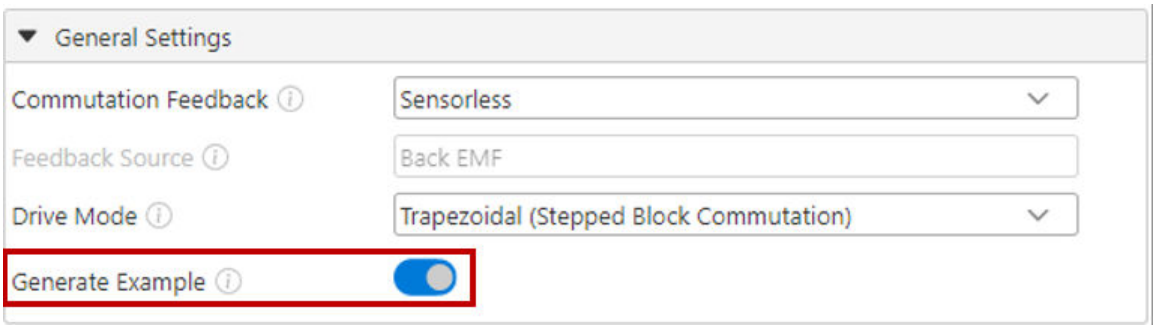
  For Sensorless feedback, the option is grayed out, as only BEMF is accepted as a source in this version. It is fixed to Hall for the current version (if Sensored), but the Quadrature Encoder and the Inductive sensors will be among the selection options in future versions.
- Drive Mode: The actual PWM Low-Frequency Drive Modulation mode can be selected from Trapezoidal drive, also known as six-step commutation, and Sinusoidal drive

  Sinusoidal drive modulation has three different Look-up Tables (LUT) generating options: normal Sinus, Space Vector Modulation (SVM) or Saddle mode

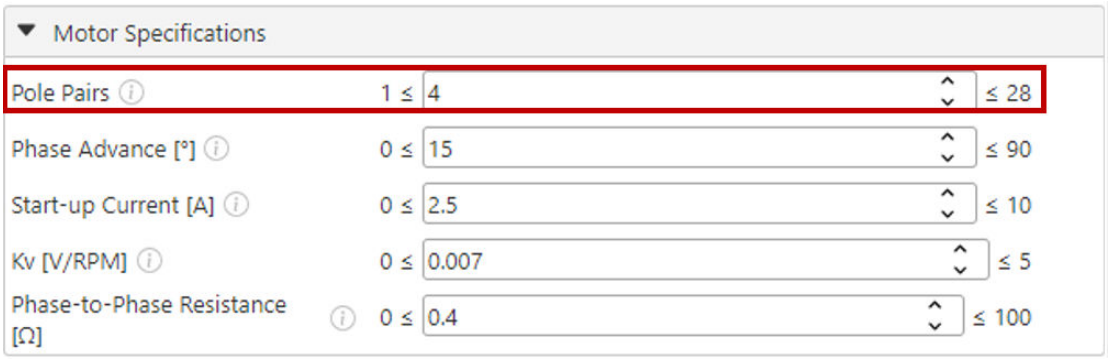


- Generate Example: This option enables the generation an example application that uses the necessary APIs to spin the motor with a potentiometer-controlled speed. The initialization

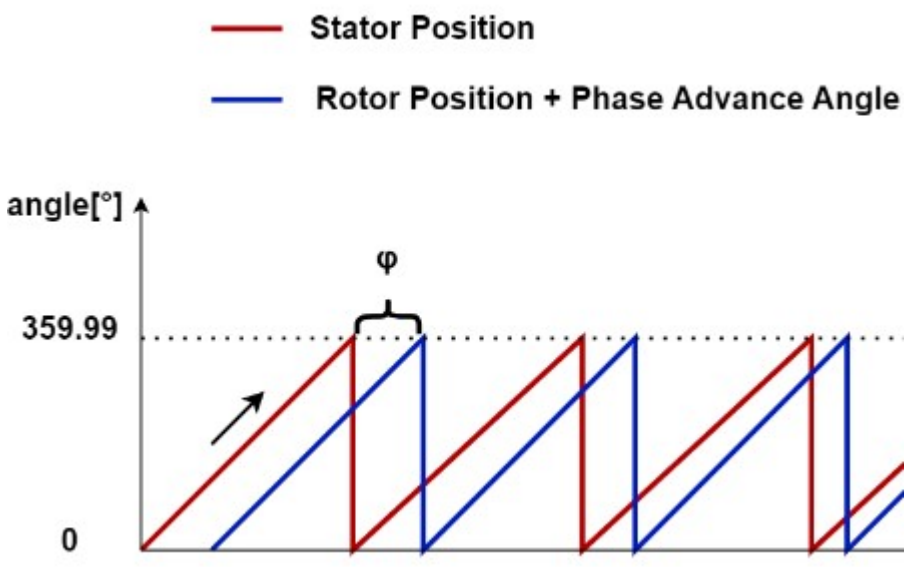
and loop logic functions are called in the `main.c` file demo.



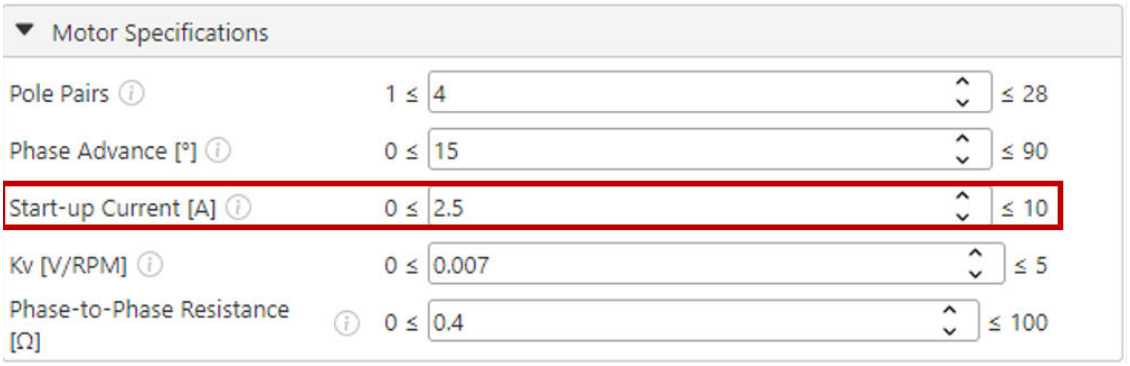
2. **Motor Specifications**
   In this section, the user can customize the motor-related characteristics used for control and speed measurement.
   - Pole pairs: This value is taken from the *Number of Poles* section in the Motor data sheet – divide that value by two to get the pole pairs value. As the pole pair information is used for speed measurement, the motor will spin even if this is written wrong, but the measured speed will not be accurate.



   - Phase Advance: Can range from 0 to 90 electrical degrees. The sensing algorithm estimates the rotor's position and then calculates the difference between the stator (driving field) and the rotor (motor field). The result (in electrical degrees) represents how much the drive field is out of phase with the motor field. Based on this difference, the drive speed is updated to keep a constant phase shift between the stator and rotor. If the phase shift is too small, the torque might not be enough to keep the synchronization. If the phase shift is too big, the synchronization is also lost. An optimal value is determined to ensure sufficient torque is produced and power consumption is efficient.

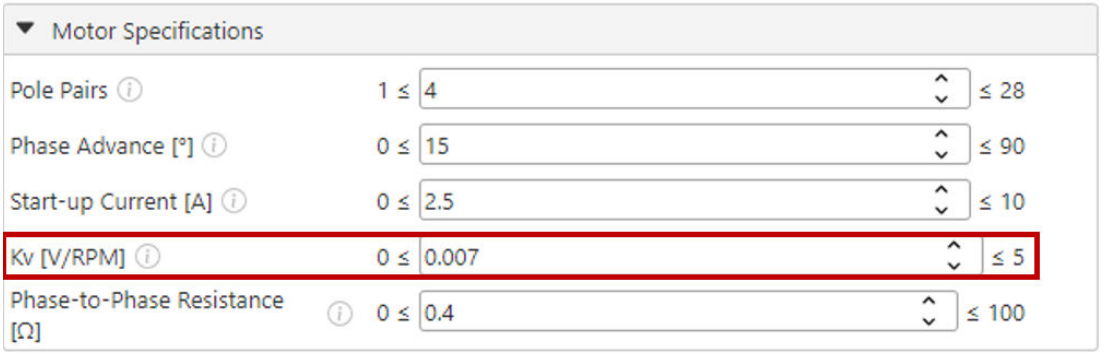


- Start-up Current: The voltage supplied to the motor phases during the start-up phase is given in volts. Based on the voltage supplied, this is transformed into a duty cycle in software. If the supply voltage is below the start-up voltage, the application will show an error message, and the motor ramp-up will fail.

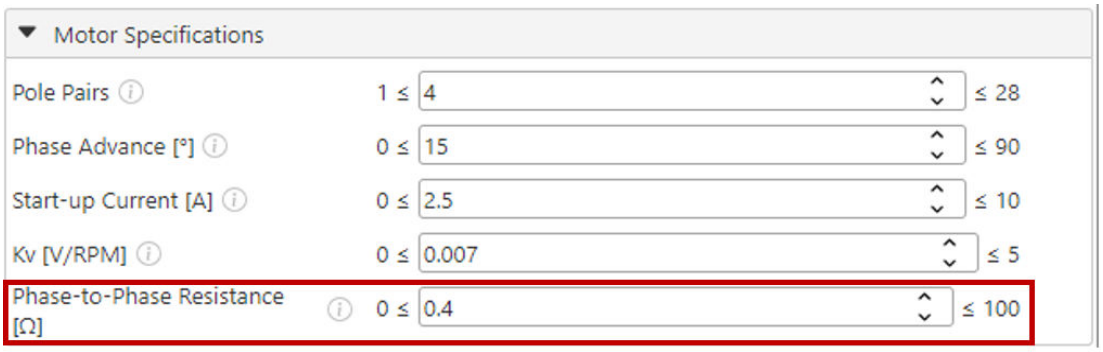


- Kv [V/RPM]: In the library, it represents the inverse of the motor Speed Constant (also known as BEMF constant) and it is measured in volts per revolutions per minute (RPM). You can find this value in the motor data sheet. It can be measured when the motor is driven as a generator without load. The voltage measured on the terminals is perfectly proportional to the RPM per the Kv of the motor/generator.

  E.g.: Measuring 1V while spinning the motor at 100 RPM results in Kv = 100, while in the library, it is 1/100, so the user has to enter 0.01.



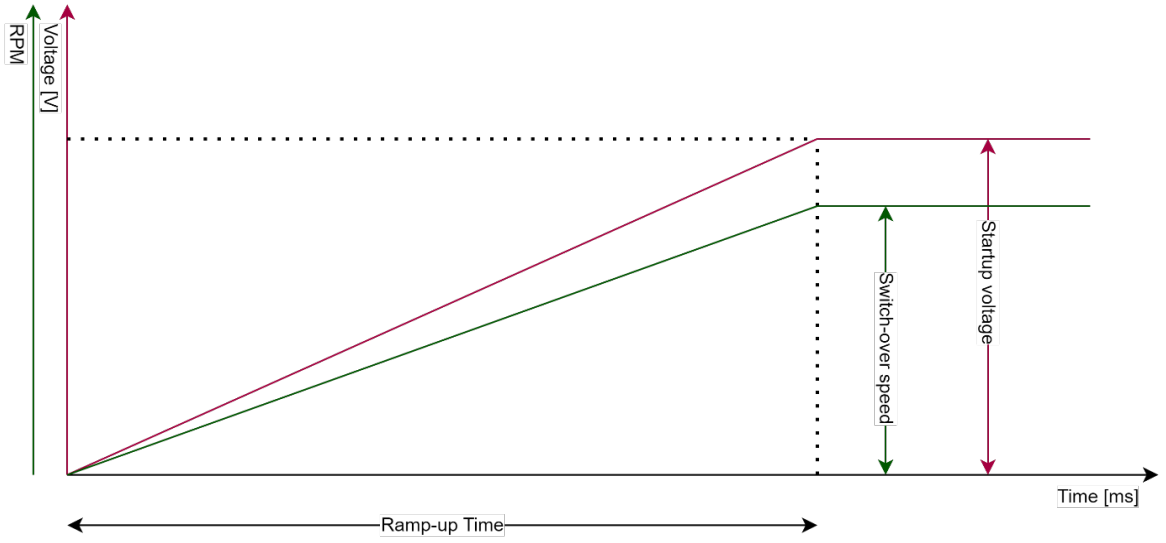
– Phase-to-Phase Resistance: The electrical resistance (in Ohms) measured between any two phases of the motor.



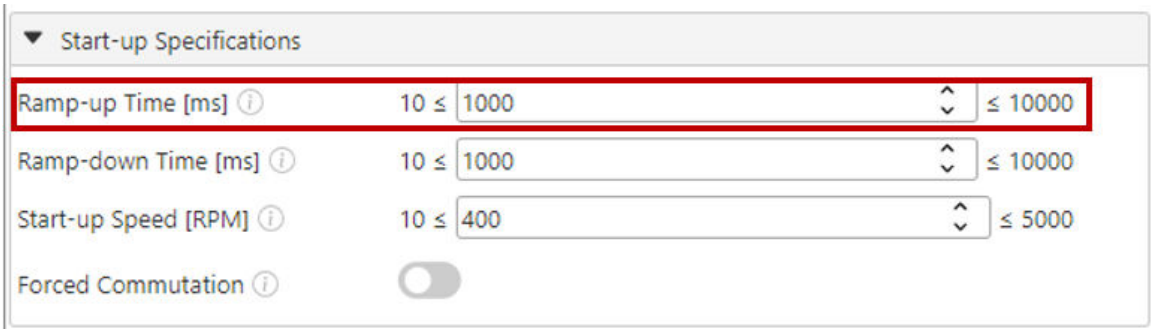
When Sensored mode is selected in *General Settings>Commutation Feedback*, more options are available in the Motor Specification settings.

3. **Start-up Specifications**

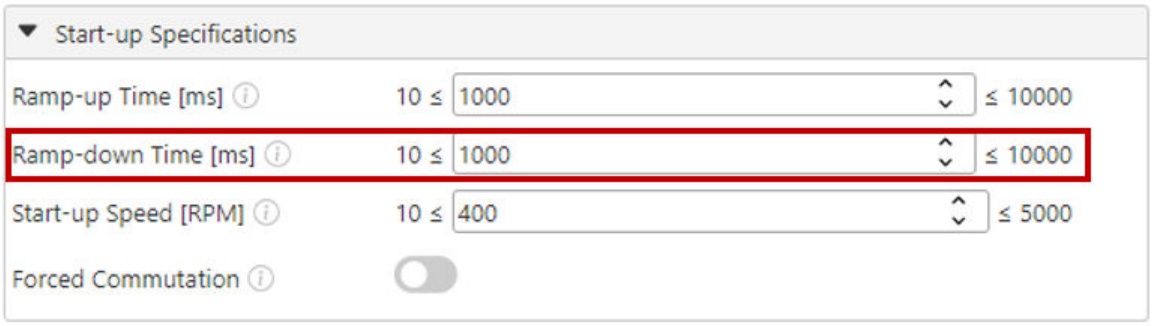
In this section, the user can customize the time it takes for the drive voltage (identified by the PWM duty cycle) to ramp up from 0 to the value dictated in the Motor Start-up Voltage parameter and back to 0 in a ramp down situation. The parameters are linked to the motor start-up routine and include switch-over speed transition from forced commutation to Drive and Motor synchronization.



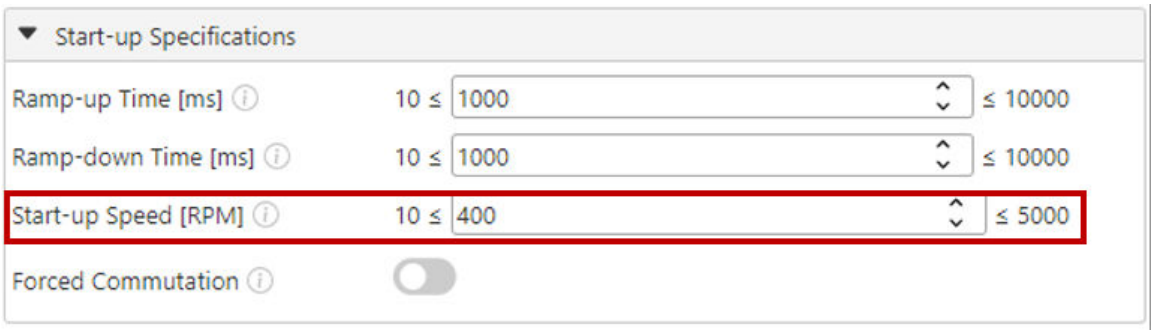
– Ramp-up Time: The time, in milliseconds, provided to the start-up algorithm to ramp up the voltage from 0V to the desired value and to ramp up the drive speed from 0 RPM to the desired switchover speed



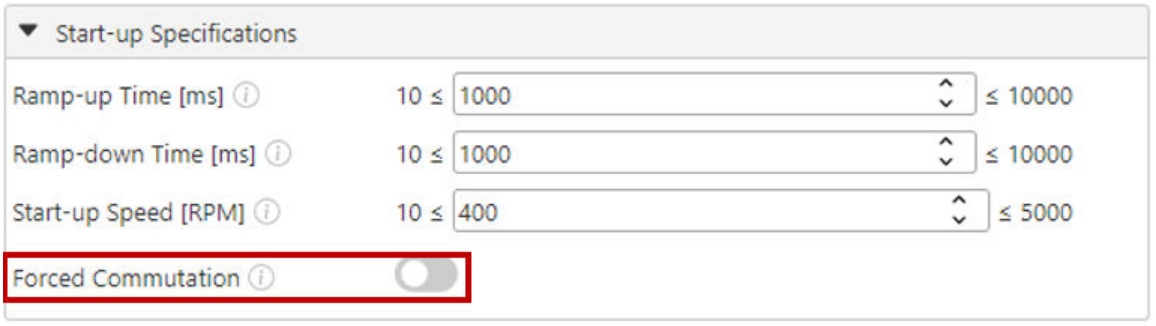
- Ramp-down Time: The time, in milliseconds, provided to the motor-stopping algorithm to ramp down the voltage from the current value to 0V and also to ramp down the drive speed from the current RPM to 0 RPM



- Start-up Speed: Used as target RPM speed by the ramp-up algorithm, spinning up from zero to the target speed. This is the speed at which the switchover between forced commutation and synchronization loop starts. If Sensorless mode is used, this parameter must be set to at least 15-20% of the nominal speed of the motor.



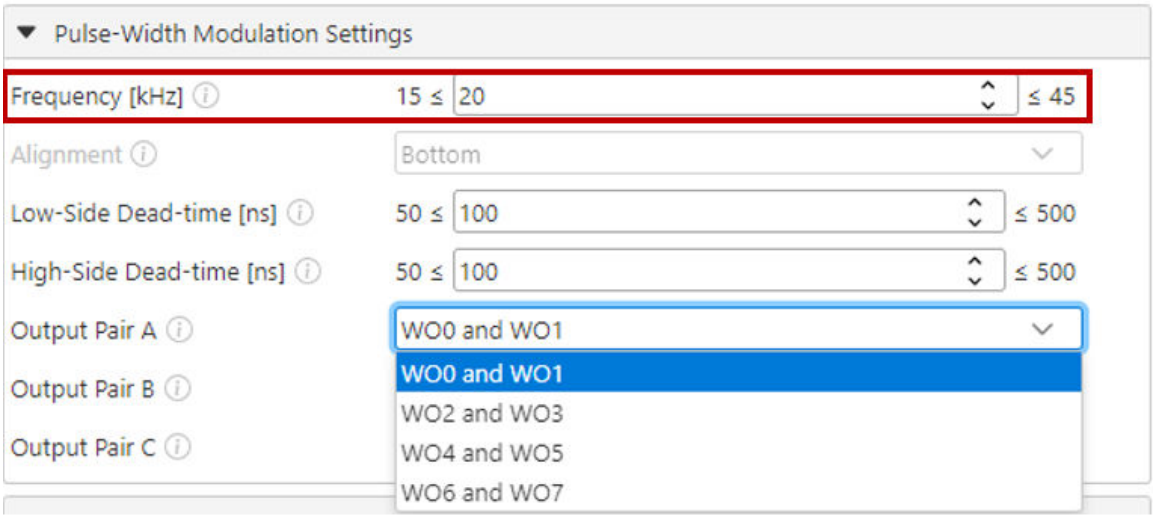
- Forced Commutation: When this function is enabled, the drive will not synchronize with the motor feedback. Instead, it will stay forced to a desired voltage and speed.



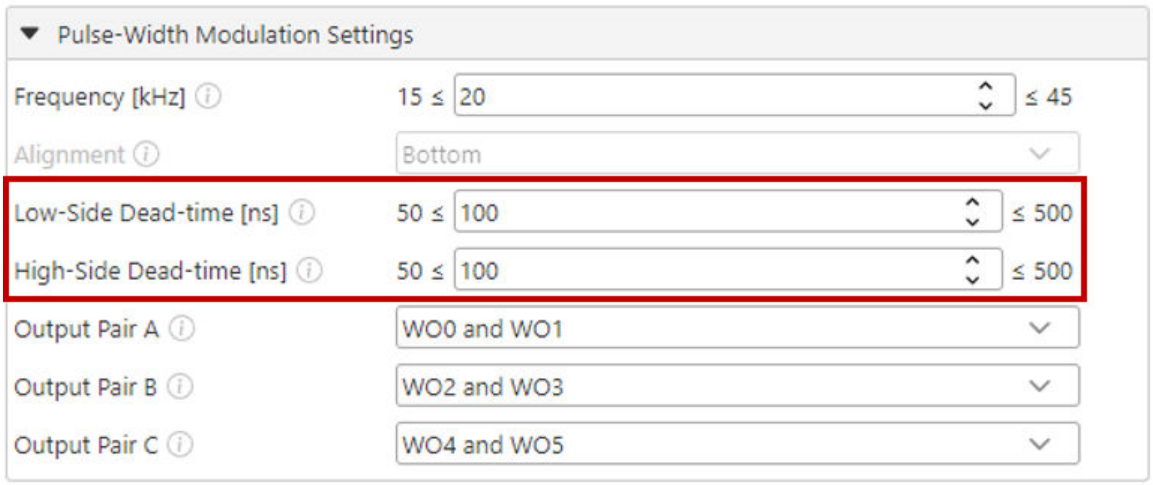
4. **Pulse-Width Modulation Settings**

In this section, the user can customize the driving PWM signals, starting from the main frequency, dead-time insertion to the output pair configuration. This allows for a low-level peripheric configuration to the abstract motor control.

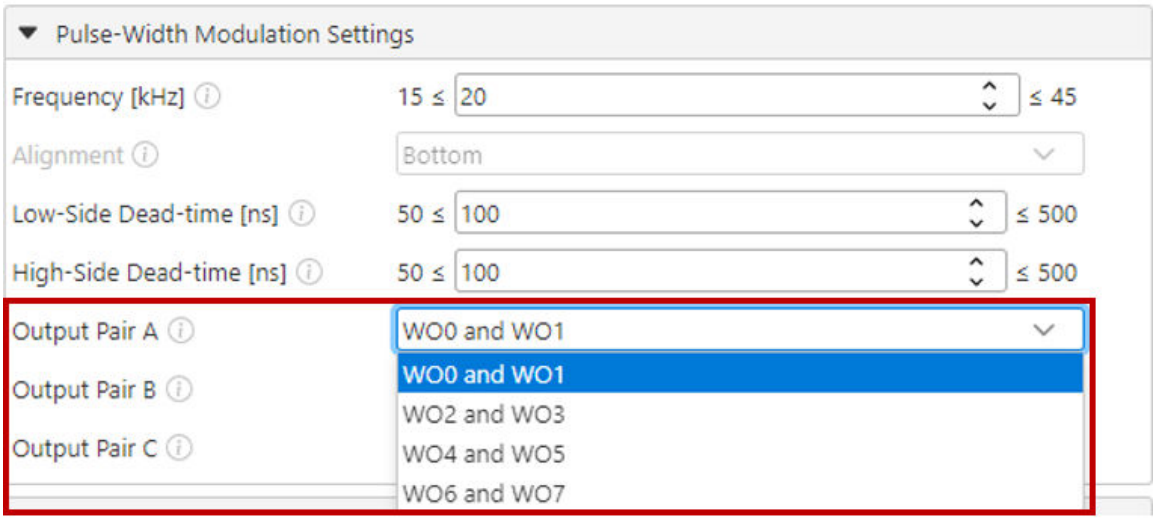
- Frequency: The PWM frequency for any drive mode, user selectable from 15.0 to 45.0 kHz



- Low-Side and High-Side Dead-time: The complementary signals that drive the High-Side and Low-Side MOSFETs need a dead time, when both signals are low, to prevent current shoot-through and hardware damage. The value is in nanoseconds, and the step is dependent on the MCU main clock period (e.g., for a 20 MHz main clock, there will be a 50 ns dead time step).



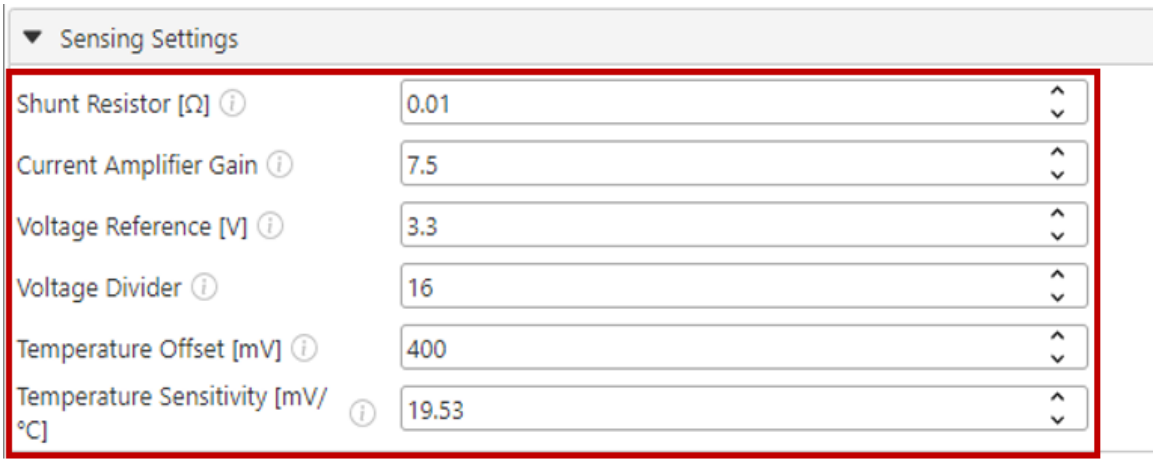
- Output Pair [A, B, C]: The library allows the driving complementary pair signals to be placed at different port pins. This option permits the change of a given pair on the MCU Input/Output (I/O). The capability to change the position of the driving signals provides flexibility during the hardware design of the control board.



5. **Sensing Settings**

   In this section, the user can customize the physical values of the components for the sensing layer.

   - Shunt Resistor: The current sense shunt resistance, given in ohms, dependent on the power board
   - Current Amplifier Gain: The current sense amplifier gain, dependent on the power board
   - Voltage Reference: The ADC voltage reference value must be the V_{DD} value of the microcontroller. It is used for ADC measurement computation and interpretation. Changing this value will not change the ADC reference because it is set to V_{DD}.
   - Voltage Divider: The VBUS measuring resistor ladder dividing factor
   - Temperature Offset: The added ADC measurement point offset for the temperature sensor, dependent on the power board
   - Temperature Sensitivity: The used temperature sensor sensitivity as provided by the device data sheet



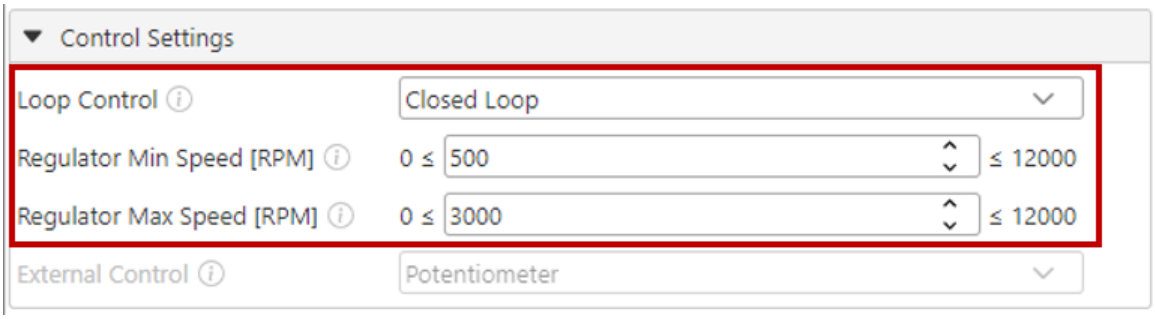
6. **Control Settings**

   In this section, the user can customize the control layer wrapper and the input for the dynamic speed requirement.

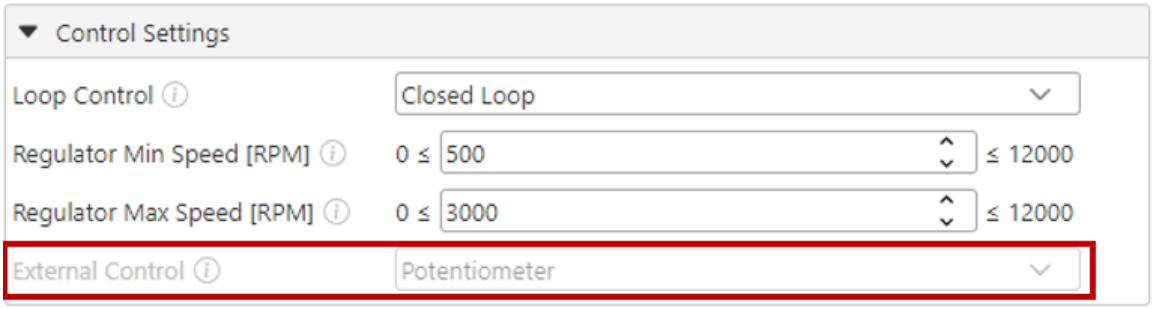
   - Loop Control: This option allows the user to switch between the Open Loop Control (Drive and Motor synchronization without speed regulation) and Closed Loop Control (where a

slower control layer is added for speed control besides the drive and motor synchronization control). When the Forced Commutation option is enabled, this option is unavailable.

- Regulator Min Speed [RPM]: The minimum speed allowed to be achieved when the potentiometer is at 0%
- Regulator Max Speed [RPM]: The maximum speed allowed to be achieved when the potentiometer is at 100%



- External Control: The selected input source for PWM amplitude (duty cycle). This option is unavailable now and is locked on “Potentiometer” input, meaning that the user can change the motor voltage amplitude (by doing this, the rotational speed is increased or decreased) only from the on-board potentiometer. The user can add their own source, such as a PWM input or a different analog source.

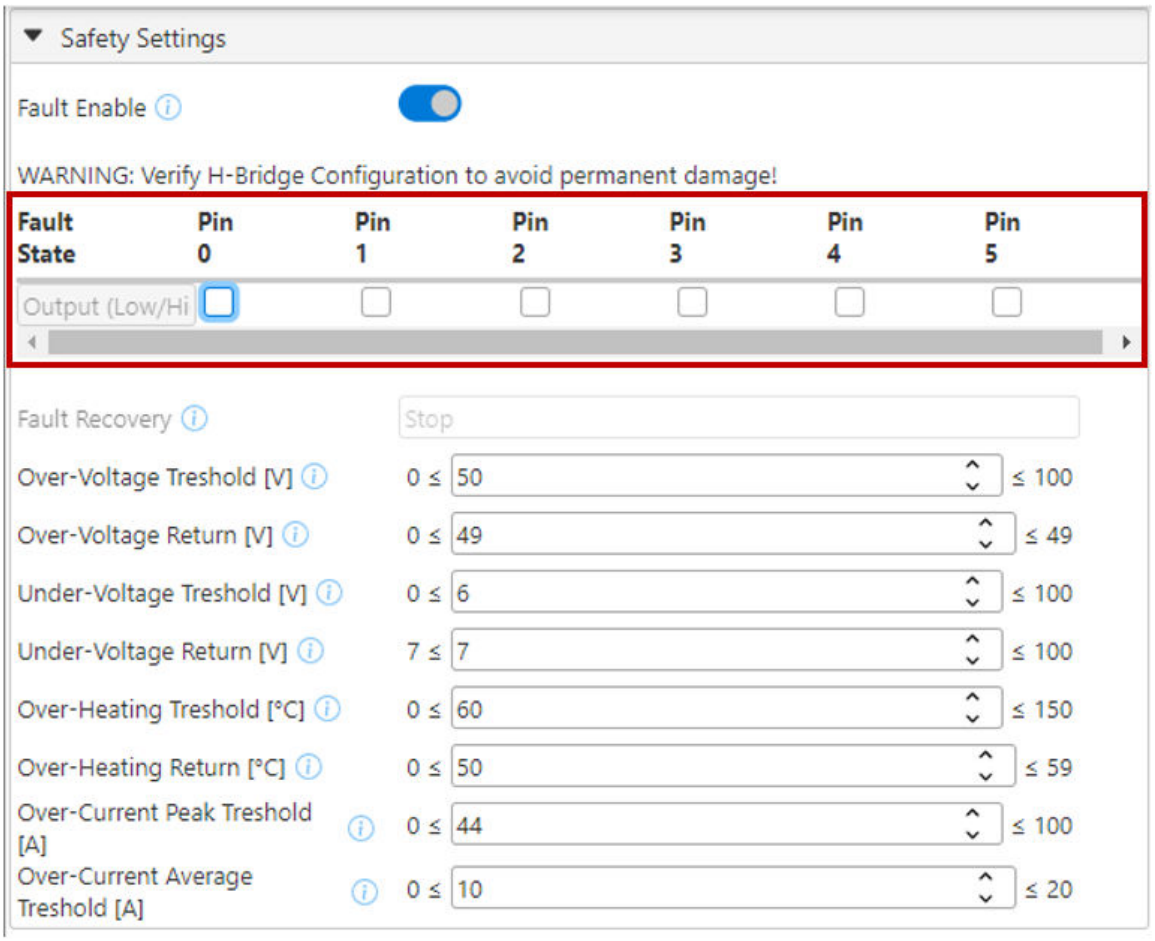


7. **Safety Settings**

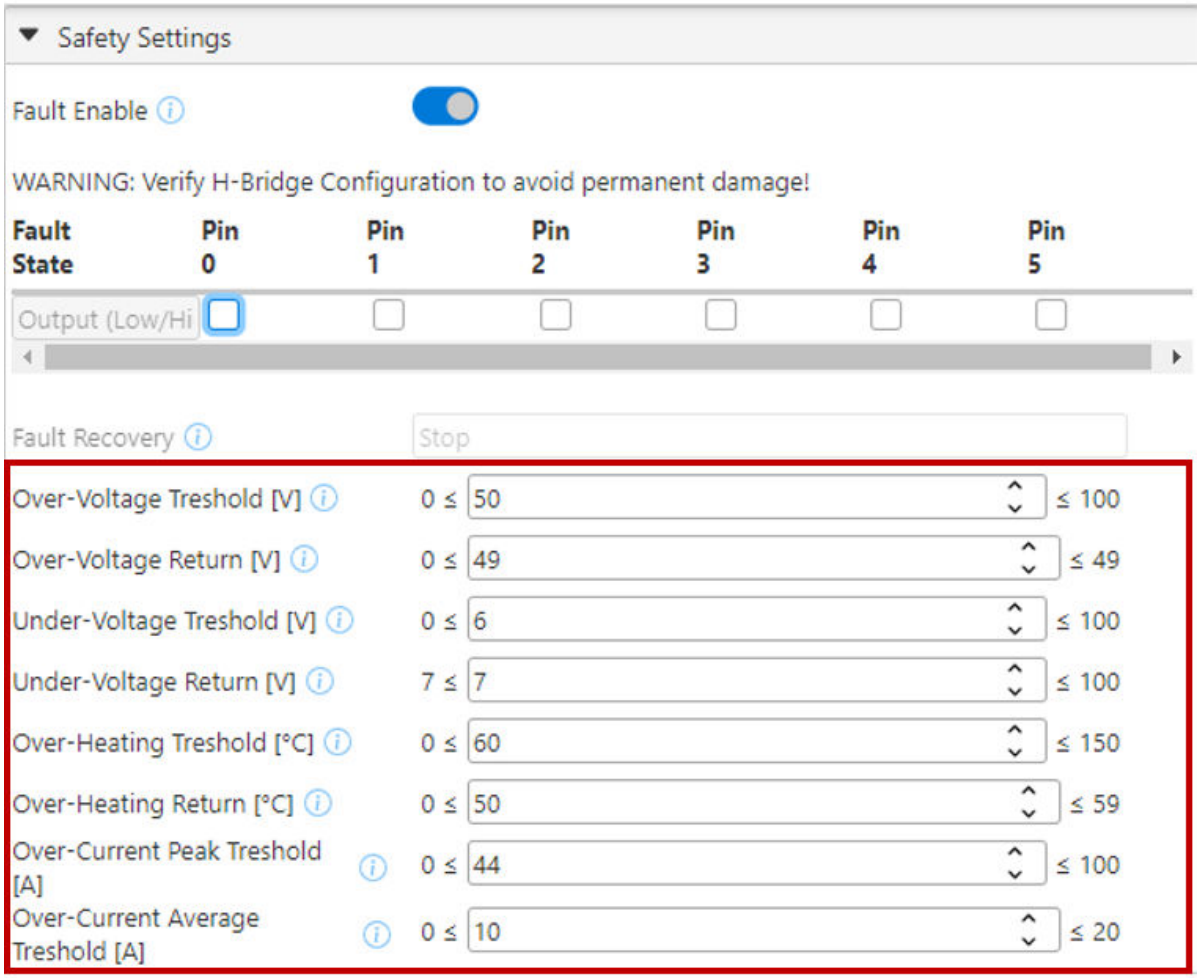
   In this section, the user can customize the fault events and stop the motor when needed to prevent irreversible damage.

   - Fault Enable: An activation switch for the Fault Handling layer. The Fault options are hidden when this option is disabled.

   - Fault State: The state of the driving signals during a Fault condition. If unchecked, it means that the I/O will be set to `0`, if checked, then the the I/O will be set to `1`.



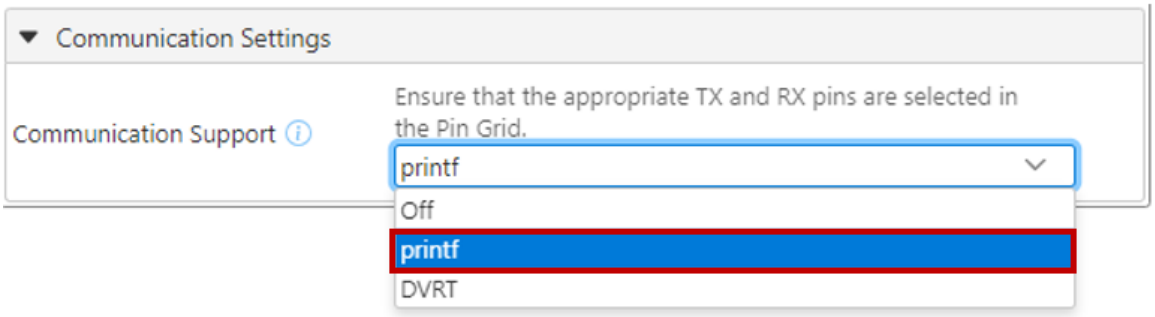
- Over-Voltage Threshold: The voltage bus limit given in volts. It serves to protect the hardware setup from any irreversible damage
- Over-Voltage Return: The voltage bus restore value given in volts. When reaching this value, the OVP fault is cleared.
- Under-Voltage Threshold: The minimum voltage bus limit given in volts. If this value is not reached, the motor will not start.
- Under-Voltage Return: The minimum voltage bus restore value given in volts. When reaching this value, the UVP fault is cleared.
- Over-Heating Threshold: The temperature limit given in degrees Celsius. It serves to protect the MOSFETs or Motor from irreversible damage.
- Over-Heating Return: The temperature restore value given in degrees Celsius. The fault can be cleared, and the motor drive can restart when reaching this value.
- Over-Current Peak Threshold: The peak current limit given in amperes. It serves to protect the MOSFETs from irreversible damage.
- Over-Current Average Threshold: The continuous average current limit given in amperes. It is used for overload protection or if the phase advance difference between the stator and the rotor is too big.



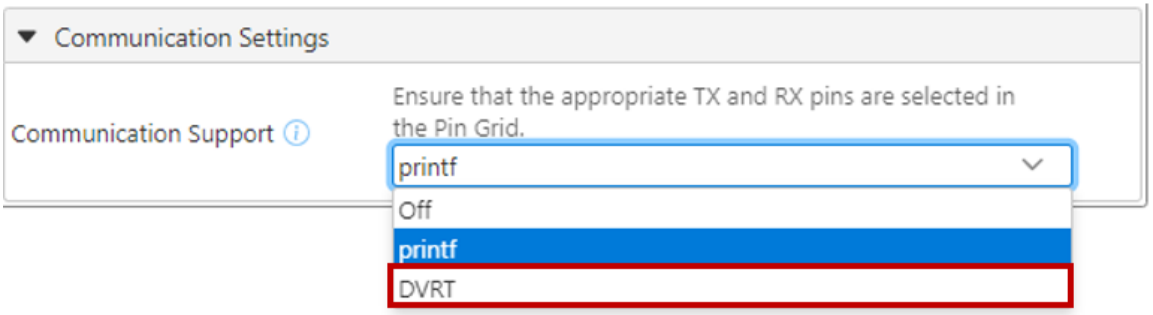
8. **Communication Settings**

   In this section, the user can customize the communication type. The library supports two communication interfaces: serial interface and Data Visualizer Run Time (DVRT). When enabling any of the communication methods, the USART module is added to the used project resources.

   - *Communication Support>printf*: The serial interface is enabled with a default of 460800 baud rate. The reading can be done with any tool, including the MPLAB Data Visualizer extension. The terminal depicts the motor speed in RPM, average current consumption, input voltage, temperature sensor, and the potentiometer value in percentage.

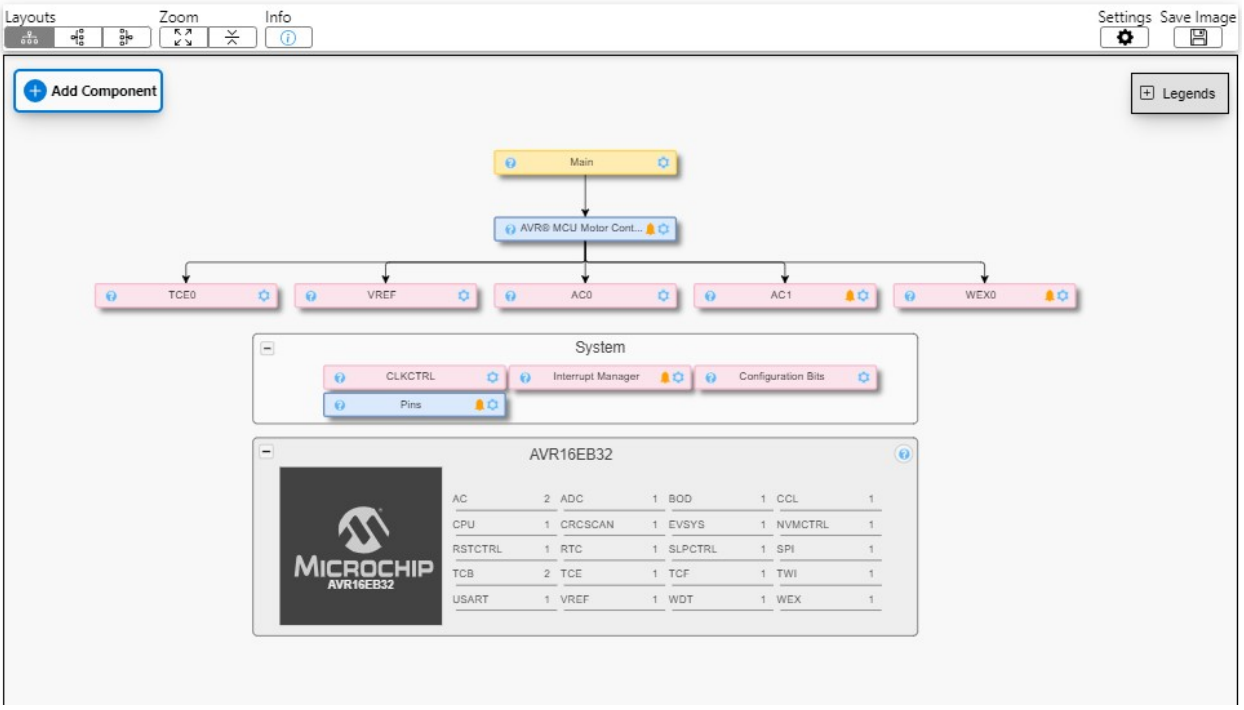


   - *Communication Support>DVRT:* The MPLAB Plug-in Data Visualizer has a run-time option that can plot real time in a graph the desired variables. The generated DVRT option allows for depicting the potentiometer variable and the motor speed variable.



6.2 Application Builder Panel

Figure 6-1. Builder in MCC



6.3 Pin Grid Panel

- **BEMF Neutral:** The neutral connection for the comparator input that detects the motor BEMF
- **BEMF A-C:** The BEMF acquisition lines for the A-C phases of the motor
- **Hall A-C:** The Hall acquisition lines for the A-C phases of the motor
- **Current Sense Sum:** Signal that provides the amplified sum current from the measuring shunt resistor
- **Current Sense Ref:** The reference input for the current sense amplifier
- **Voltage Sense:** Divided input voltage signal
- **Temperature:** Input for the temperature sensor
- **Potentiometer:** Input for the on-board potentiometer that controls the motor speed
- **Button:** Input for the on-board button that controls the motor start and stop state
- **LED:** Output for the on-board indication LED
- **UART:** Serial communication lines

- **WEX0:** The driving signals that connects to the half bridge drivers

Figure 6-2. Pin Grid View With BEMF

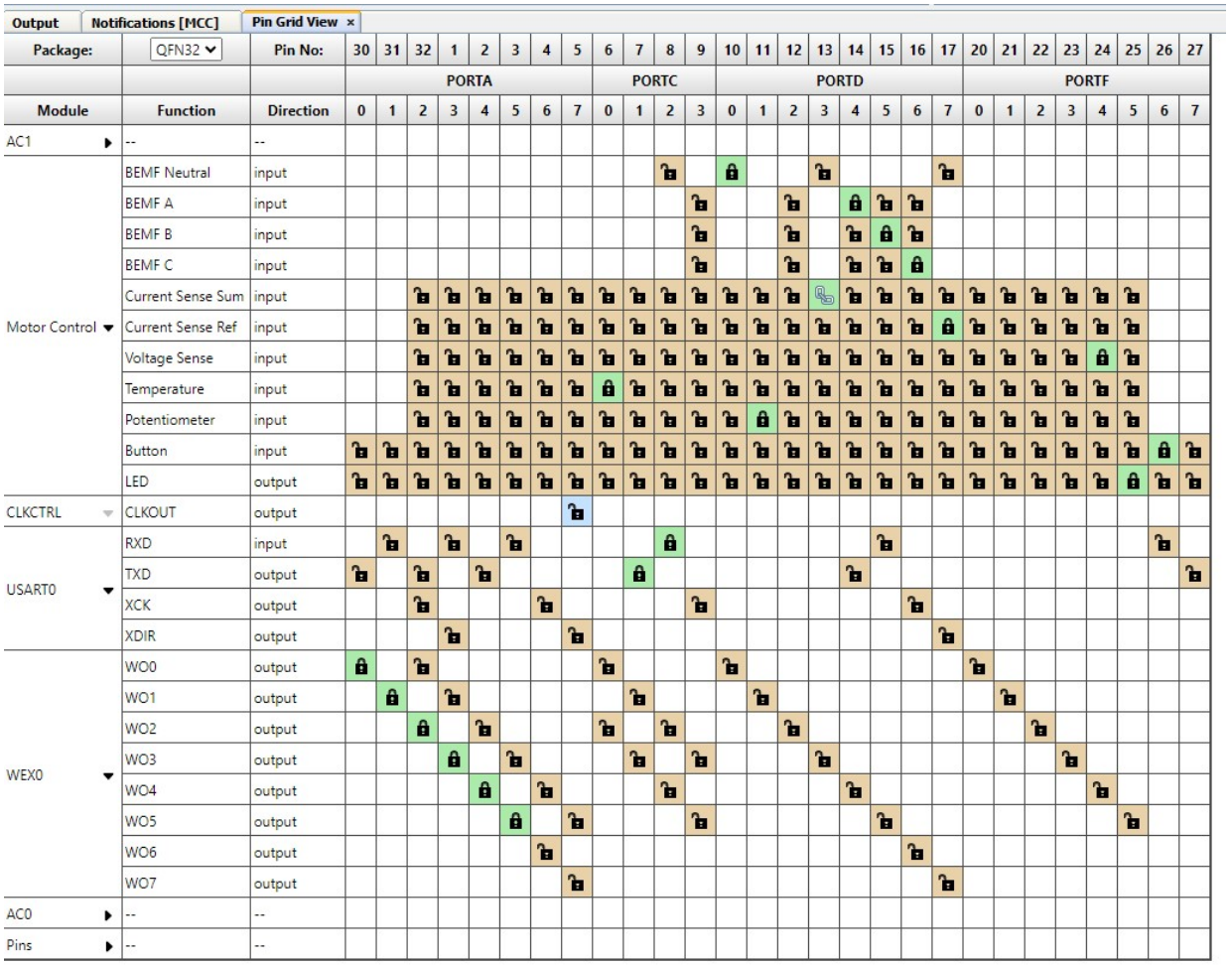
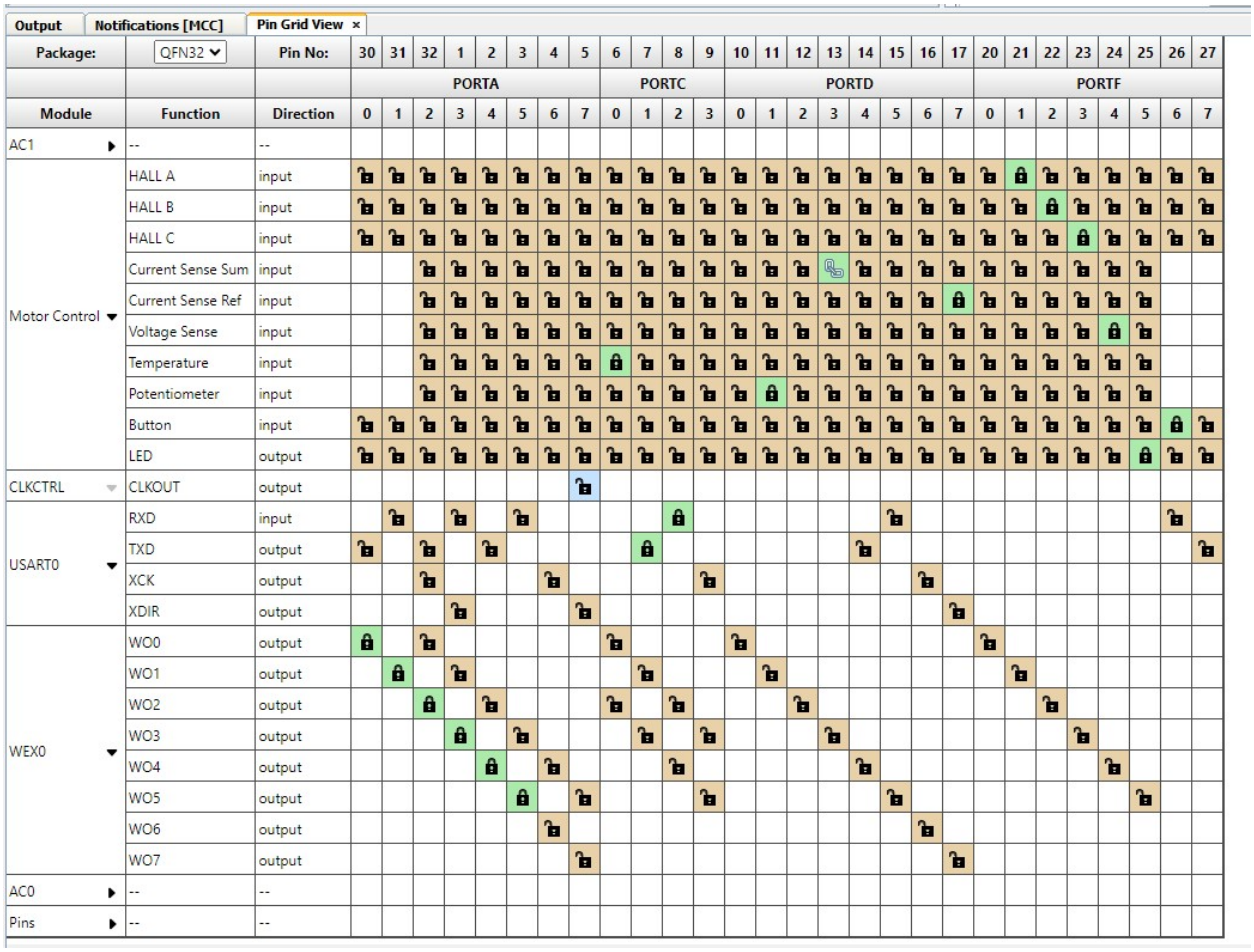
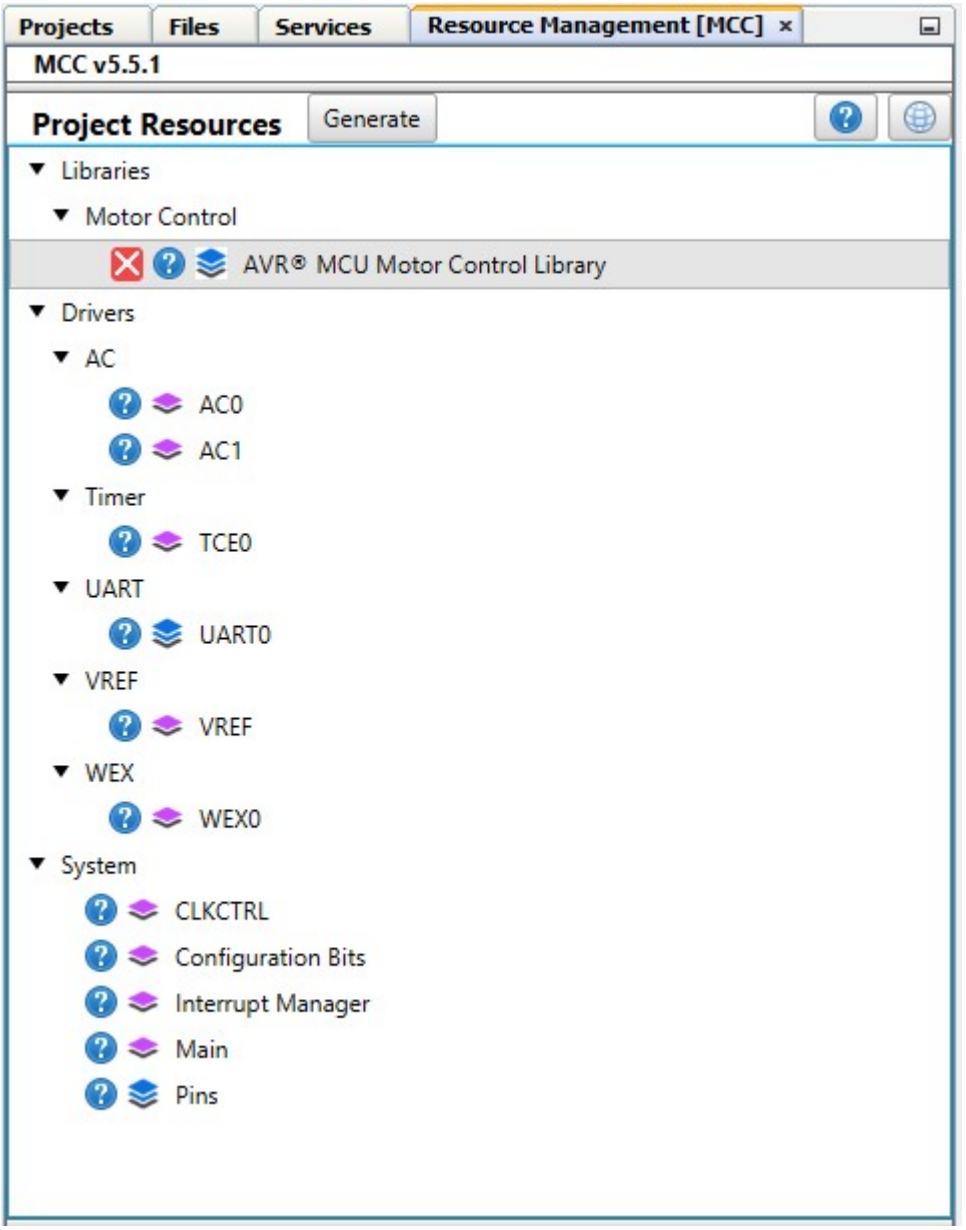


Figure 6-3. Pin Grid View With Hall



6.4 Resource Management Panel



7. Generated APIs

The generated library offers a list of user-friendly APIs that abstract and simplify the low-level configuration.

The APIs from the Motor Control Stack found in `motor_control.h` are the following:

- `MC_Initialize` - The initialization function must be called before any other function
- `MC_StartStop` - Starts or stops the motor depending on which state it is found in when the start-stop event is received
- `MC_DelayMs` - Performs a delay using a timer in the background, without interrupts
- `MC_ReferenceSet` - Sets the reference point for speed in closed loop or the amplitude level in open loop synchronization
- `MC_PotentiometerRead` - Returns potentiometer value, expressed in percentage
- `MC_FastPotentiometerRead` - Returns potentiometer value, expressed in raw Analog-to-Digital (A/D) format
- `MC_VoltageBusRead` - Returns voltage bus value expressed in volts
- `MC_TemperatureRead` - Returns MOSFET transistors temperature, expressed in degrees Celsius
- `MC_CurrentRead` - Returns mean application current consumption, expressed in milliamperes
- `MC_Speed_Get` - Returns the rotational speed expressed in `mc_speed_t`
- `MC_PeriodicHandlerRegister` - Registers a custom software callback for the main application. This function must be called from the main and not the interrupt context.
- `MC_StatusGet` - Returns the state of the motor (MOTOR_IDLE, MOTOR_RUNNING or MOTOR_FAULT), direction of the motor (MC_DIR_CW or MC_DIR_CCW), and the fault status (FAULT_UNDERVOLTAGE, FAULT_OVERVOLTAGE, FAULT_STALL, FAULT_OVERTEMPERATURE or FAULT_OVERCURRENT)

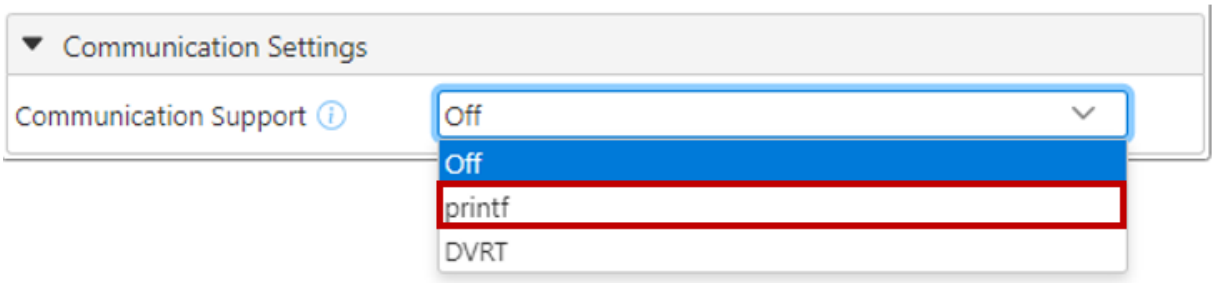
Users must only use the public APIs from the `motor_control.h` file so as not to alter the library's functionality.

8. Communication

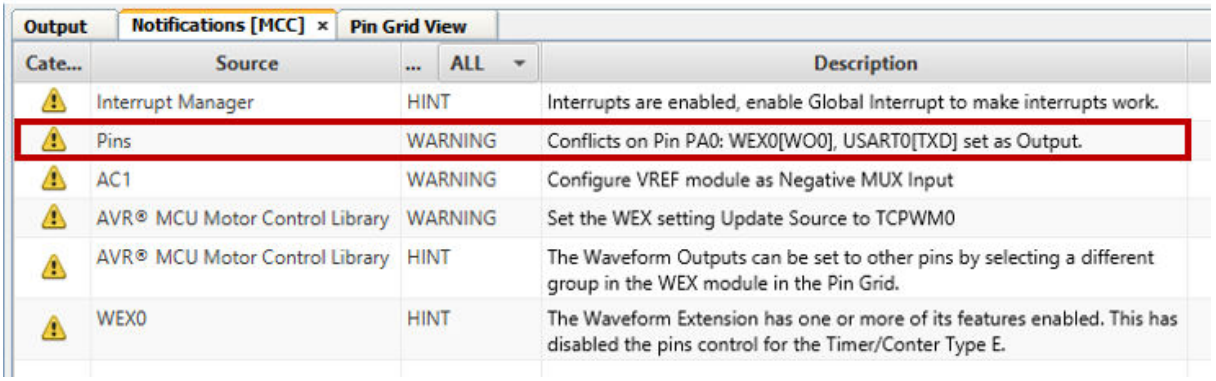
The following steps depict establishing communication with the board in either supported mode using the Multi-Phase Power Board kit.

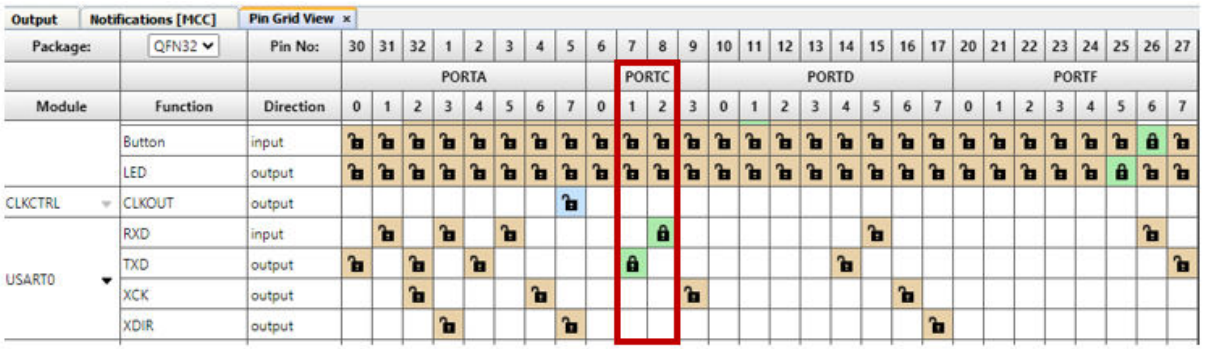
8.1 Communication via Terminal (printf)

Select “printf“ support in the Communication Settings section of the library.

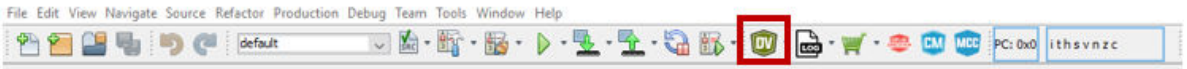


If a warning of pin conflict appears, switch to the **Pin Grid View** tab and select PORTC pin connections for RXD and TXD. If using another setup besides the one described at the beginning of the document, select the desired port pin configuration.

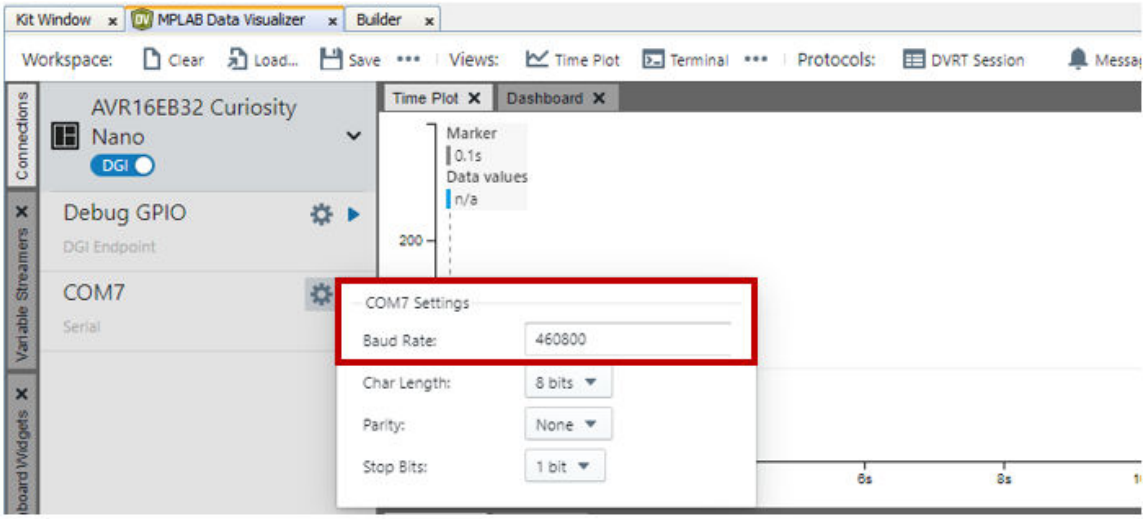




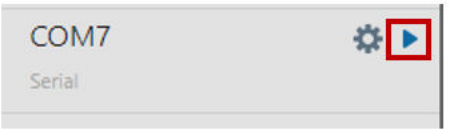
Generate, build the project, and program the device. Open Data Visualizer in the MPLAB toolbar.



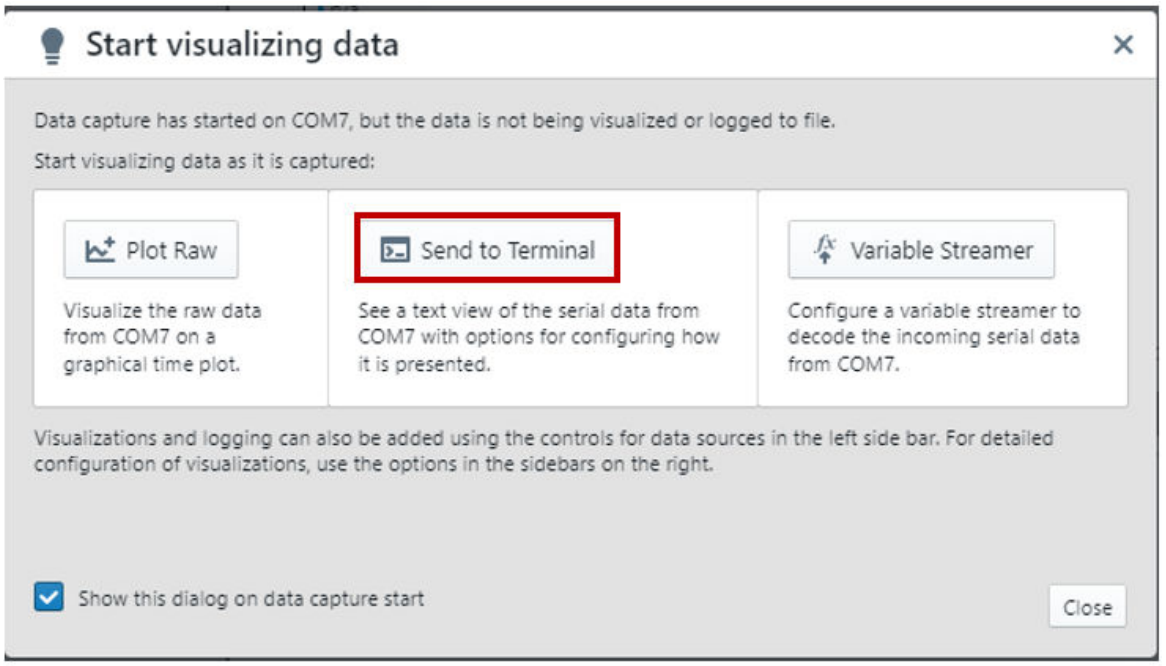
In the MPLAB **Data Visualizer** tab, select the Curiosity Nano COM channel settings and update the baud rate to 460800.



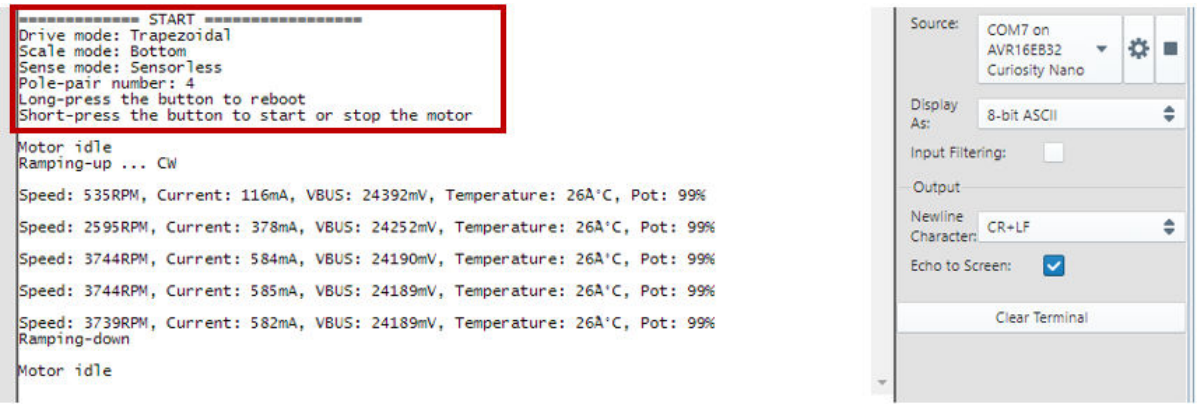
Select the **Connect** button to start the communication.



When prompted, select the Send to Terminal option.



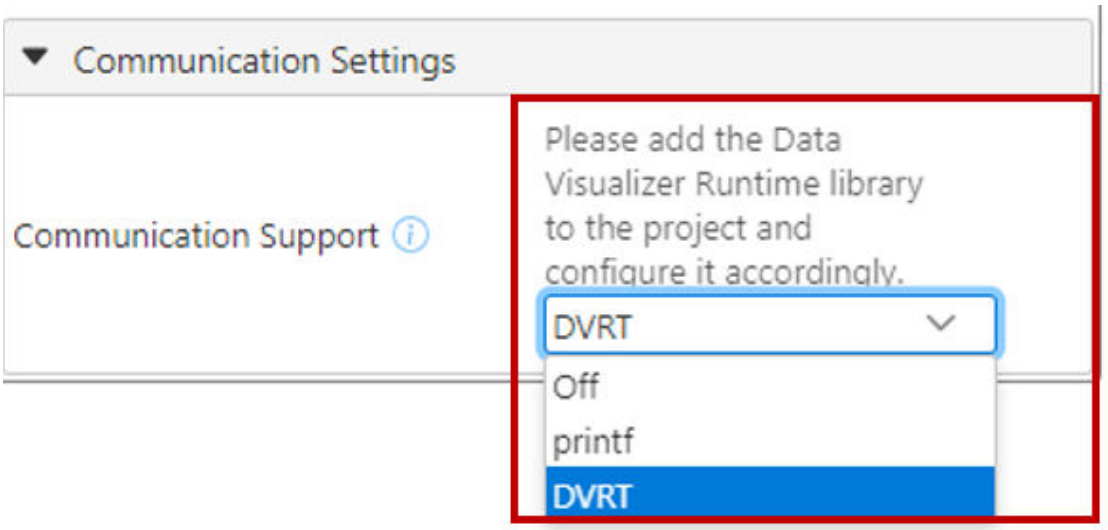
The COM port will change the status to “CAPTURING” when the communication line is active. Long press the **MPPB** button to reboot and see the starting message prompt on the terminal.



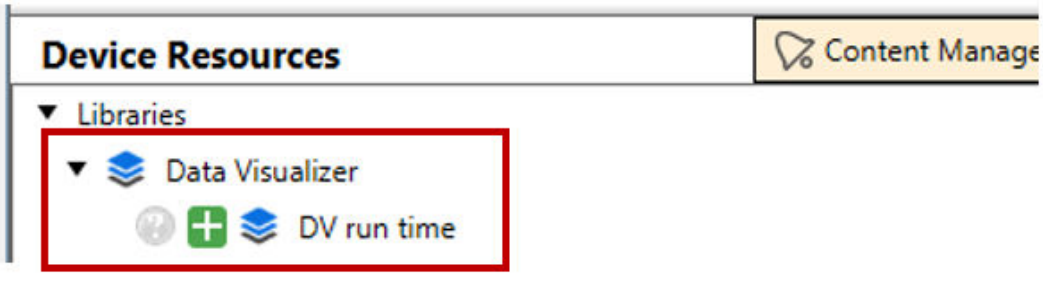
After a short press, the motor is ramping up, and the terminal depicts the Motor Speed in RPM, average current consumption, input voltage, temperature sensor, and the potentiometer value in percentage.

8.2 Communication via DVRT

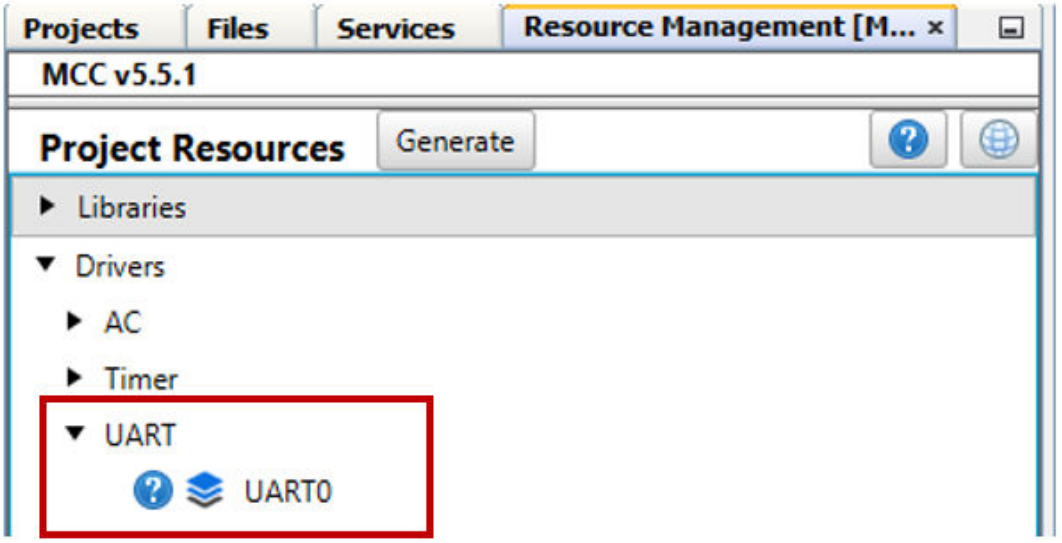
Select the DVRT option in the *Motor Control Library>Communication Support*. Follow the instructions provided above the selection drop-down icon.



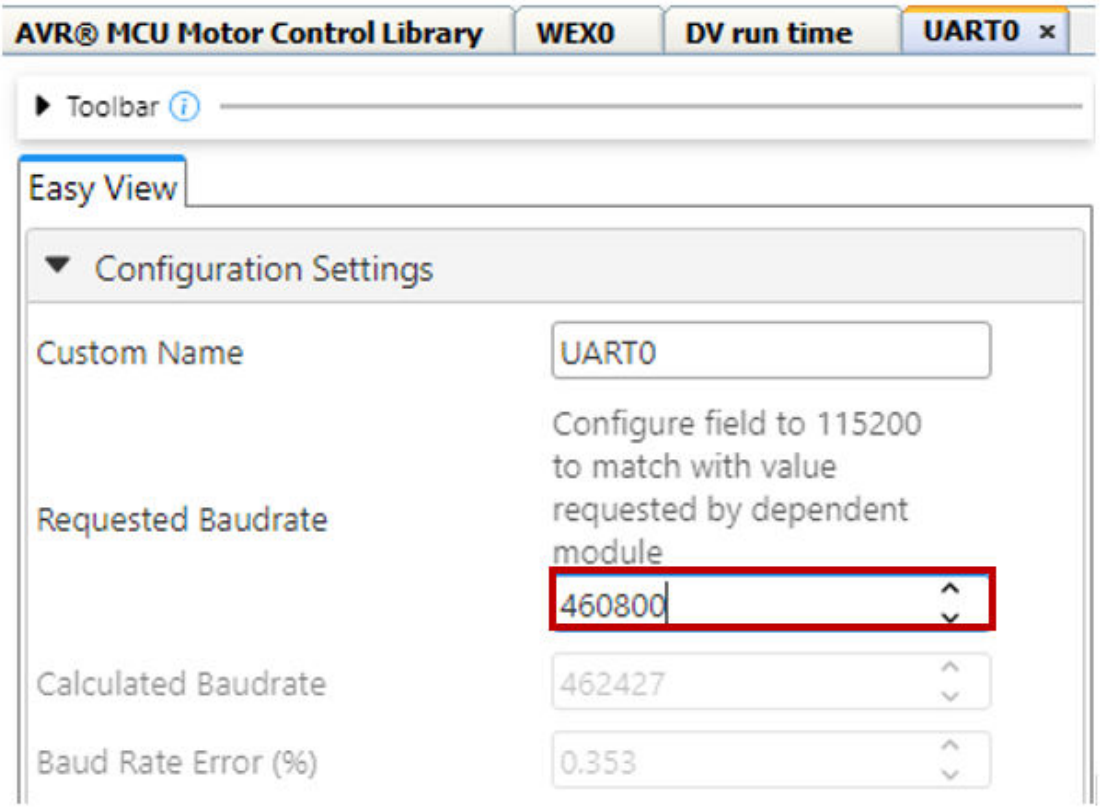
Add DV run-time from *Device Resources>Libraries>Data Visualizer*.



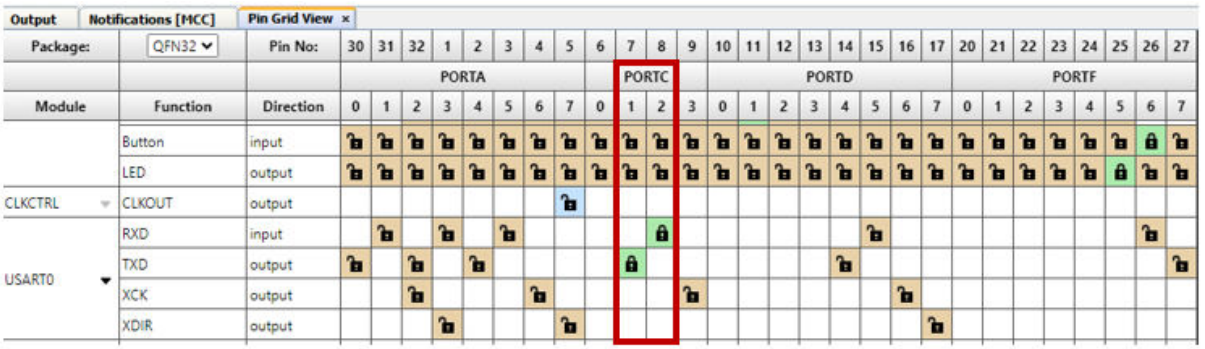
Select UART0 from **Project Resources**.



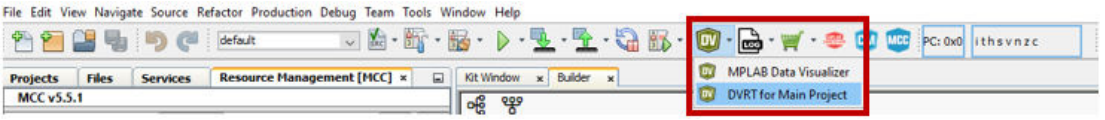
Update the requested baud rate to 460800.



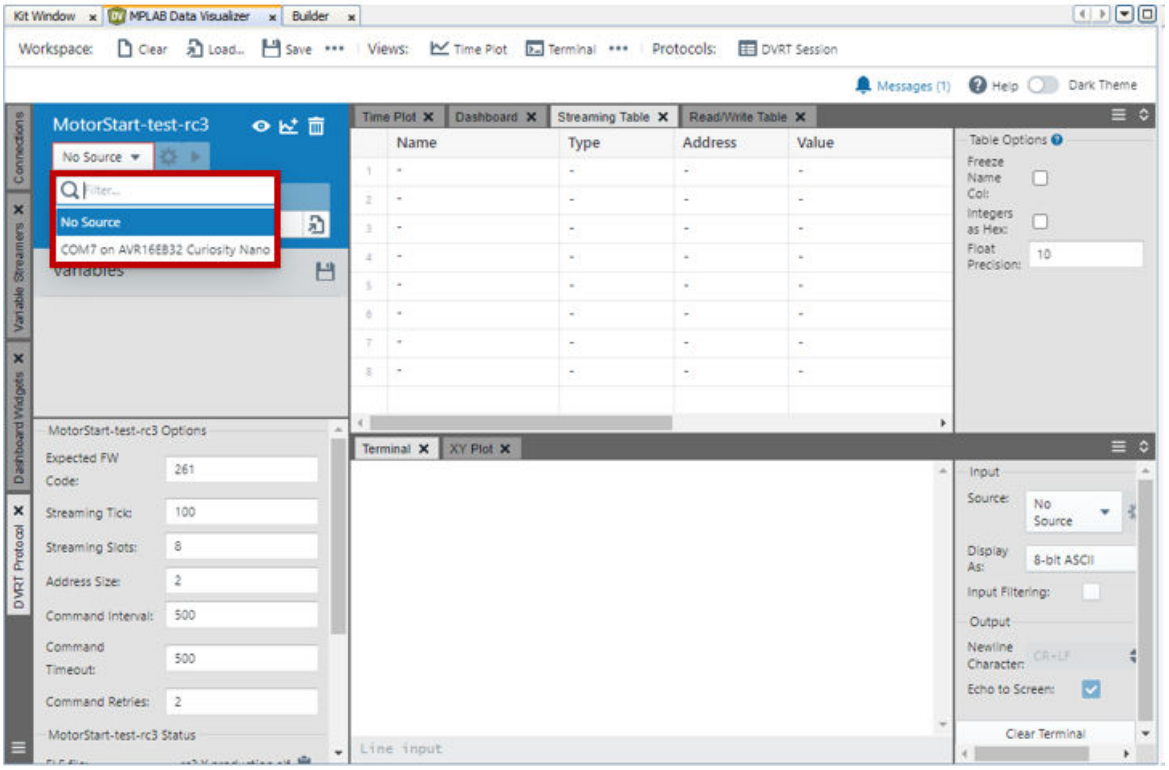
Select PORTC pin connections for RXD and TXD.



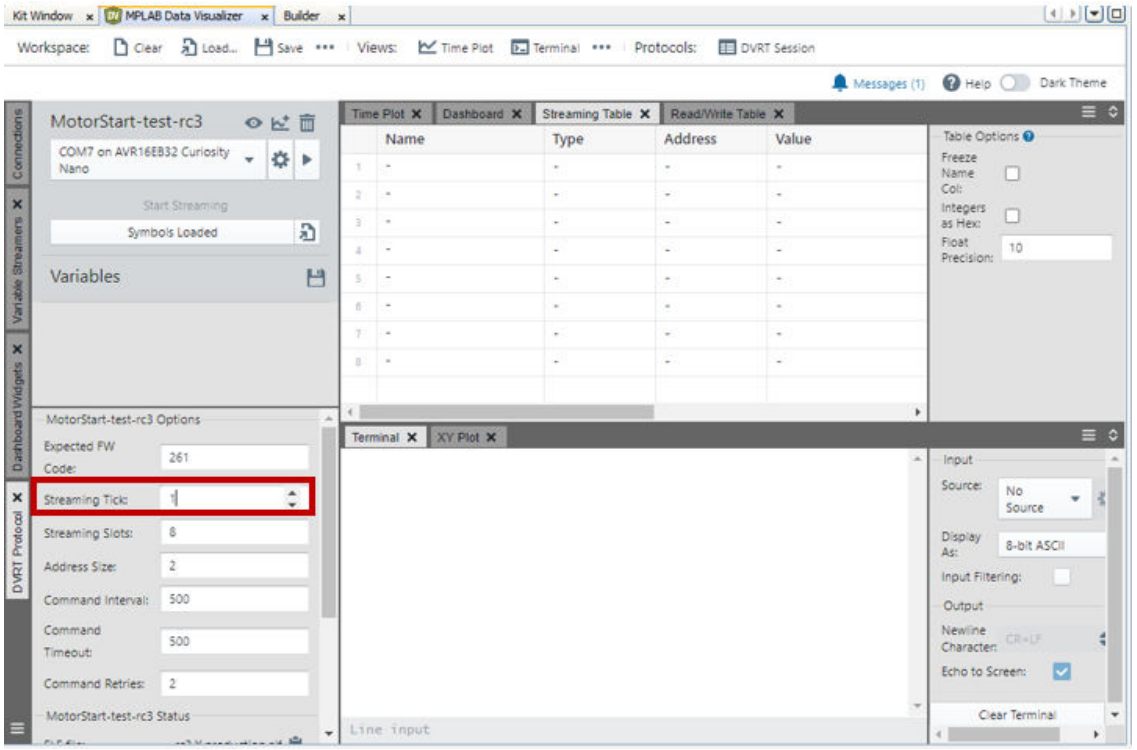
Generate, clean build, program, and open DVRT for the Main Project.



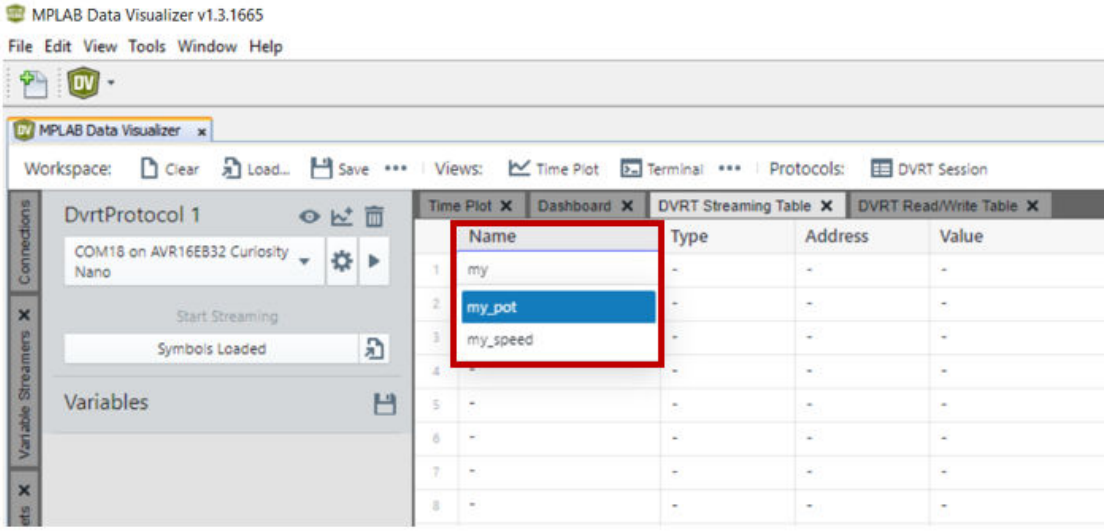
Select the COM"X" for the AVR16EB32 CNANO connection.



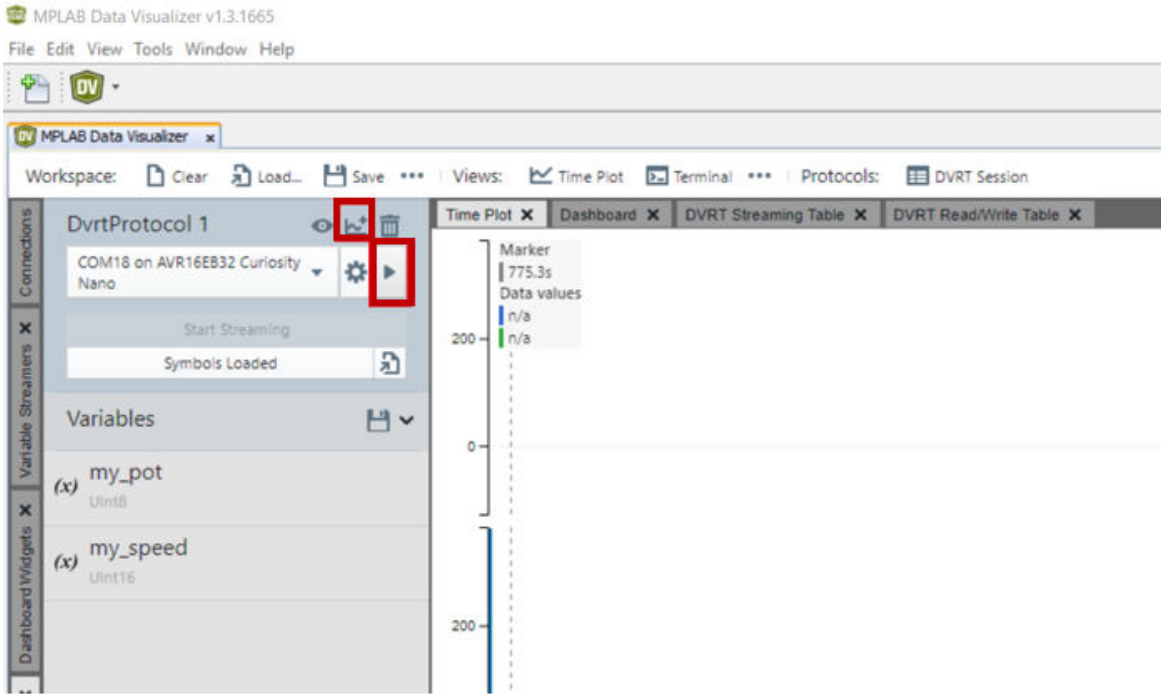
Change Streaming ticks to 1.



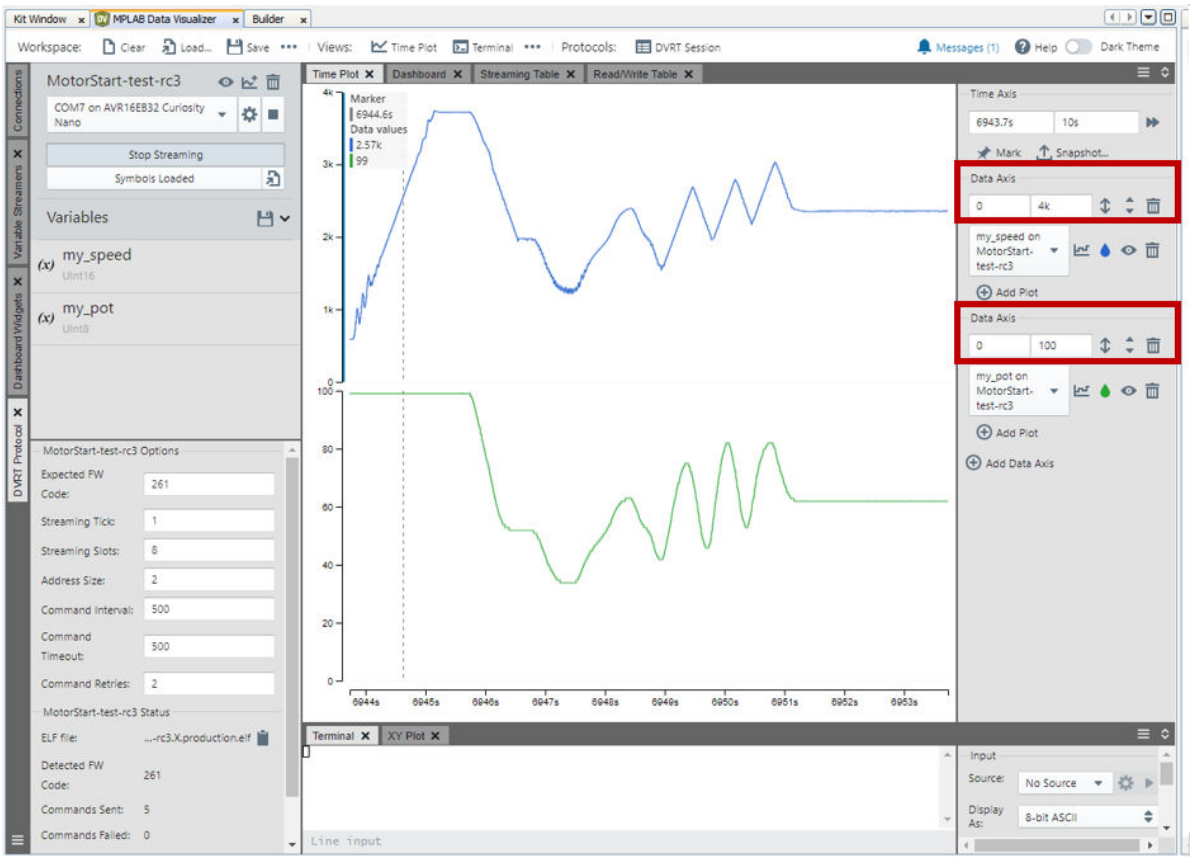
Add the Speed and POT variables.



Click the **Play** button and **Plot** all variables button.



Adjust the data axis. On my_speed set from 0 to the motor nominal RPM speed and on my_pot from 0 to 100. See the results while changing potentiometer values.

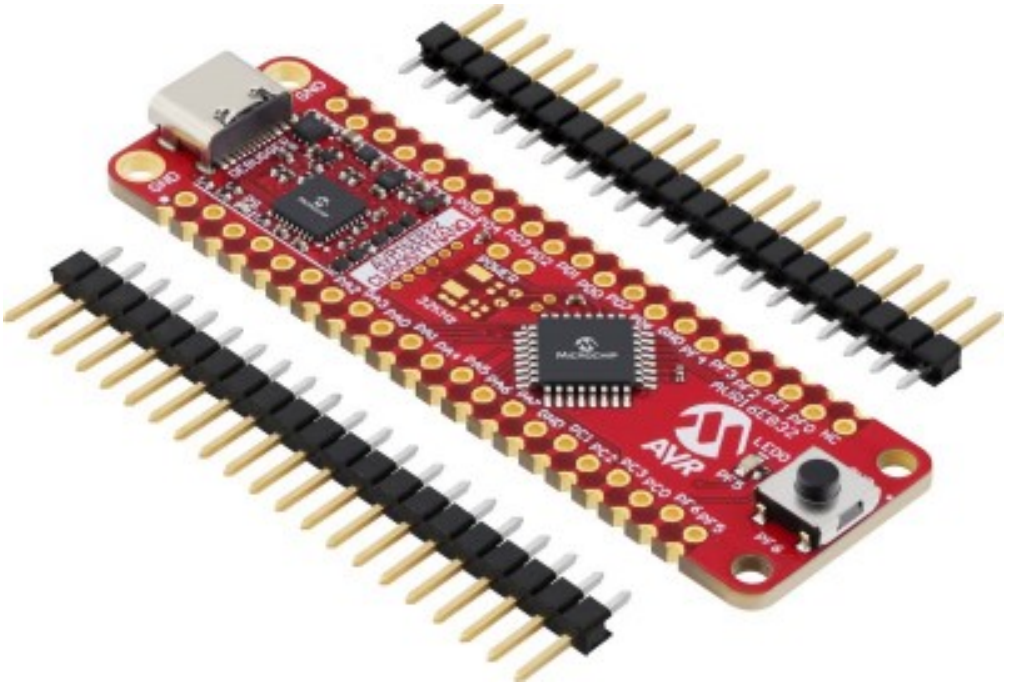


9. Demo

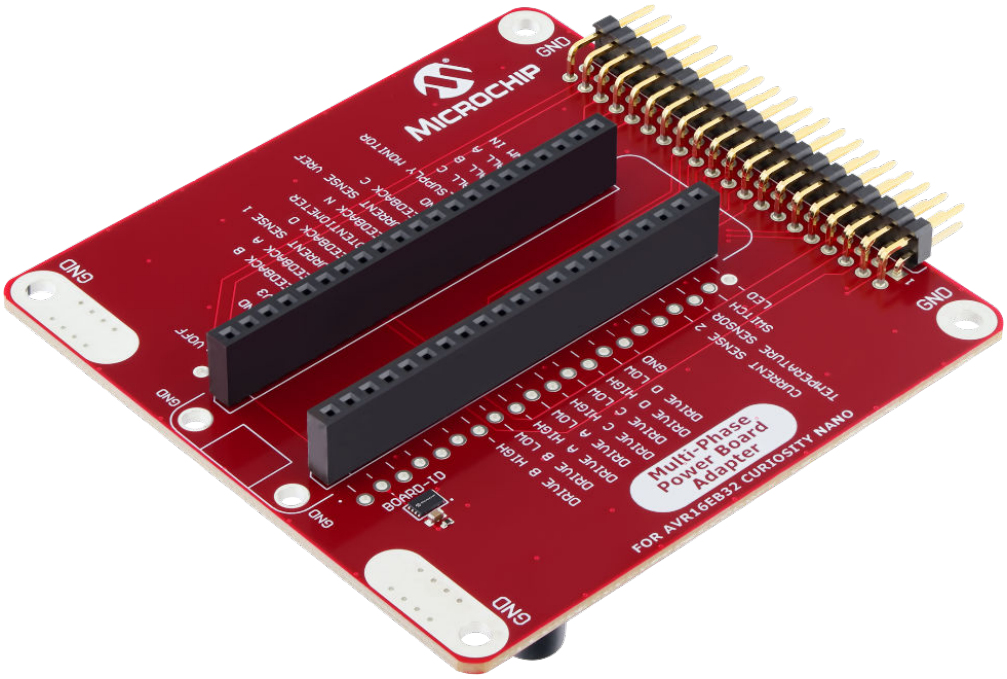
In this section, a complete step-by-step guide to programming the trapezoidal sensorless using the Multi-Phase Power Board kit is described.

Here is the list of hardware needed to recreate the demo:

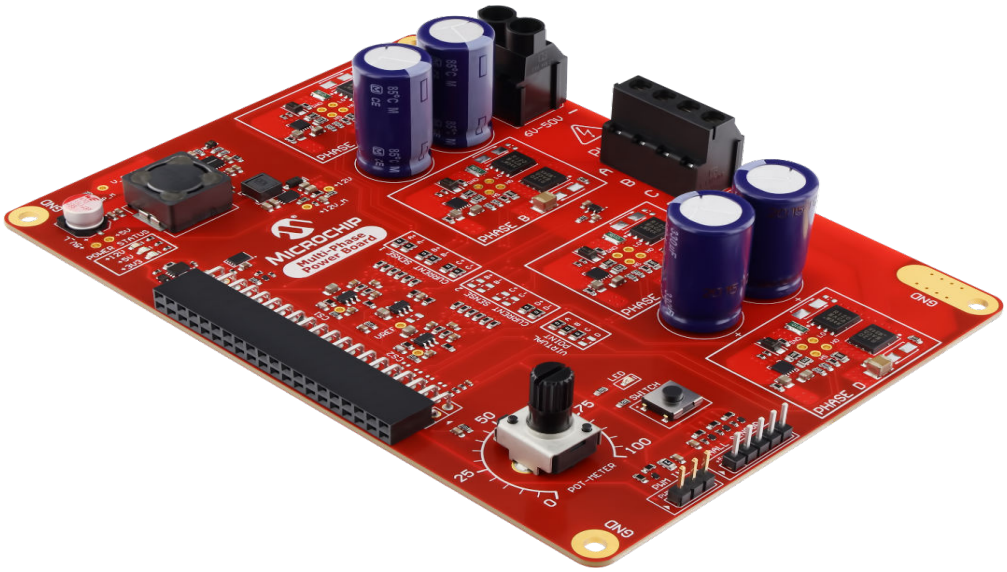
- AVR16EB32 CNANO - EV73J36A
  - AVR16EB32 CNANO
  - 1x USB type-C cable



- AVR16EB32 CNANO to Multi-Phase Power Board adapter – EV88N31A



- Multi-Phase Power Board – EV35Z86A

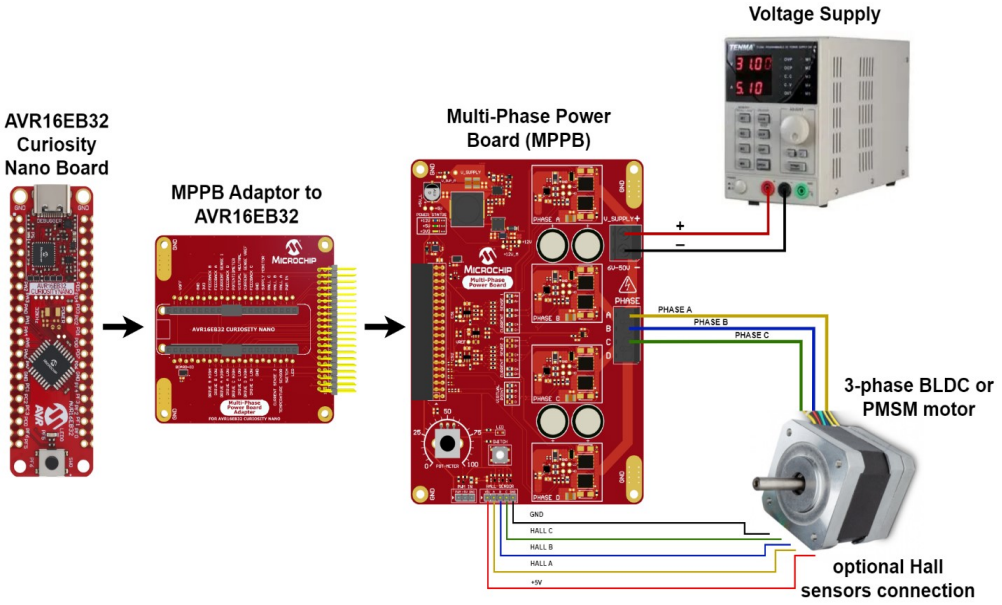


- 3-phase BLDC or PMSM motor
- Universal adjustable Power Supply or at least a 24V/3A fixed power supply

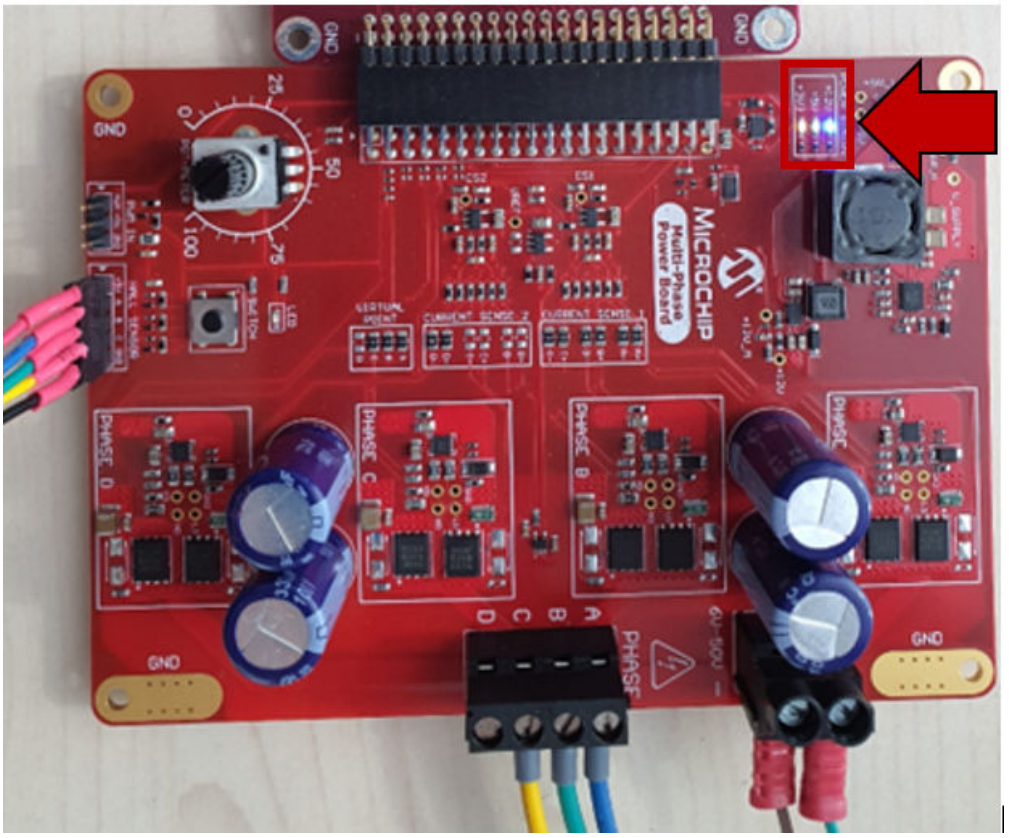
10. Procedure

10.1 Step 1. Hardware Setup

1. Assemble the CNANO board with the MPPB Adapter and the Multi-Phase Power Board.



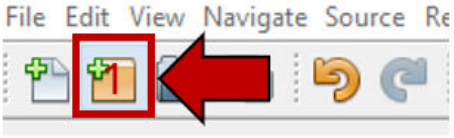
2. Connect the CNANO board to the PC with the help of the USB-C cable.
3. Connect the Motor to the MPPB. When using Hall sensors, ensure the Hall sensor is in the same order as the motor phases, as described in the motor data sheet).
4. Connect the Power Supply and set the voltage to the Motor's nominal voltage (24V). The 12V LED must be ON. When the power is connected correctly and turned on, the MPPB LEDs indicate the needed DC voltage levels.

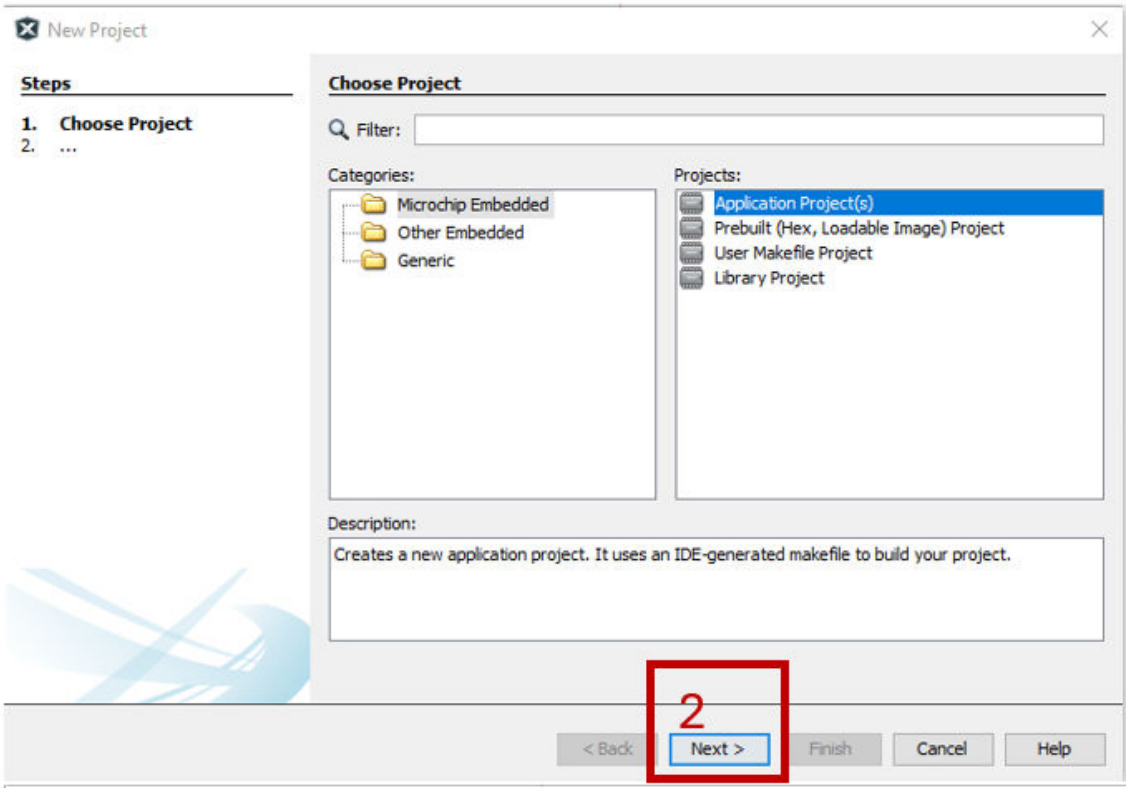


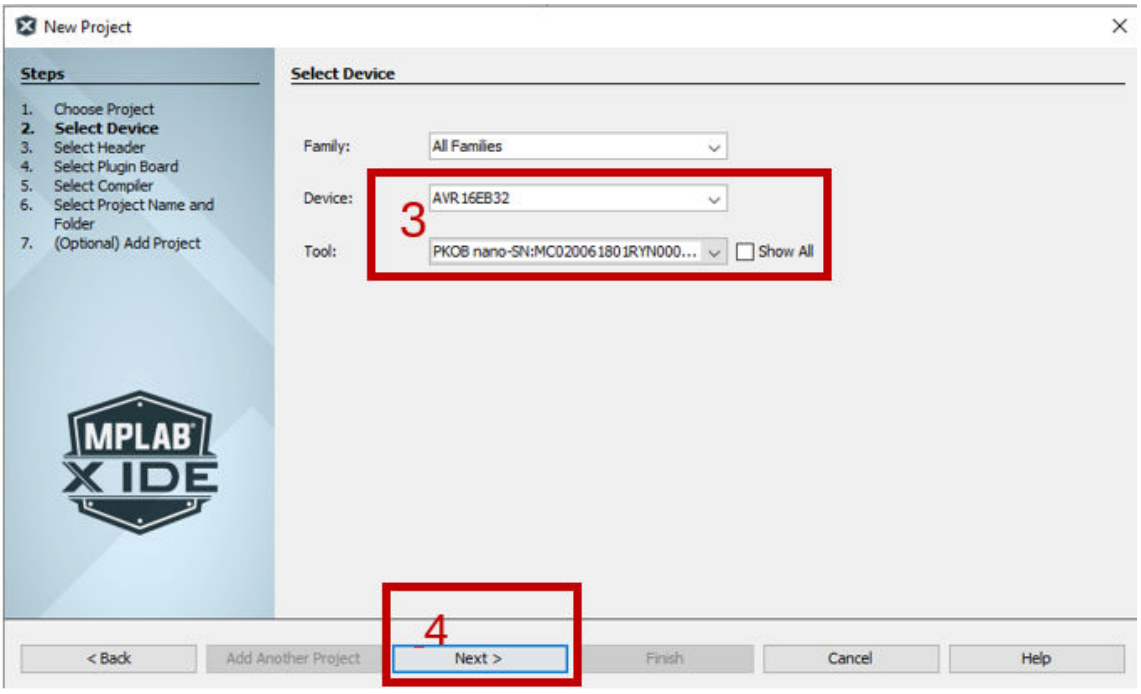
The hardware setup is done.

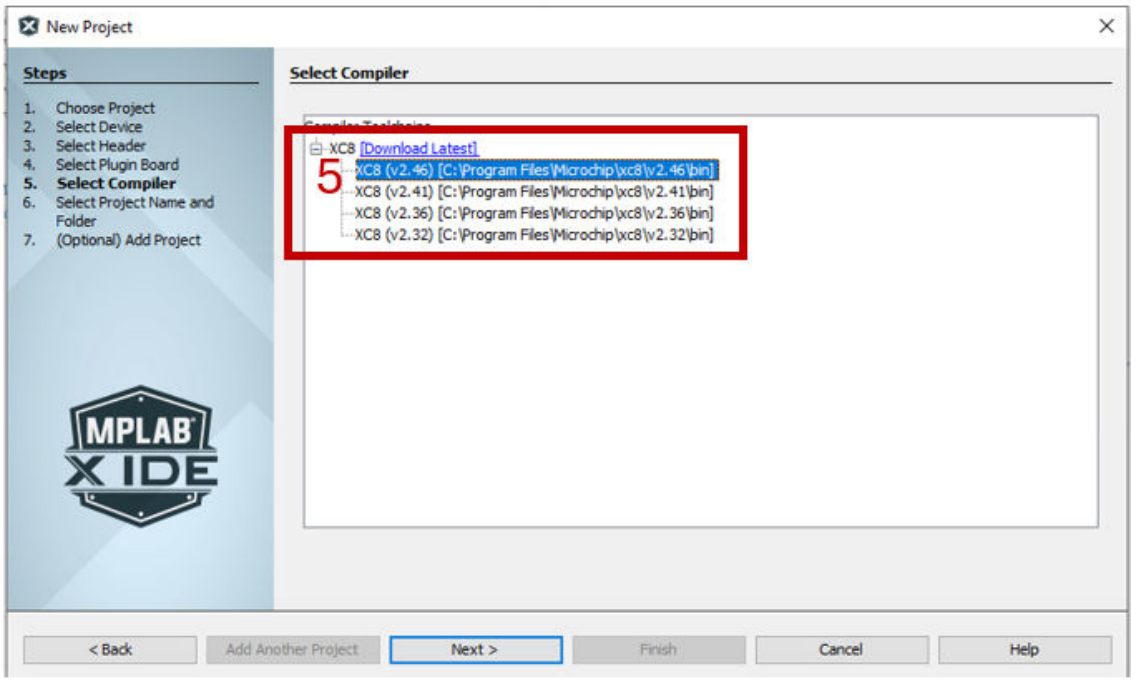
10.2 Step 2. MPLAB® Setup

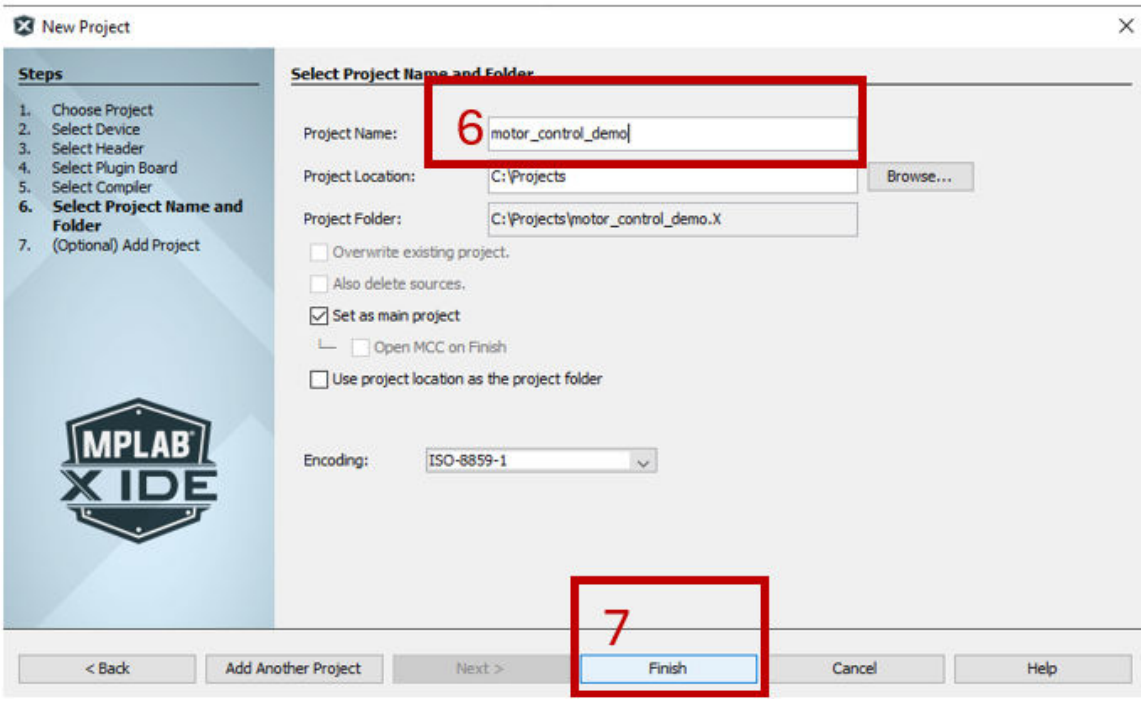
1. Open MPLAB X IDE and create a new project for the AVR16EB32 device.



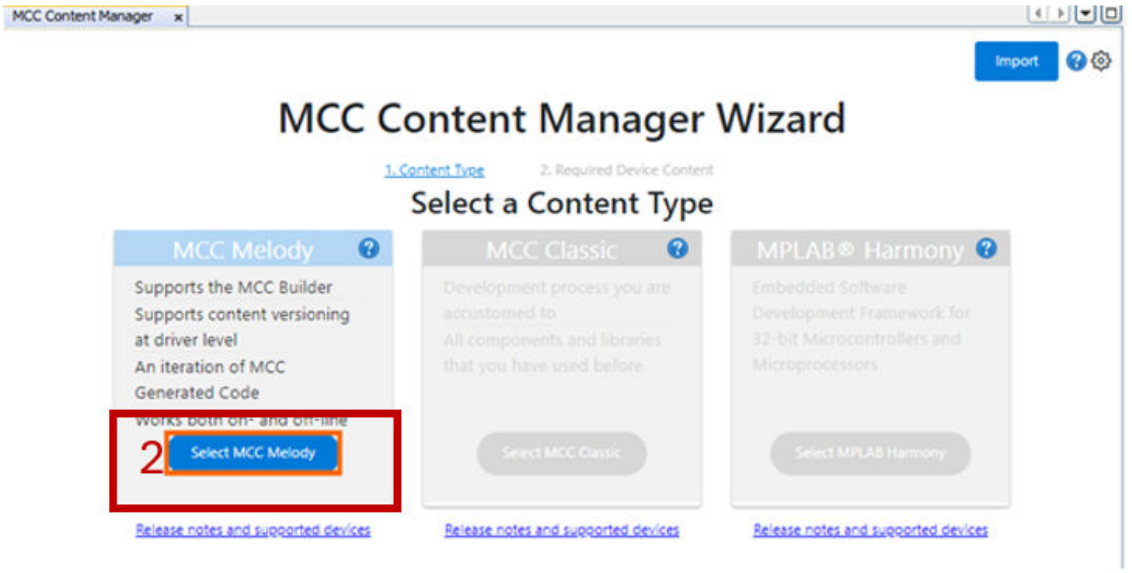


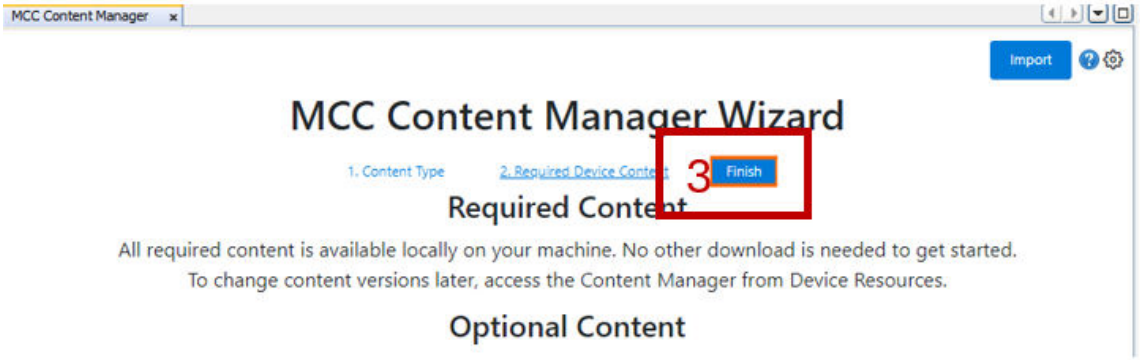




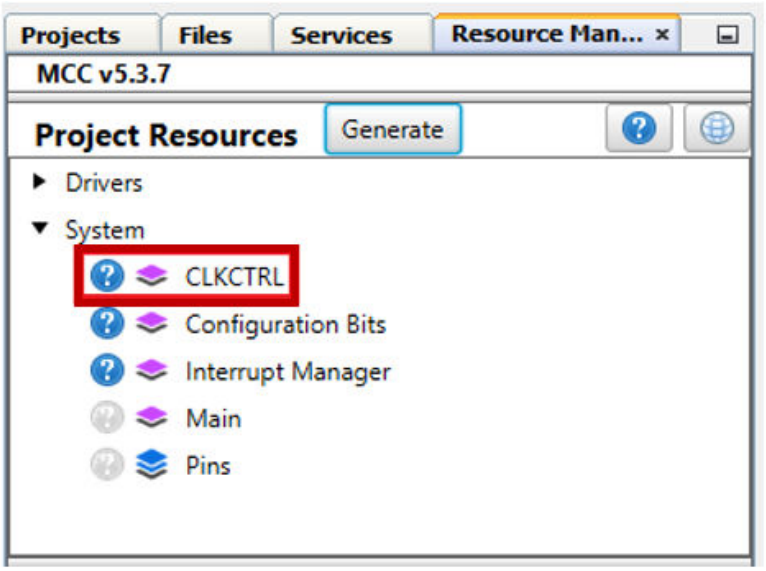


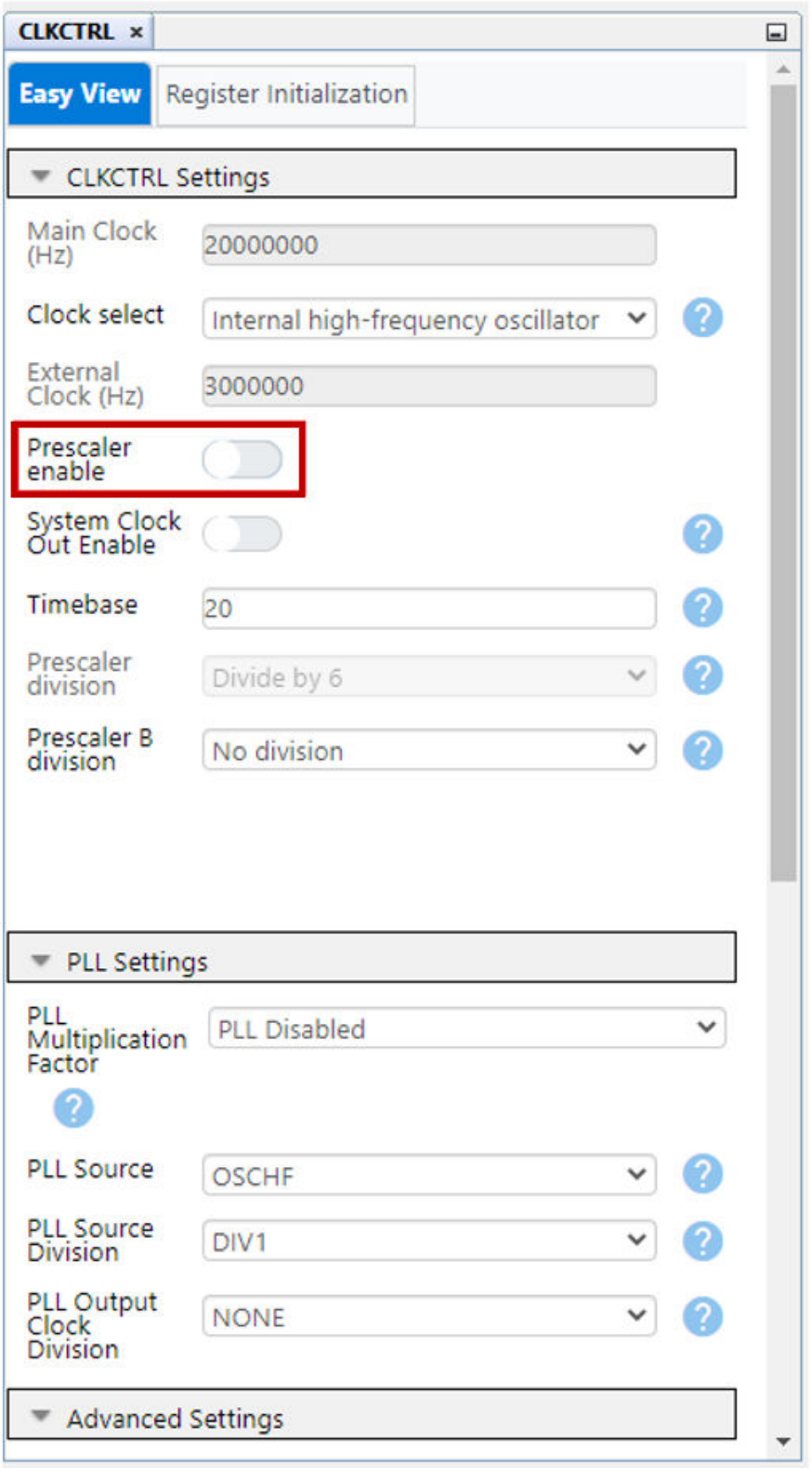
2. Open MCC from the toolbar. Here you will find more information on installing the MCC plug-in.



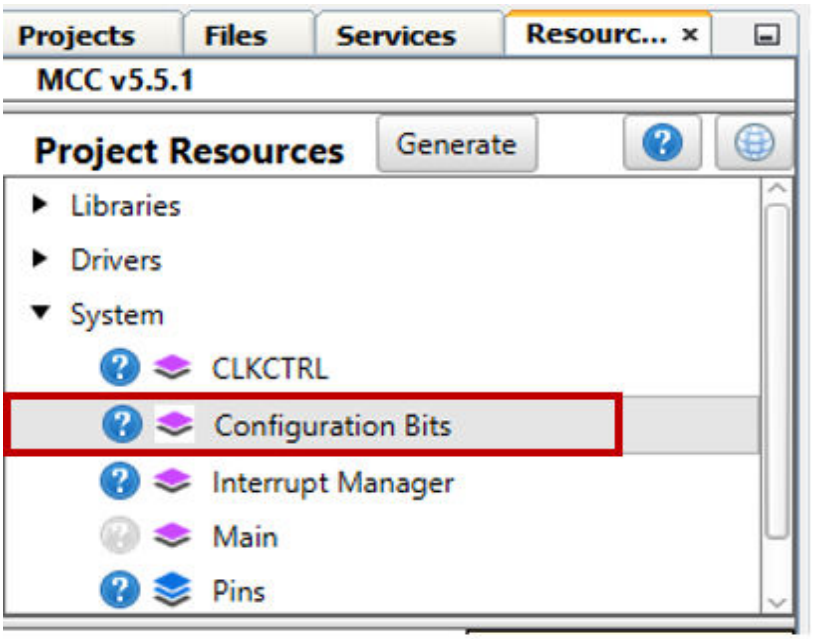


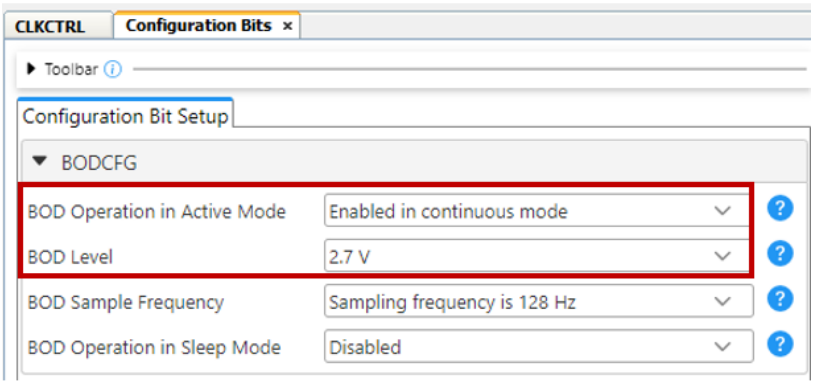
3. In the **Resource Manager** tab, click *Project Resources>System>CLKCTRL*. In the **CLKCTRL** tab, disable the Prescaler enable switch (enabling the 20 MHz core clock).





4. BOD settings. In the System resources, select the Configuration Bits. In the opened tab, change BOD Operation in Active mode to “Enabled in continuous mode” and BOD Level to “2.7V”. This step allows a proper MCU reset when the power is turned off.

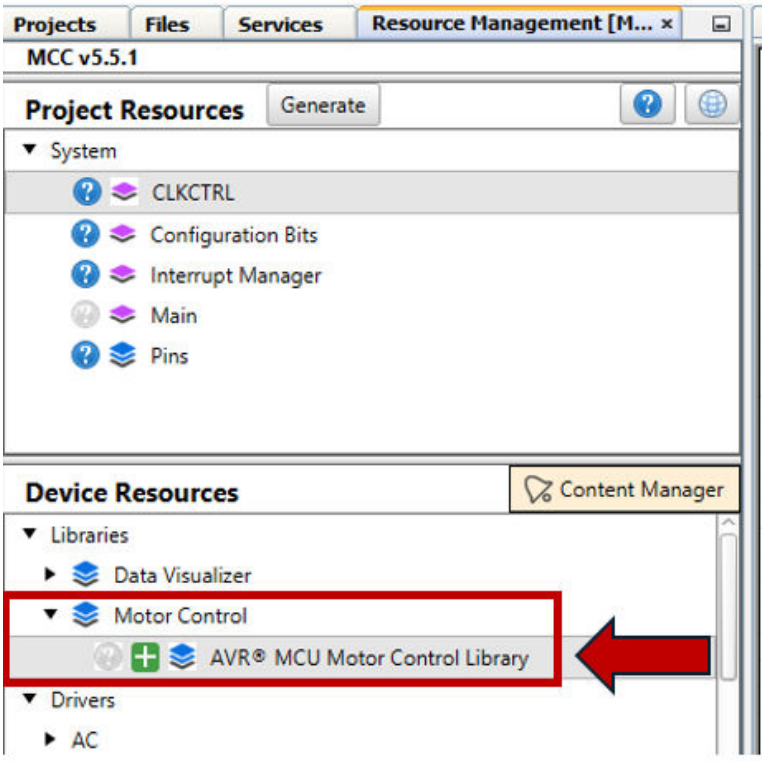




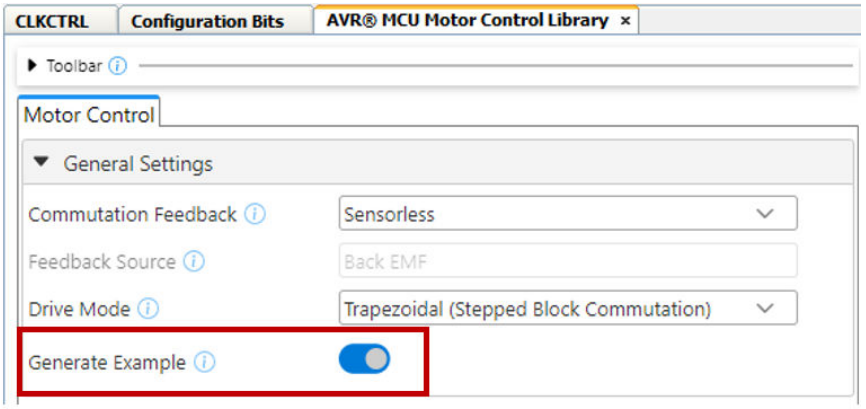
MPLAB setup done with MCU settings set to a 20 MHz clock.

10.3 Step 3. AVR® MCC Motor Control Library Setup

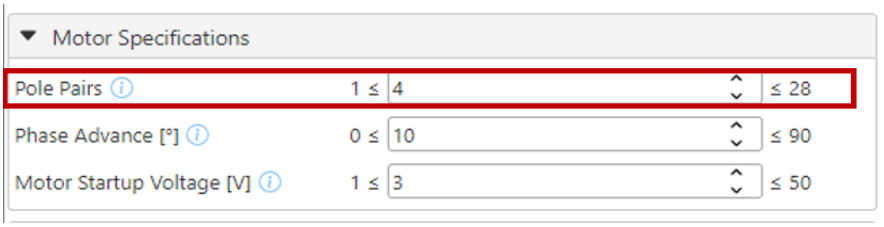
1. To add the Motor Control library, go to *Device Resources>Libraries>Motor Control>AVR MCU Motor Control Library*, and click the green plus button.



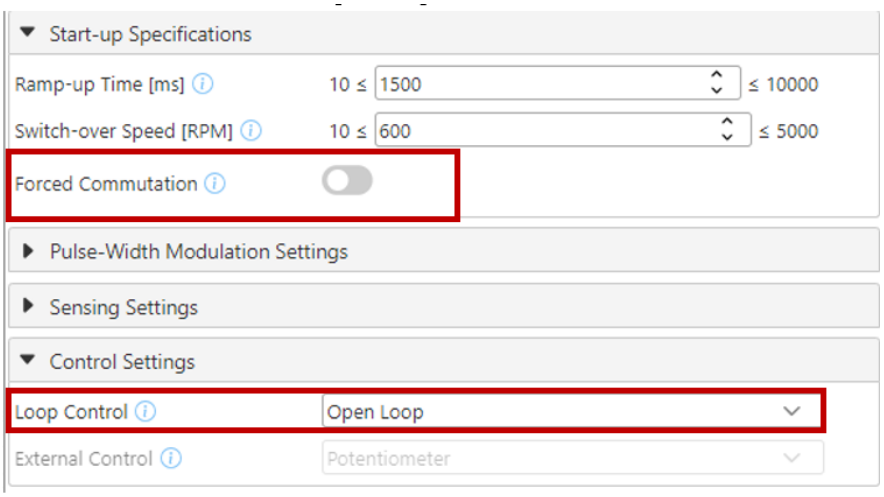
2. In the **Motor Control Library Configuration** tab, enable the Generate Example option to create the out-of-the-box demo layer besides including the motor control layers. This completes the default MPPB-related settings and PIN connections.



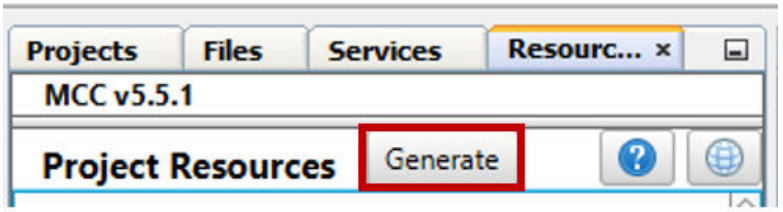
3. Look for the Motor data sheet and find the number of pole pairs and nominal voltage. Set the correct motor pole pair (the demo used an ACT57BLF02 that has eight poles, so a total of four pole pairs). The Phase Advance and Start-up Voltage will remain unchanged.



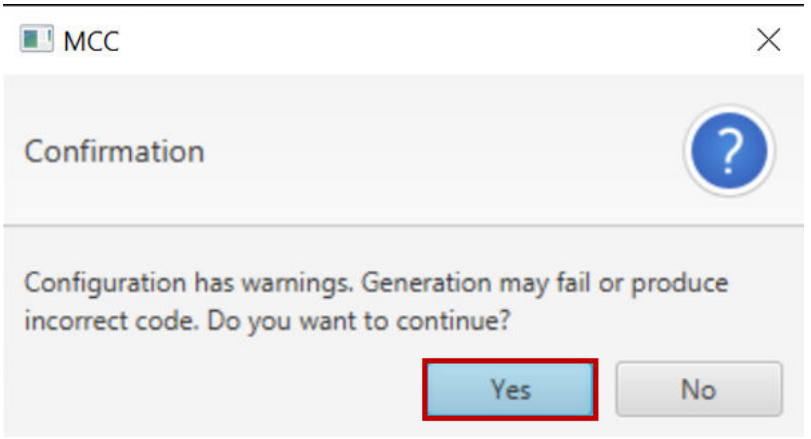
4. Turn off Forced Commutation and select “Open Loop”.



5. Generate the Code demo with APIs and the Motor Control library.

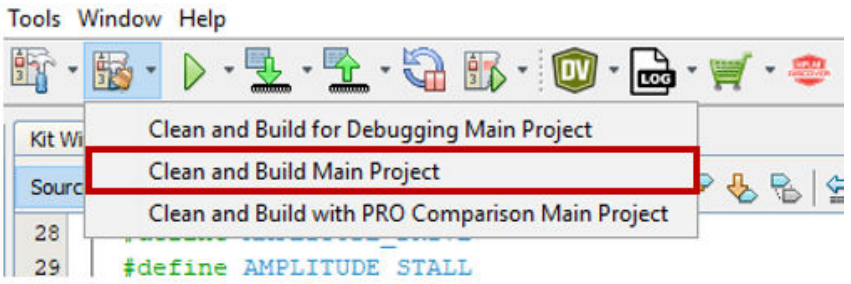


6. Click **Yes** to confirm generation with warnings.

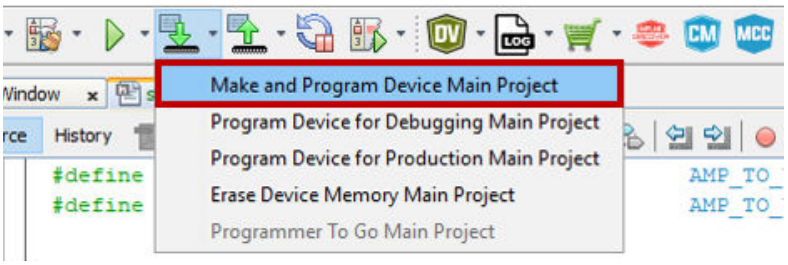


10.4 Step 4. Programming Setup

1. Build the project by clicking **Clean and Build Project**.



2. Click **Make and Program Device** to program the project to the AVR EB CNANO board.



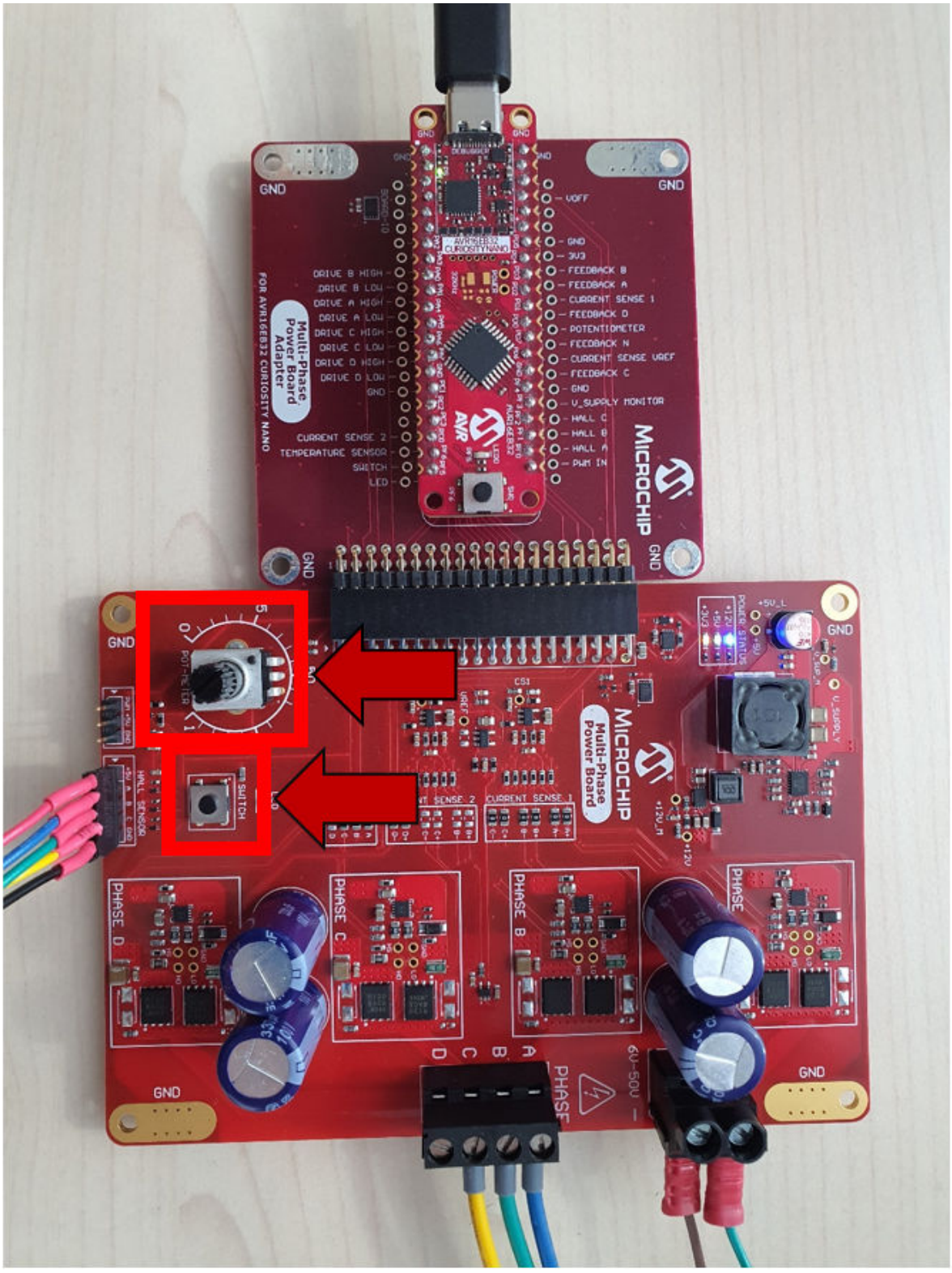
The microcontroller is programmed with the Synchronized Commutation demo.

10.5 Step 5. Demo Instructions

To start the motor drive, follow the instructions:

1. Long-press the button to reboot.
2. Short press the button to start/stop the motor. In Synchronization mode, the amplitude is controlled using the MPPB potentiometer.

3.



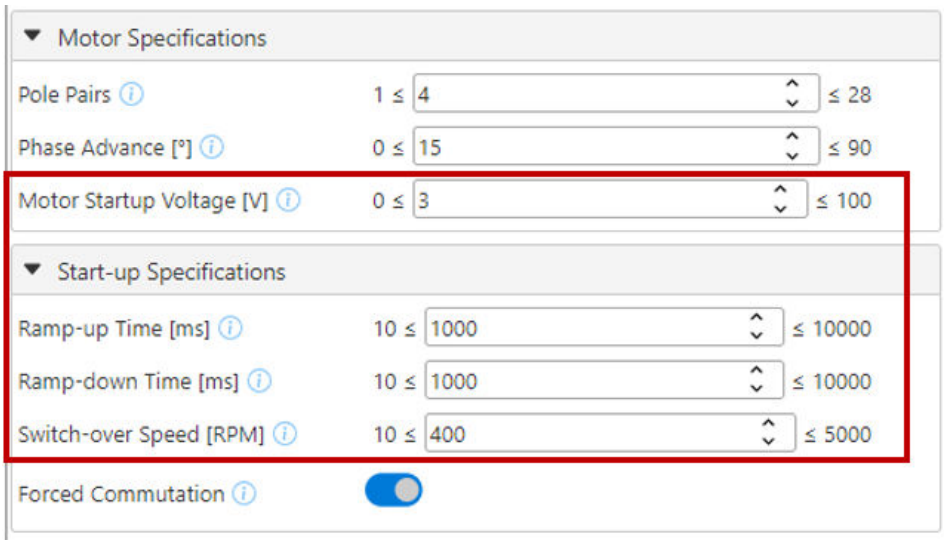
10.6 Step 6. Demo Tuning

If the motor has a hard time at start-up, consider tuning the following values:

Start-up Voltage – Changes the modulated voltage achieved at the end of the ramp-up time

Ramp-up Time – The amount of time it takes to achieve from the offset voltage to the start-up voltage

Switch-over Speed – The electrical RPM speed reached at the end of the ramp up procedure



Rule of thumb for calculating the commutation values:

- V_{BEMF}: Back EMF(V/kRPM) – from the Motor data sheet
- S_{RPM}: X% of Rated Speed – from the Motor data sheet
- Ramp-up Time – The time it takes to achieve from the offset voltage to the start-up voltage
- Switch-over Speed – X% of Rated Speed

This formula is a good starting point. After that, tuning might be necessary to obtain the desired results.

In Open Loop, we ensure drive synchronization with the motor's feedback, and the voltage amplitude (duty cycle) is regulated by the potentiometer.

11. Revision History

Doc. Rev.	Date	Comments
A	02/2025	Initial document release

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0517-8

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.