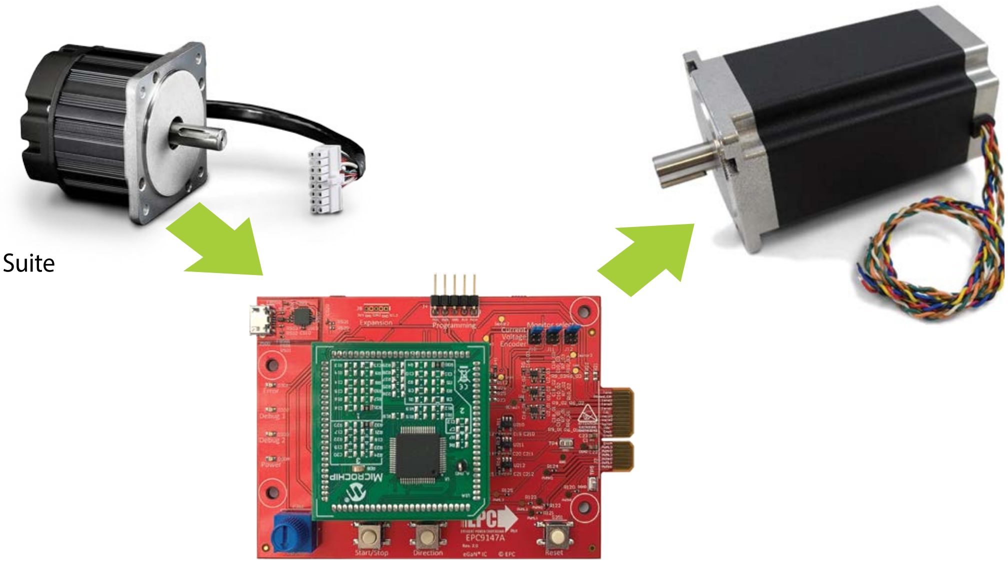
Commissioning a Motor for use with EPC motor drives that operate using Microchip motorBench® Development Suite and EPC9147A-Rev.2.1

Revision 11.0

EFFICIENT POWER CONVERSION

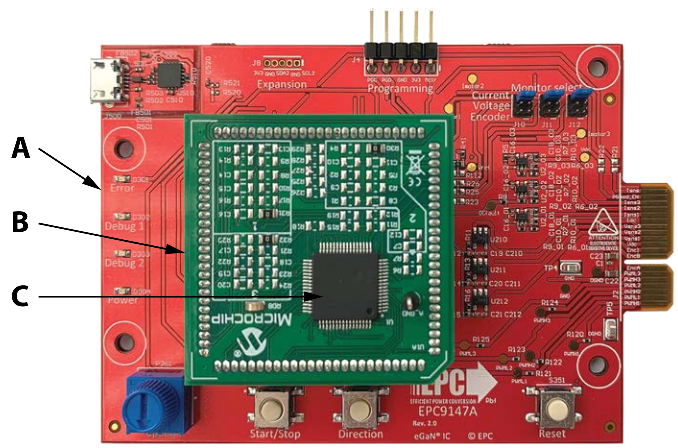
OVERVIEW OF THE PROCESS

- Background
- Equipment needed
- Measuring the motor parameters
- Inputting the motor parameters into Microchip's motorBench® Development Suite
- Generating the control firmware:
  - Compiling
  - Build
  - Flash
- Operating the motor drive system



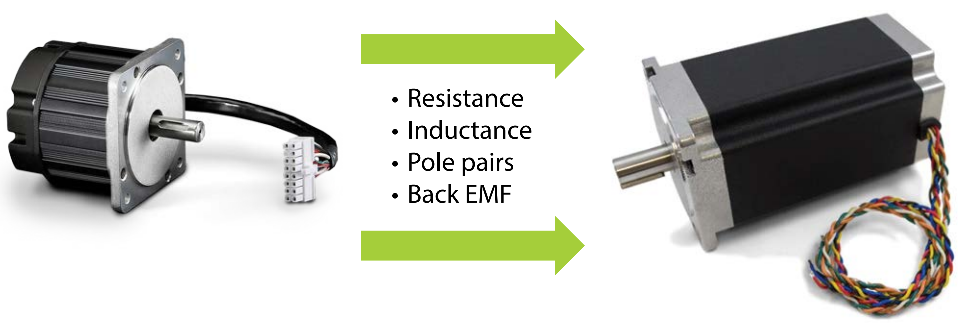
CONTROLLER BOARD BACKGROUND

- Process is for **EPC9147A Only** **(A)**, equipped with MA330031-2 PIM **(B)** with dsPIC33EP256MC506 **(C)** and that uses Microchip®
- motorBench® Development Suite
- EPC9147A (Provided with motor drive KIT's)
  - Pre-programmed with a **sensor-less motor control algorithm** for a **specific motor Teknic_M-3411P-LN-08D** **(D)**



MOTOR CONTROL BACKGROUND

- For sensor-less motor control algorithms:
- Only the three motor terminals connect to the inverter board
- Depends on specific motor parameters (a model of the motor is used for control)
- New motor parameters **must be programmed before** operating a different motor



EQUIPMENT NEEDS, MOTOR ACCESS

Motor Access

- Direct access to the motor terminals
  - Motor terminal must be disconnected from inverter board
- Direct access to the motor shaft
  - Need to turn it by hand

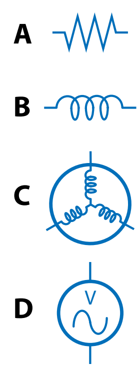
Equipment

- LCR meter
  - To measure line-to-line resistance and inductance
- Oscilloscope
  - To measure line-to-line Back EMF (BEMF)

MEASURING THE MOTOR PARAMETERS

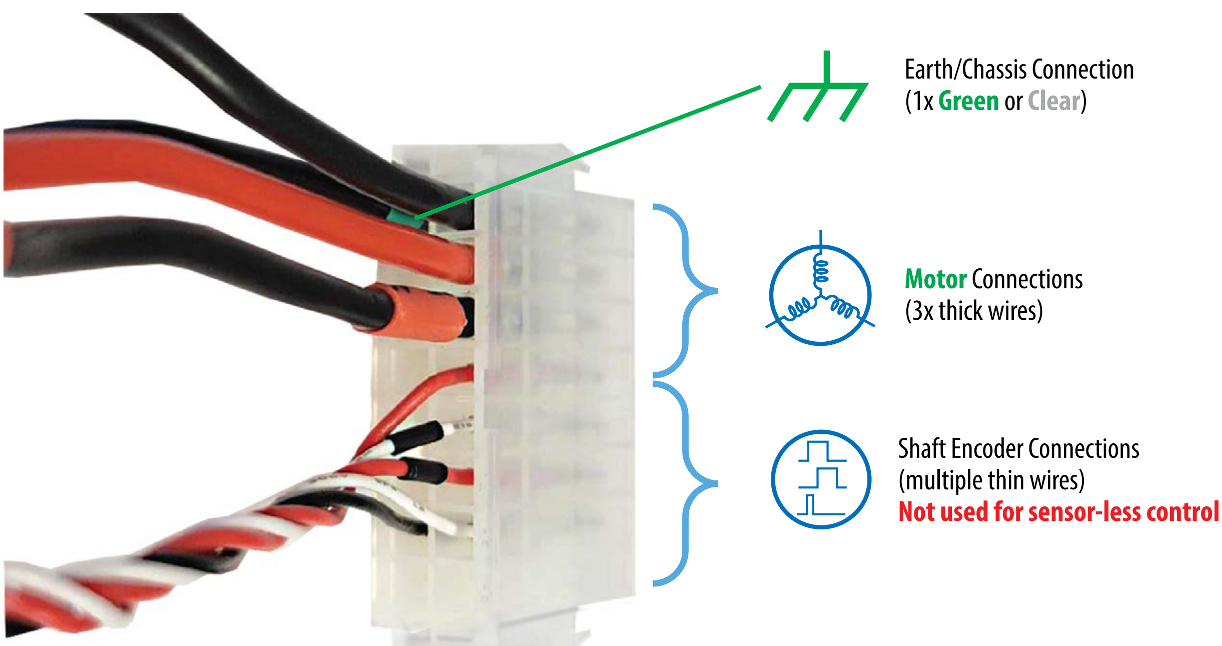
Motor Parameters Needed

- Terminal resistance **(A)**
  - Line-to-line
- Terminal inductance **(B)**
- – Line-to-line
- Pole pairs **(C)**
- Back EMF constant **(D)**



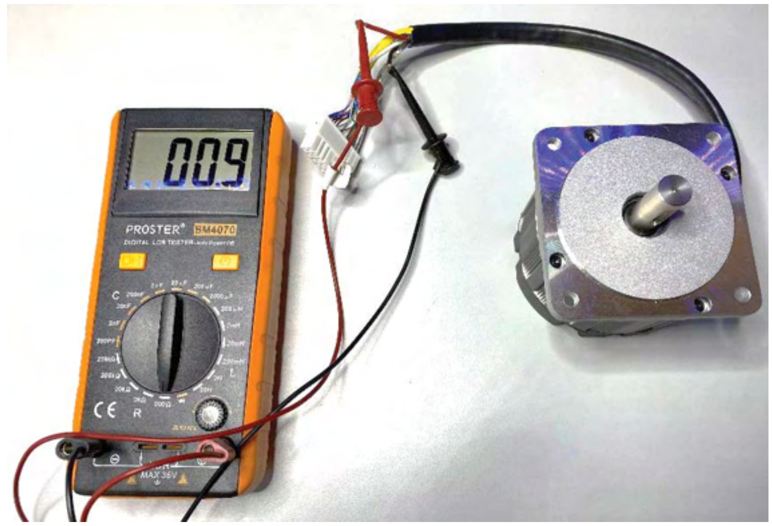
Identification of Motor Terminals

Example for Teknic Model M-3411P-LN-08D



Line-to-Line Resistance Measurement

1. Disconnect all three motor terminals from inverter
2. Connect **only two motor** terminals to an ohm-meter, third terminal is left floating
3. Measure the **line-to-line resistance**
4. **4-wire** resistance measurement is more accurate (if available)



This motor has **R_{L-L} = 800 mΩ** line to line resistance
(100 mΩ due to LCR meter leads)

Line-to-Line Inductance Measurement

1. Disconnect all three motor terminals from inverter
2. Connect **only two motor** terminals to the LCR-meter, third terminal is left floating
3. Measure the **line-to-line inductance**
4. **Note –** long leads will add inductance. Twisting the leads will help reduce inductance. More important for low inductance motors.
5. For motors with **varying inductance with shaft angle**, find the minimum and the maximum inductance values, by measuring at different angles.
6. Determine the average inductance:

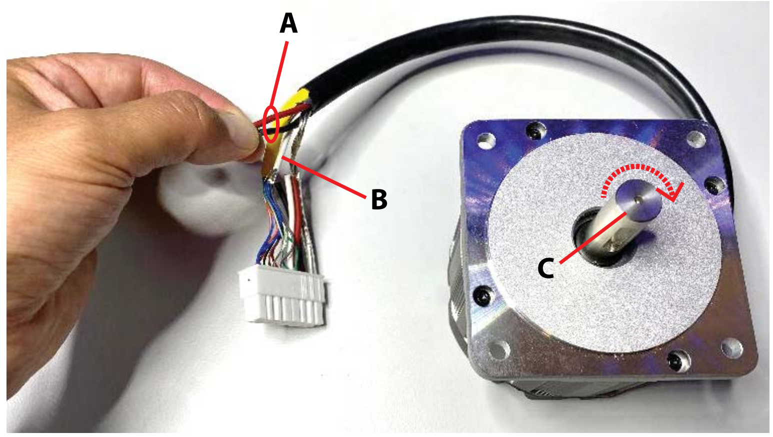
$$L_{avg} = \frac{L_{min} + L_{max}}{2}$$

7. For the example: Rounded 932 µH to 1 mH.
8. Use the same value for L_d and L_q

This motor has **L_{L-L} = 932 µH** line to line inductance
(LCR meter leads may also have inductance, **use autozero function if available**)

Determination of the Pole Pairs Number

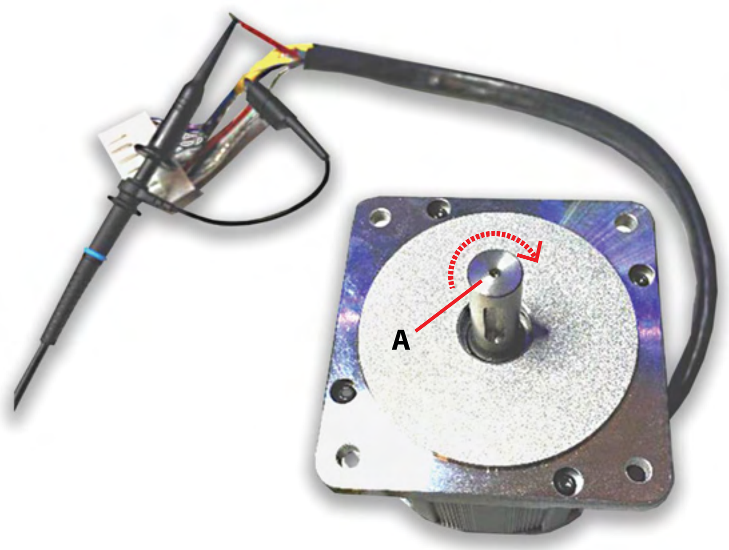
1. Disconnect all three motor terminals from inverter
2. Short **any two (A) motor** terminals, third terminal is left floating **(B)**
3. ***Gently*** and ***slowly*** hand spin the motor shaft **(C)** and make **one mechanical turn only**
   - Count the notches/steps/jumps that you feel with as the motor axle is rotated = motor poles number
4. Divide the **motor poles number** by 2 = **Pole Pairs number (pp)**

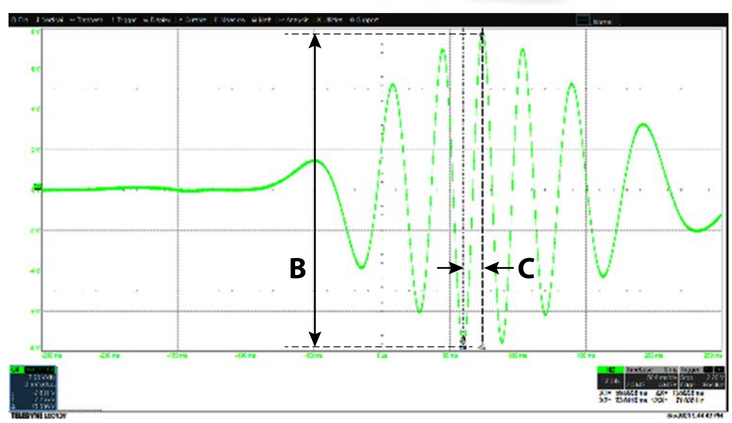


This motor has **pp= 4 pole pairs**

Line-to-line BEMF constant Measurement

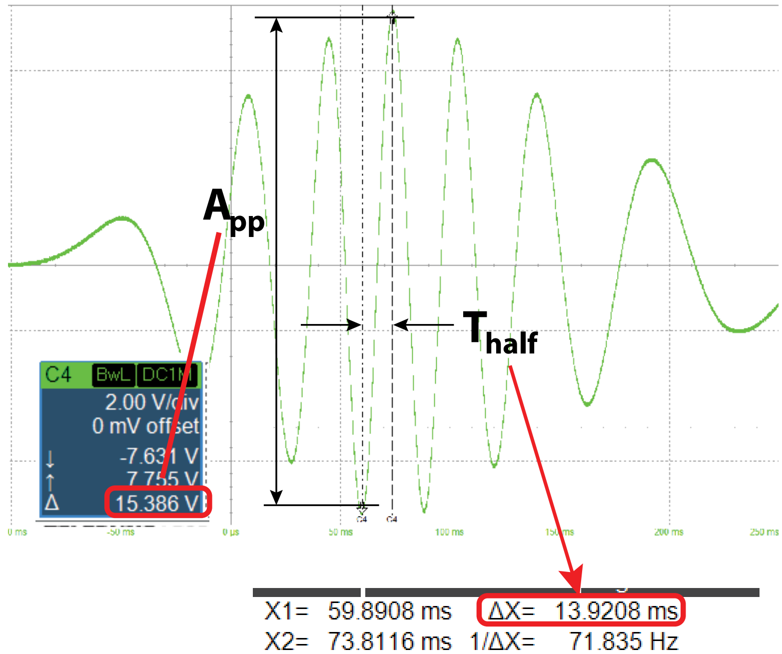
1. Disconnect all three motor terminals from inverter
2. Connect **one** of the **motor** terminals to an oscilloscope probe **ground** lead and the **other motor** terminal to the **tip**. The third motor terminal is left floating
3. Hand **spin** the motor shaft **(A)** and record the voltage signal on the oscilloscope.
4. **(B)** Measure the **peak-to-peak** voltage of **one-half sinusoid** (details on next slide)
5. **(C)** Measure the time period between the **same two peaks** (details next slide)





Line-to-line BEMF Constant Calculation

- $\mathbf{A_{pp}}$ = Half-sinusoid peak-to-peak voltage amplitude (A_{pp} = 15.836 V_{pp})
- $\mathbf{T_{half}}$ = Half sinusoid peak-to-peak period (T_{half} = 13.92 ms)
- **pp** = Pole Pairs (pp = 4)
- Calculate BEMF (for 1 krpm):
  - Units: A_{pp} [V], T_{half} [s]
  - $K_e = \frac{A_{pp}}{2 \cdot \sqrt{2}} \cdot \frac{1000 \cdot pp}{60} \cdot (2 \cdot T_{half})$
  - $K_e = 11.785\ pp \cdot A_{pp} \cdot T_{half}$ **Use this**
  - $\mathbf{K_e}$ **= 10.096** Vrms/krpm for example motor (will use 10.2 in motorBench)

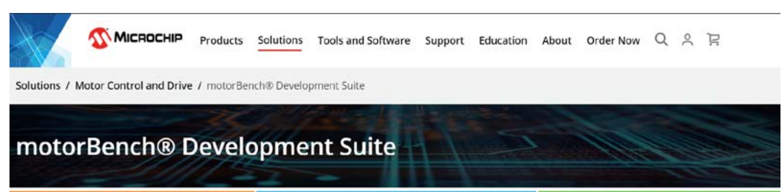


INSTALLING MICROCHIP'S motorBench® DEVELOPMENT SUITE AND INPUTTING THE MOTOR DRIVE AND MOTOR PARAMETERS INTO A PROJECT

Install motorBench® development suite

Refer to Microchip website to install following software, follow exactly the steps indicated in Microchip website

1. MPLAB X IDE (i.e. **5.45 version**), make sure to install the recommended updates **(A)**.
2. Microchip code configurator plugin **(B)**
3. Microchip motorBench plugin **2.35** **(C)**
4. MCLV-2 project to start (or EPC project for EPC914xKIT) called **sample-mb-33ep256mc506-mclv2.X**



A https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide#

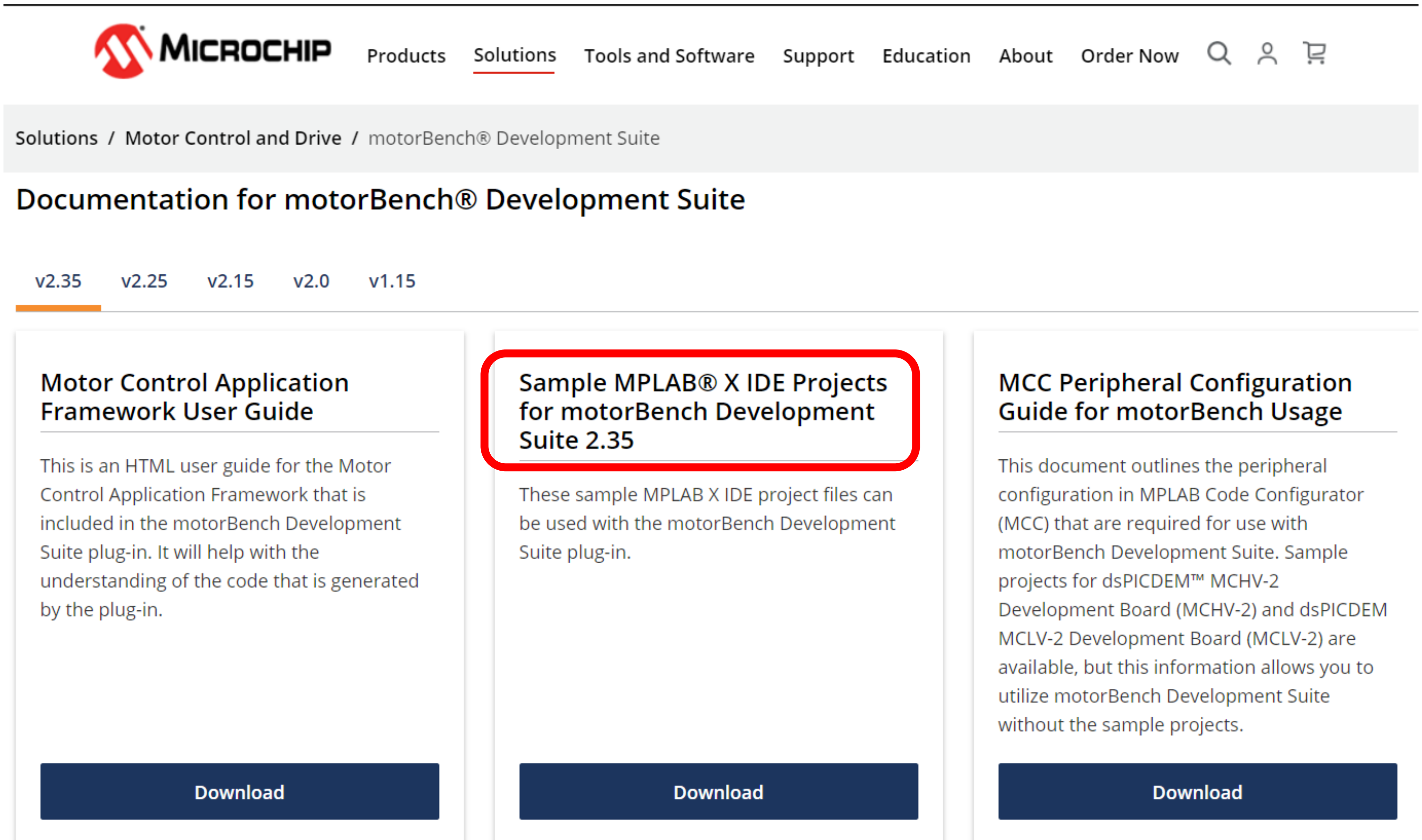
B https://www.microchip.com/en-us/development-tools-tools-and-software/embedded-software-center/mplab-code-configurator#Downloads

C https://www.microchip.com/en-us/solutions/motor-control-and-drive/motorbench-development-suite

Download Sample Project

Refer to Microchip website to install following motorBench sample project

- Sample MPLAB® X IDE Projects for motorBench Development Suite 2.35

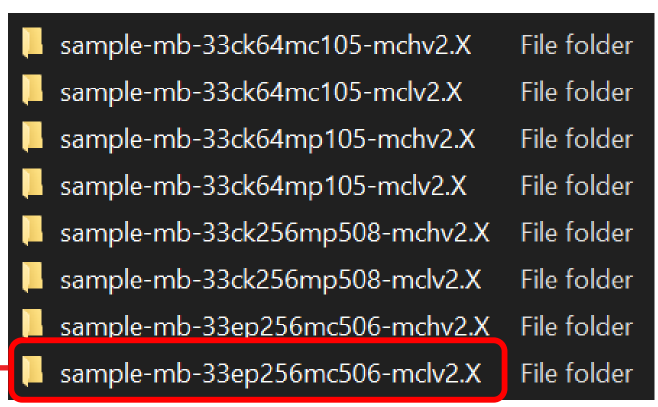


Select Sample Project

Unzip the Sample projects

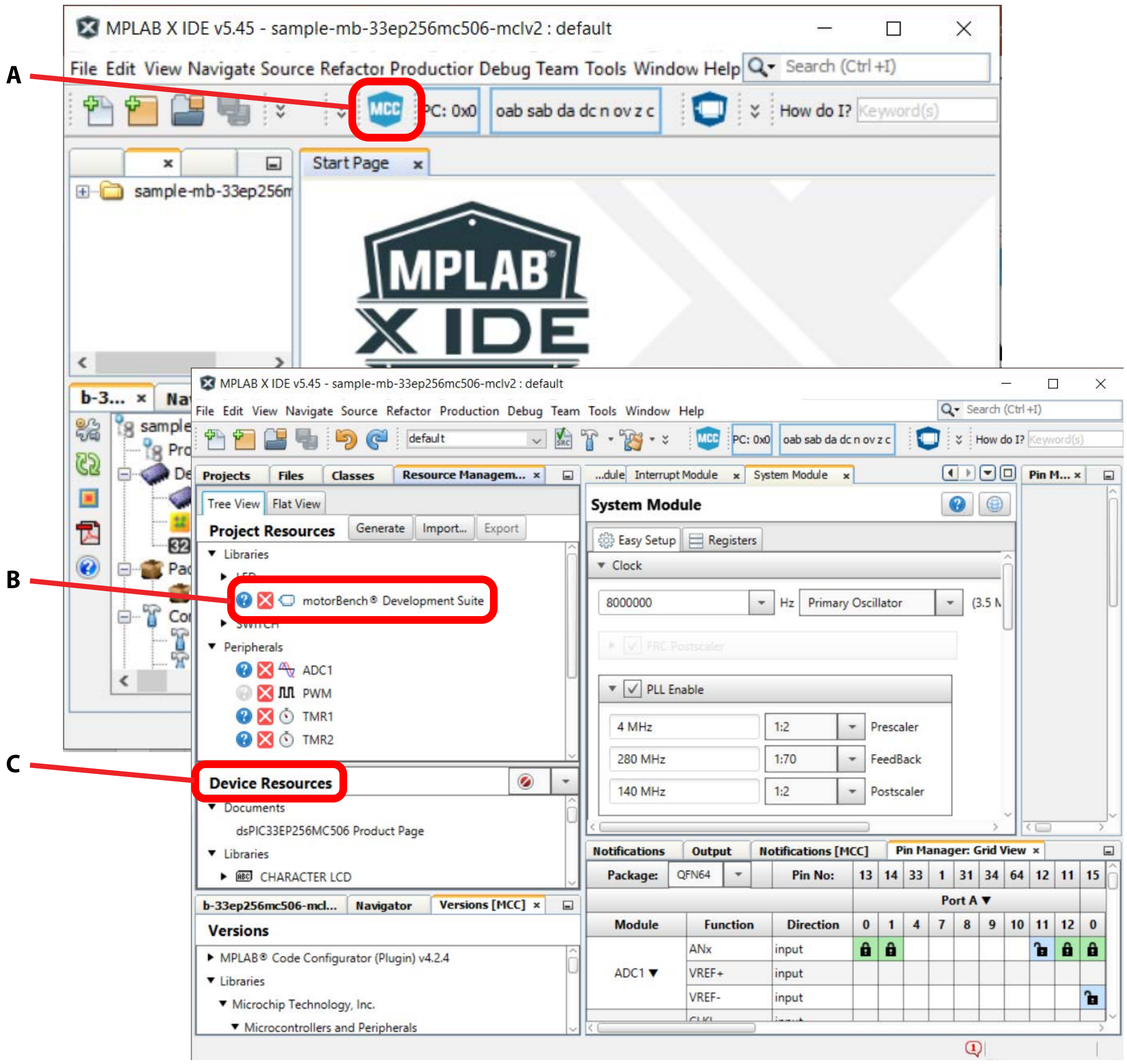
- We will be working with this project folder's contents **(A)** which is specific to the MA330031-2 PIM with dsPIC33EP256MC506 and that uses Microchip®

A



Launch motorBench® development suite

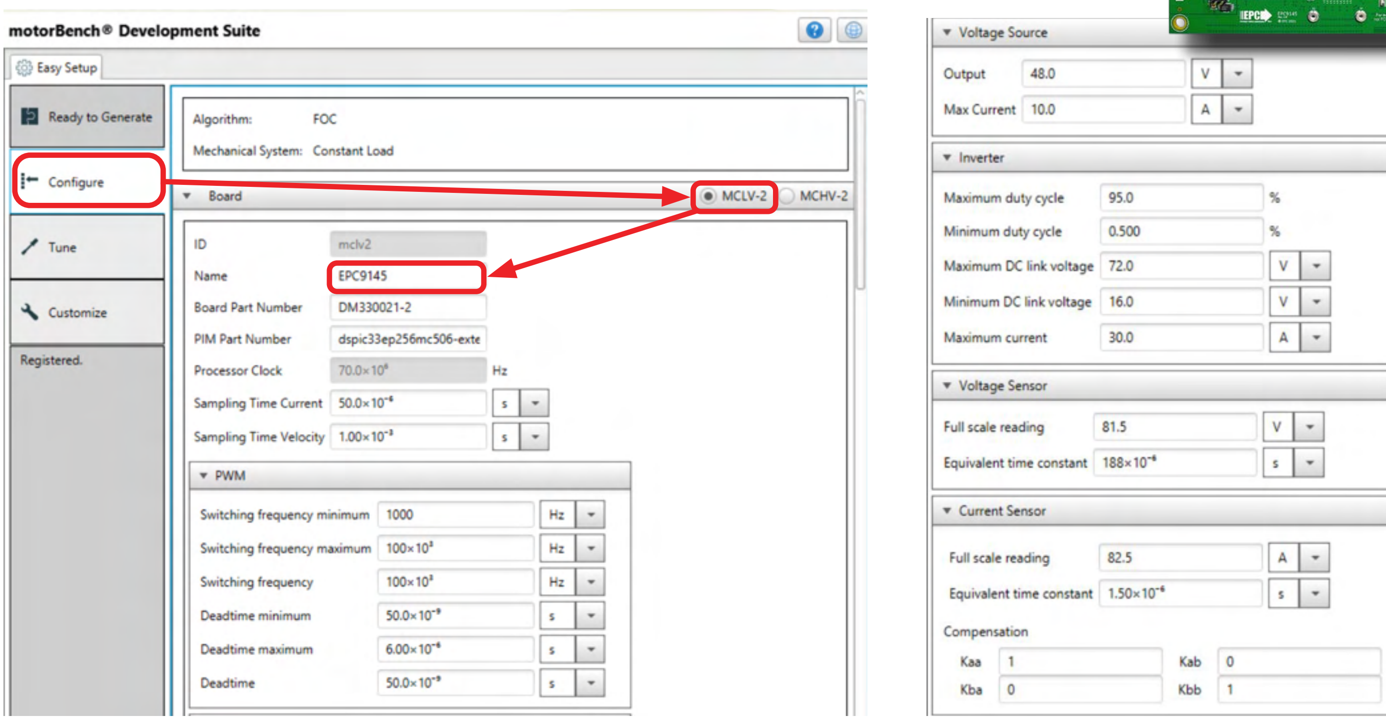
- Start MPLAB X IDE
- Open sample project
- Click on MCC icon **(A)**
- Click on motorBench Project resource **(B)**
  – If motorBench is not visible, check Device Resources **(C)**



CONFIGURE motorBench® TO THE INVERTER BOARD
PICK ONE OF THE FOLLOWING OPTIONS

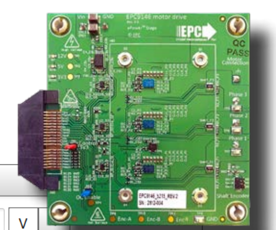
motorBench Configure/Board Specific Parameters for the Power Board EPC9145

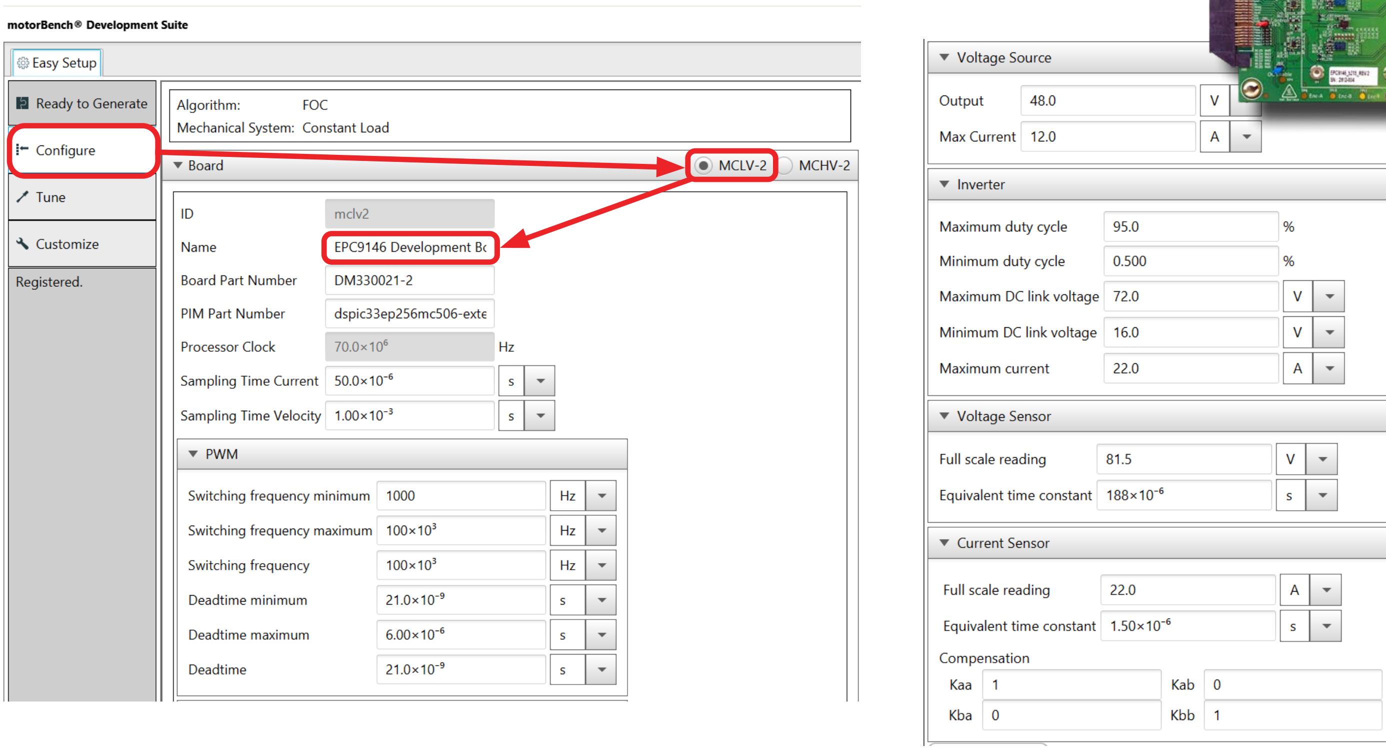
- Make sure that all parameters are set as shown



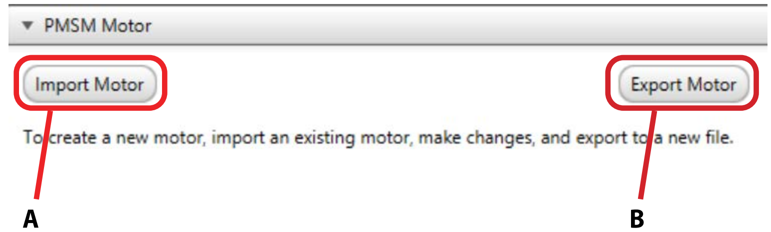
motorBench Configure/Board Specific Parameters for the Power Board EPC9146

- Make sure that all parameters are set as shown. **Note: Processor clock does not need to change**





CONFIGURE motorBench® TO THE MOTOR



motorBench Configure/PMSM Motor Parameters

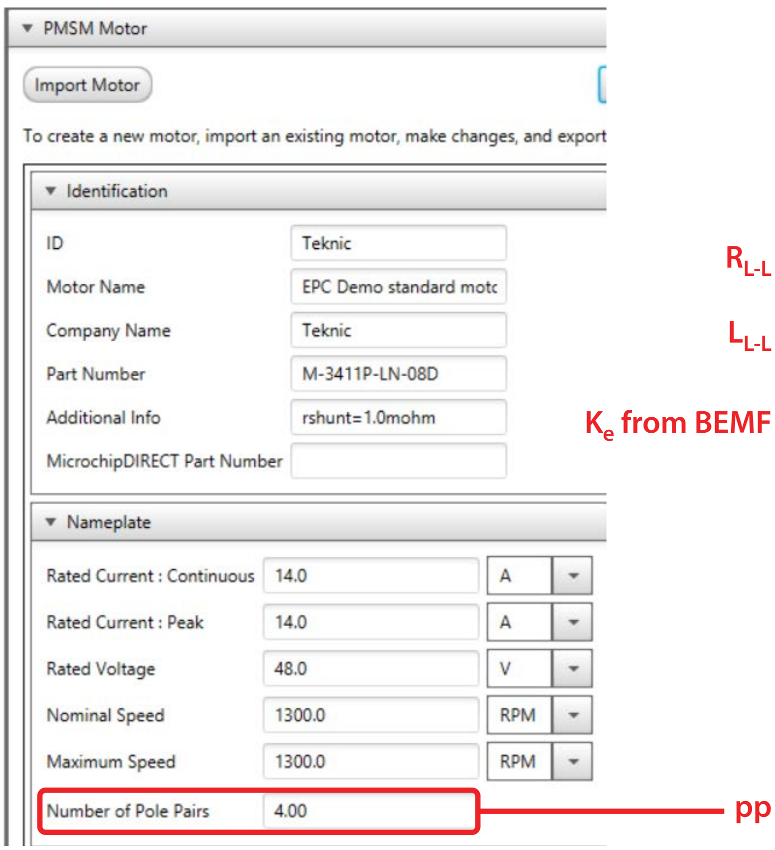
Have an **existing motor config file *.xml**

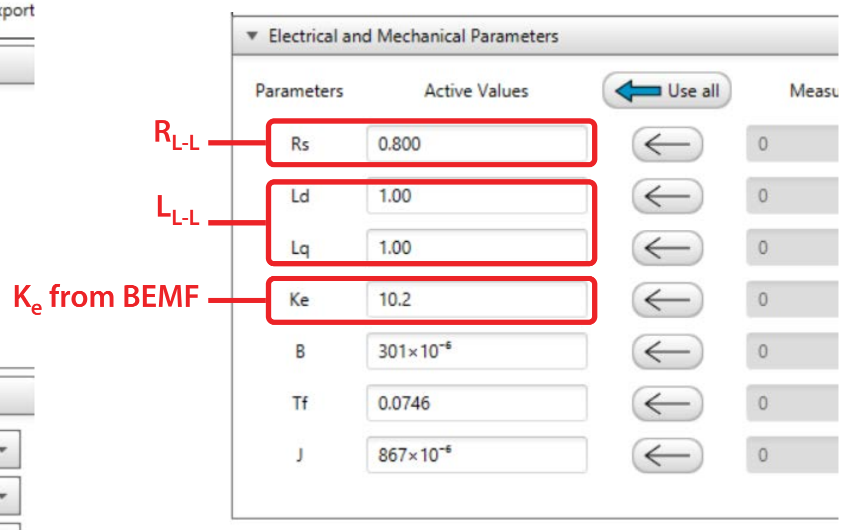
- click on "Import Motor" **(A)**
- Xml file available on EPC website for specific motor

OR

Need a **new motor config file *.xml**

- click on "Export Motor" **(B)**
- This will export a blank *.xml motor file, which you can then import using the "Import Motor" button
- Make sure that all parameters are set as shown below.
- Parameters are not explicit to board used.
- Used $L_d = L_q$ = 1 mH in this example, despite measuring 932 µH.



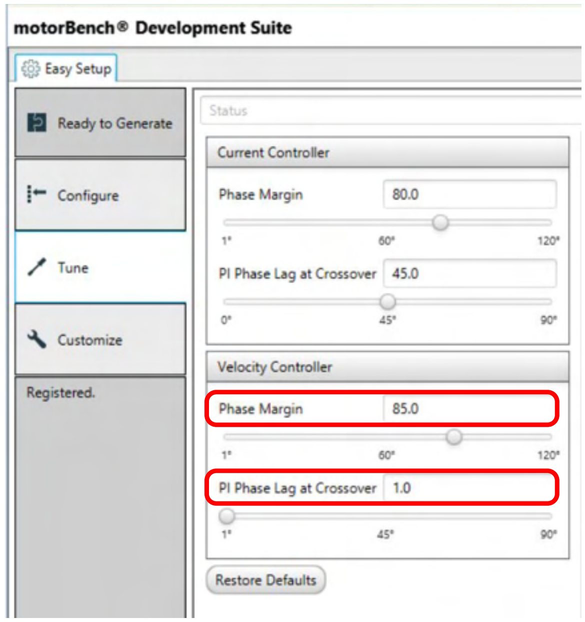


CONFIGURE motorBench® TO THE CONTROLLER

motorBench Configure/Controller Parameters

Make sure that all parameters are set as shown

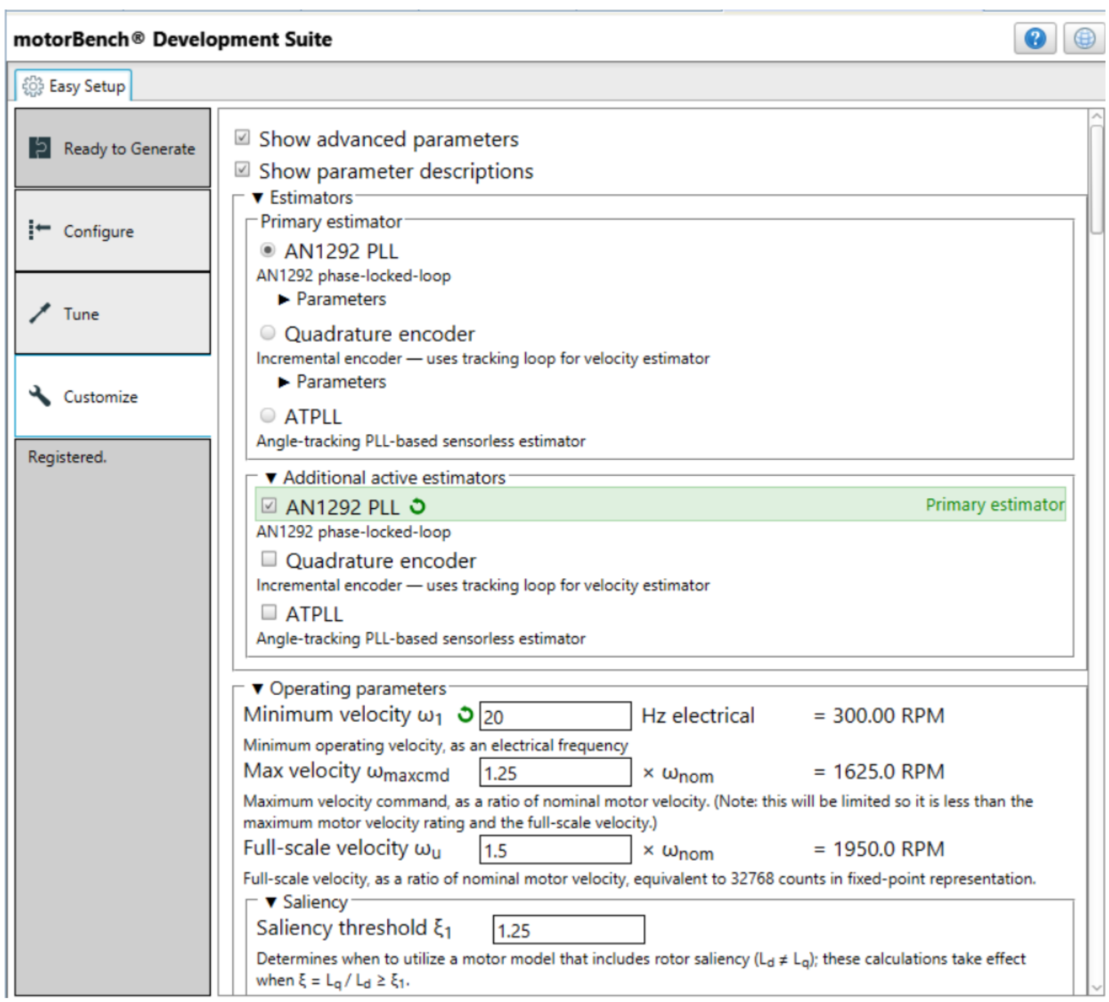
- Fine tune speed loop dynamics in subsequent step (by modifying the C code)



motorBench Customize Parameters 1

Make sure that all parameters are set as shown

- Ensures FOC sensor-less algorithm is set and correctly configured



Parameter 1 screen continues on next page

motorBench Customize Parameters 1 *(continued)*

when $\xi = L_q / L_d \geq \xi_1$.
Impacts some sensorless estimator calculations (ATPLL, for example) as well as MTPA/flux weakening.

Saliency ratio ξ — L_q/L_d = 1.0000

Ratio of q-axis inductance to d-axis inductance

▼ Coastdown

Velocity threshold k_c — [0.05] × ω_{max} = 65.000 RPM

Determines expected velocity at which waiting is no longer required before restarting. This is normalized to the maximum operating velocity ω_{max}.

Time — [1.2] × t_{c1} = 1.4195 s

Coastdown time, normalized to natural coastdown time $t_{c1} = -(J/B)\ln\frac{k_c\omega_{max}+\omega_{fr}}{\omega_{max}+\omega_{fr}}$ where $\omega_{fr} = T_{fr}/B$

⚠ **WARNING:** Coastdown time is the delay time to allow the motor to come to a stop before a restart is allowed. Reducing the coastdown time below 1.0 × t_{c1} for large inertia motors may cause large motor currents to be generated from the motor back-emf. **Excessive current flow may damage components such as sense resistors or power transistors, which may pose a risk of injury or property damage.**

▼ Slew rate

Max acceleration α_+ — [0.5] × α_{max} = 8.7815 kRPM/s

Determines maximum acceleration in motoring quadrants. This is normalized to the maximum expected acceleration α_{max}, taking into account friction torque and maximum current.

Max deceleration α_- — [1] × α_n = 0.92584 kRPM/s

Determines maximum acceleration in generating quadrants. This is normalized to natural deceleration α_n at minimum operating velocity ω_1, where $\alpha_n = (T_{fr} + B\omega_1)/J$.

⚠ **WARNING:** Deceleration faster than 1.0 × α_n may regenerate energy back onto the DC link, requiring either energy storage or dissipation. **Failure to manage regeneration energy may cause excessive DC link voltage and may damage components connected to the DC link, such as electrolytic capacitors and power transistors, which may pose a risk of injury or property damage.**

▼ Flux control

Flux control method

◉ None
No flux control $(I_d = 0)$

○ Equation-based
Equation-based flux control with flux-weakening and MTPA

▼ Dead-time compensation

Method

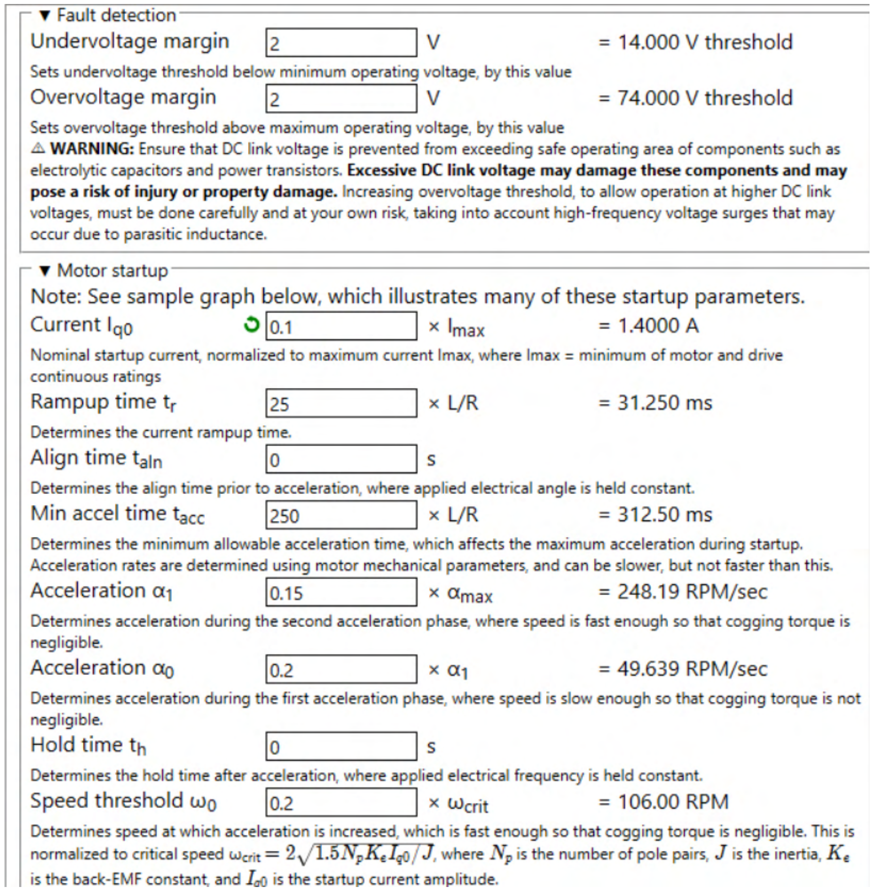
◉ None
No dead-time compensation

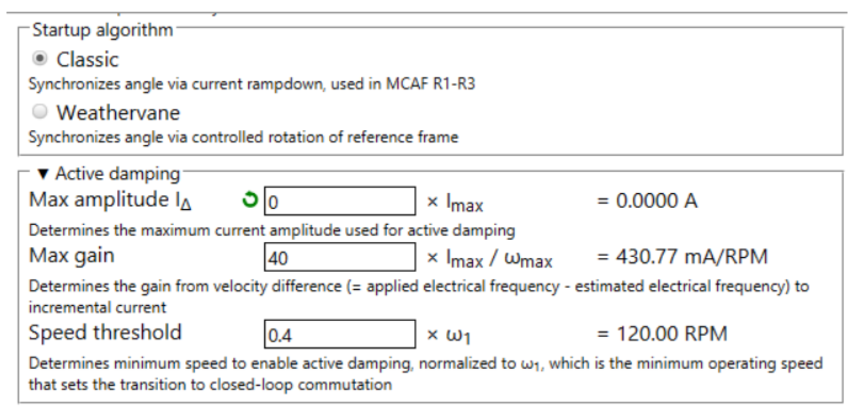
○ Per-phase
Per-phase dead-time compensation

motorBench Customize Parameters 2

Make sure that all parameters are set as shown

- Ensures FOC sensor-less algorithm is set and correctly configured





Parameter 2 screen continues on next page

motorBench Customize Parameters 2 *(continued)*

▼ Overmodulation

D-axis limit [1] × V_{DC}

D-axis voltage limit normalized to DC link voltage. This rarely needs to be adjusted

Q-axis limit [1.15] × V_{DC}

Q-axis voltage limit normalized to DC link voltage. Represents a tradeoff between distortion and output voltage capability.

▼ Motion Control API

Filter time constant τ_{Is} [1] ms

Time constant used for calculating low pass filtered value of Is^2

Filter time constant τ_{Iq} [1] ms

Time constant used for calculating low pass filtered value of Iq

▼ Board Service

Ui service period [1] ms

Rate at which the Board Service tasks are executed

Button debounce time [7] ms

Debounce time: number of identical digital samples required before a change in button state (unpressed/pressed) is recognized

Long button press time [2.5] s

The amount of time in which it takes to register a long button press

Advice

Commutation step at maximum motor velocity

$\theta_c = \omega_{m,\max} N_p T_{PWM}$ = 0.3120°

$\theta_c < 30°$	Smooth commutation: more than 12 steps per electrical cycle
$30° \le \theta_c < 60°$	Slightly better than six-step commutation
$\theta_c \ge 60°$	Poor commutation: fewer than 6 steps per electrical cycle

Field-oriented control works best when there are at least 12 PWM periods per electrical cycle, so that the resulting waveform minimizes distortion at harmonics of the electrical frequency.

If the step size is small enough (≈ 60 PWM periods per electrical cycle), and the current controllers operate every PWM cycle, they can often compensate for distortion due to PWM dead time. This works very well at low velocity but is less effective at the upper end of the motor's velocity range.

Ripple current at maximum DC link voltage

$I_R = \frac{V_{DC} T_{PWM}}{12L}$ = 0.008571 × I_{max}

$I_R < 0.2 I_{\max}$	Low ripple current (< 1.3% additional I^2R loss)
$0.2 I_{\max} \le I_R < 0.4 I_{\max}$	Moderate ripple current (< 5.3% additional I^2R loss)
$I_R \ge 0.4 I_{\max}$	High ripple current (≥ 5.3% additional I^2R loss)

I_R describes the worst-case peak amplitude of ripple current, which occurs when the three motor phases are switching at some permutation of (0%, 50%, 100%). Ripple current can approach this value at high modulation indices. The RMS value of ripple current is $I_R/\sqrt{3}$.

It can be a concern for low-inductance motors, for three reasons:

- It causes additional I^2R dissipation in the motor windings
- It may cause the current sense signal conditioning circuitry to saturate, so that ADC readings of current are lower than their true value. (In center-aligned PWM, if the ADC samples at the pulse center, much of the ripple current component is rejected, but this relies on linearity of the signal conditioning, which is violated if saturation occurs.)
- It may cause hardware overcurrent detection to trip at a lower current, reducing design margin.

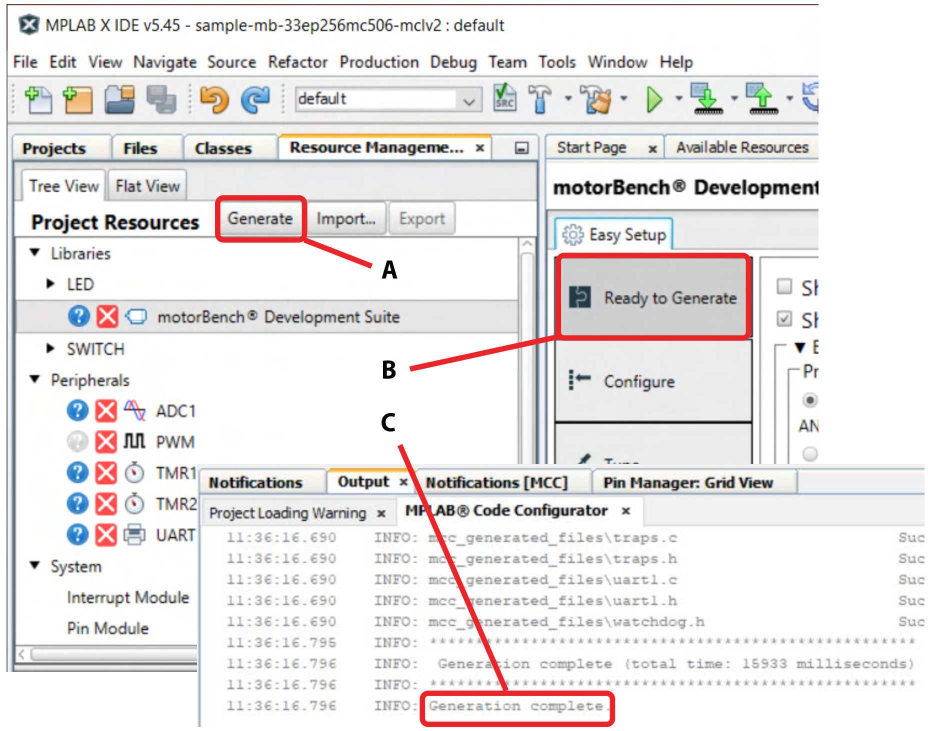
The impacts to saturation and hardware overcurrent detection can be minimal if the sensing and detection ranges are expanded to allow for ripple current, but the additional I^2R losses are unavoidable. One method of reducing ripple current is to increase the switching frequency, but this also increases the effect of dead-time distortion. Another method is to reduce the DC link voltage, as long as there is enough voltage available to allow the motor to achieve the desired torque and velocity.

Motor Control Application Framework
R6/RC8 (commit 102056, build on 2020 Aug 25 14:43)

GENERATING THE CONTROL FIRMWARE

Generate the Code

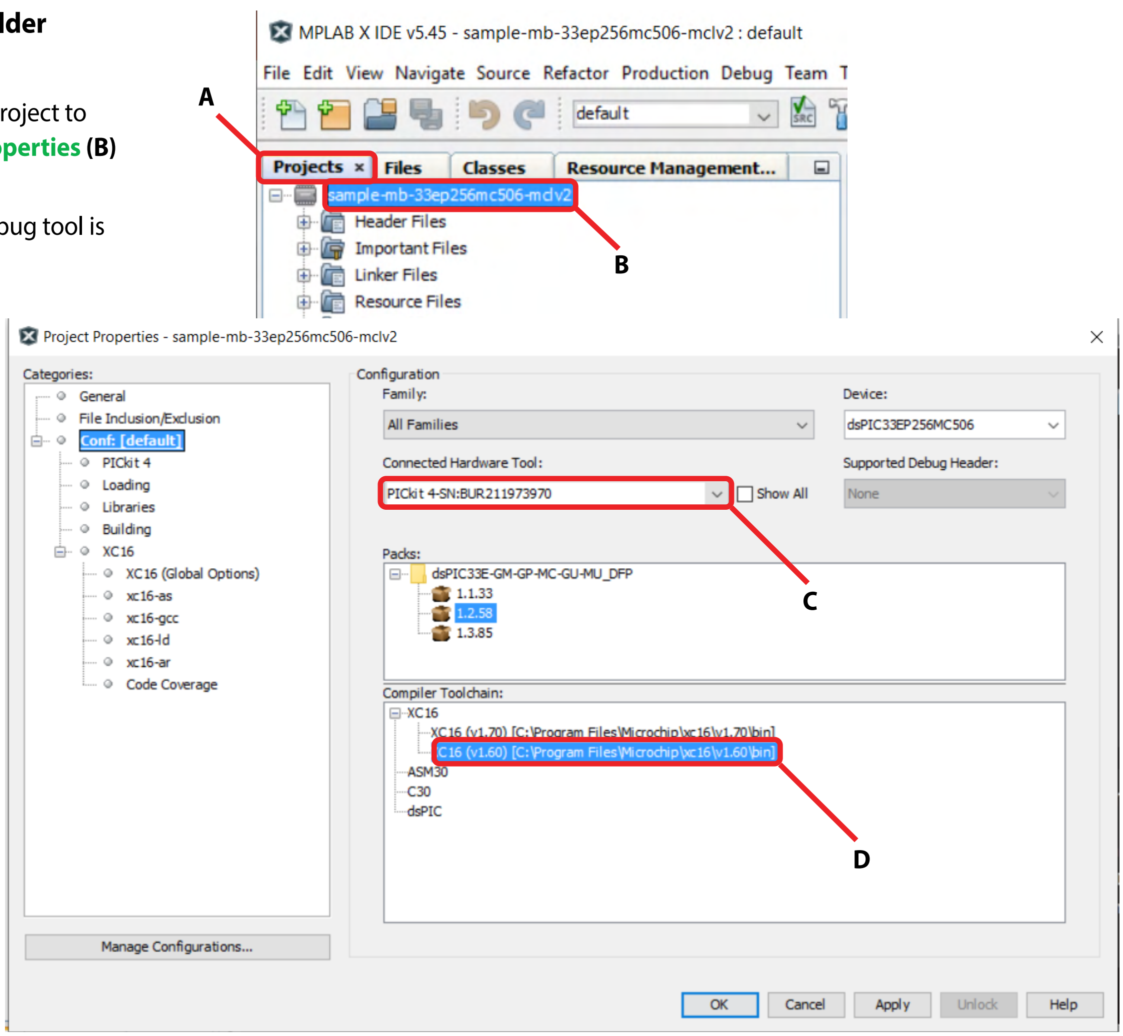
1. If everything is correct, message **Ready to Generate** **(A)** will appear.
2. Once **all** parameters are correctly set:
3. Generate code by pressing the **Generate** **(B)** button.
4. Wait for **Generation complete** **(C)** message



Setup Compiler and Builder

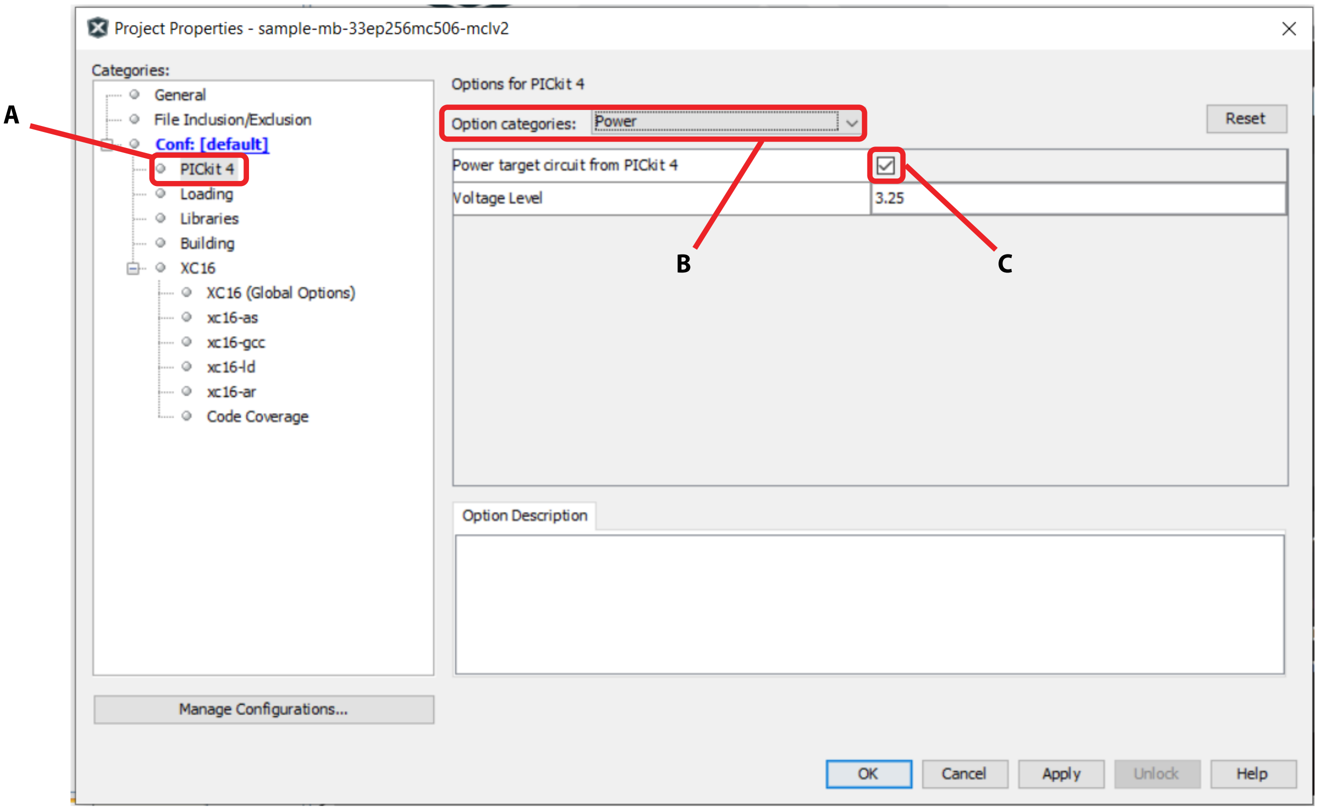
1. Select **Projects** tab **(A)**
2. Right click on the active project to configure the **project properties** **(B)** and set as main project
3. Make sure that proper debug tool is selected **(C)**

Make sure the proper compiler version is selected **(D)**



Setup Debug tool Power option

1. Select the **debug tool** (**A**) (e.g. PICkit4)
2. Select **Power** option category (**B**)
3. Make sure to a check the **Power target circuit from PICkit4** option **(C)**



Build and Flash

1. Connect the programmer (e.g. PICkit-4) to the EPC9147A as shown
2. Press the **Make button**
3. Wait for **BUILD SUCCESSFUL** and for **Programming/ Verifying complete**

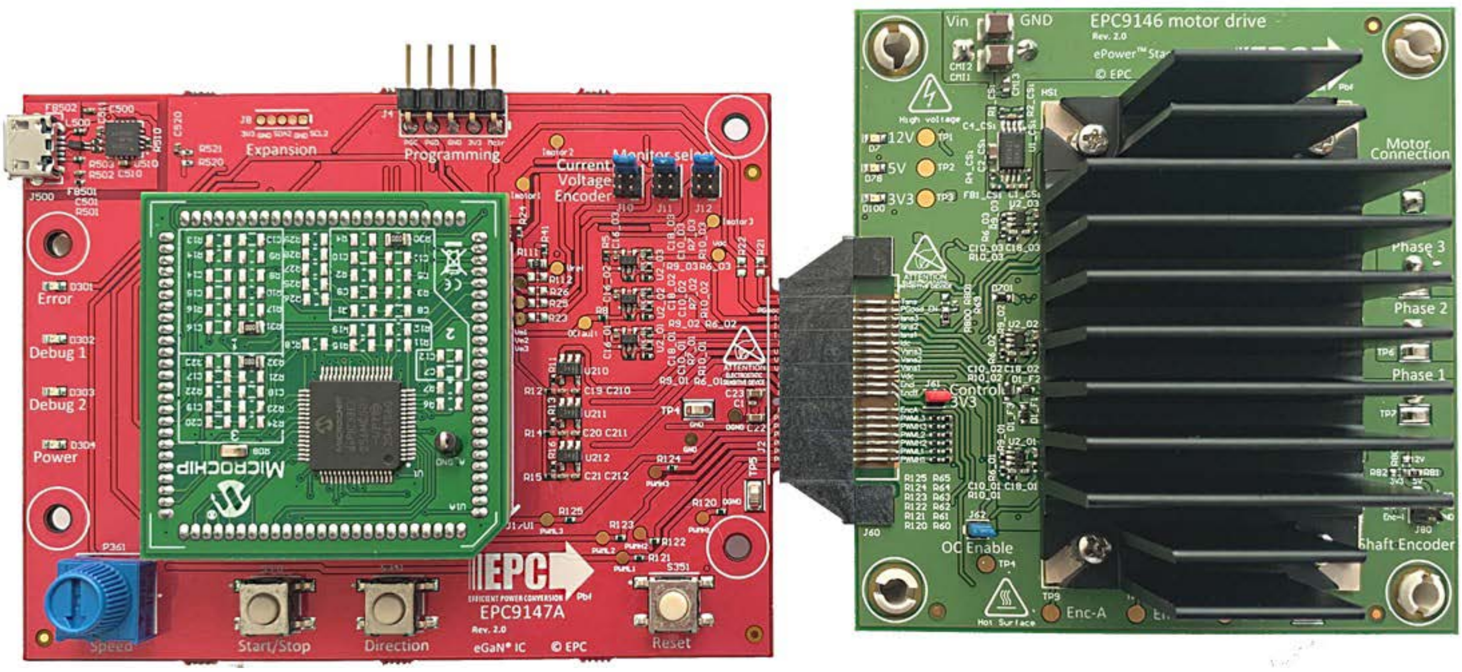
- Note: After programming **green LED** should be on and **orange** and **blue** LED's should flash

4. Disconnect programmer from EPC9147A

OPERATING THE MOTOR DRIVE SYSTEM

Operate the Motor Drive System

1. Connect the EPC9147A to a compatible inverter board; e.g. EPC9146
2. Connect the motor to the inverter board. **Follow QSG instructions**.
3. With power **OFF**, connect the power supply to the inverter board. Make sure the 3V3 jumper is installed to power the controller.
4. Set the power supply to the correct operating voltage for the inverter board. Make sure the current limit setting is sufficient to operate the motor drive system. For EPC9146 V_{sup} = 48 V and I_{lim} > 2.5 A
5. Power on and operate



For More Information:

Please contact **info@epc-co.com**
or your local sales representative

Visit our website:
www.epc-co.com

Sign-up to receive
EPC updates at
bit.ly/EPCupdates
or text "EPC" to 22828

EFFICIENT POWER CONVERSION

EPC Products are distributed through Digi-Key.
www.digikey.com