Hall Sensor-based Six-Step Commutation for BLDC Motor with Hall Sequence Identifier : MCLV-48V-300W and dsPIC33CK256MP508 Motor Control DIM

1. INTRODUCTION

This document describes the setup requirements for driving a Brushless DC (BLDC) Motor using six-step commutation with Hall sensor feedback and a Hall sequence identifier on the hardware platform EV18H47A "MCLV-48V-300W Development Board" and EV62P66A "dsPIC33CK256MP508 Motor Control Dual In-line Module (DIM)".

For details about six-step commutation of BLDC motor using Hall Sensor feedback, refer to Microchip application note AN957 "Sensored BLDC Motor Control". The Hall sequence identifier detects correct sequences of Hall and phase connections, allowing users the flexibility to connect Hall and phase wires in any order.

Enhance your embedded applications with Microchip's high-performance dsPIC® Digital Signal Controllers (DSCs). Visit our Motor Control and Drive page to stay updated on the latest motor control solutions from Microchip.

2. SUGGESTED DEMONSTRATION REQUIREMENTS

2.1 Motor Control Application Firmware Required for the Demonstration

To clone or download this application firmware on GitHub,

- Navigate to the main page of this repository and
- On the tab **<> Code**, above the list of files in the right-hand corner, click **Code**, then from the menu, click **Download ZIP** or copy the repository URL to **clone.**

> **Note:**
> In this document, hereinafter this firmware package is referred as **firmware.**

2.2 Software Tools Used for Testing the firmware

- MPLAB® X IDE **v6.25**
- Device Family Pack (DFP): **dsPIC33CK-MP_DFP v1.15.423**
- Curiosity/Starter Kits Tool Pack : **PKOB4_TP v1.19.1503**
- MPLAB® XC-DSC Compiler **v3.21**
- MPLAB® X IDE Plugin: **X2C-Scope v1.7.0**

> **Note:**
> The software used for testing the firmware prior to release is listed above. It is recommended to use

> the version listed above or later versions for building the firmware. All previous versions of Device Family Packs (DFP) and Tool Packs can be downloaded from Microchip Packs Repository.

2.3 Hardware Tools Required for the Demonstration

- MCLV-48V-300W Development Board (EV18H47A)
- dsPIC33CK256MP508 Motor Control Dual In-line Module (EV62P66A)
- 24V Power Supply (AC002013)
- 24V 3-Phase Brushless DC Motor - Hurst DMA0204024B101 (AC300022) or,
- 24V 3-Phase Brushless DC Motor - Hurst DMB0224C10002 (AC300020) or,
- 24V 3-Phase Brushless DC Motor - ACT 57BLF02 (57BLF02) or,
- 24V 3-Phase Leadshine Servo Motor (ELVM6020V24FH-B25-HD)

> **Note:**
> All items listed above except Leadshine Servo Motor (ELVM6020V24FH-B25-HD) and ACT Brushless DC Motor (57BLF02) are available at microchip DIRECT
>
> - Hurst DMA0204024B101(AC300022) is referred as Hurst300 or Long Hurst in the firmware
> - Hurst DMB0224C10002(AC300020) is referred as Hurst075 or Short Hurst in the firmware
> - ACT Brushless DC Motor 57BLF02 is referred as ACT02 in the firmware
> - Leadshine Servo Motor ELVM6020V24FH-B25-HD is referred as leadshine24v in the firmware

3. HARDWARE SETUP

This section describes the hardware setup required for the demonstration.

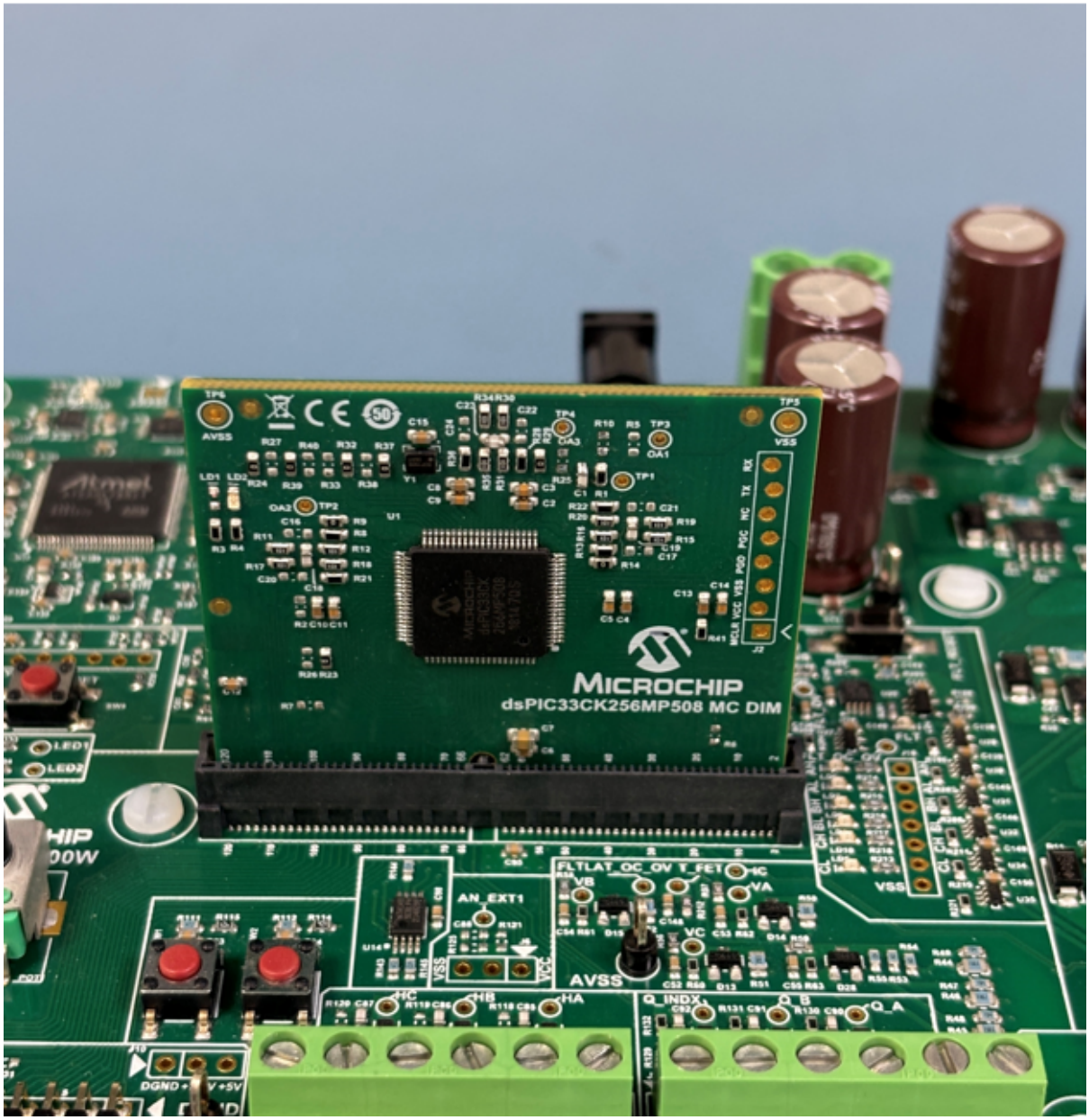
> **Note:**
> In this document, hereinafter the MCLV-48V-300W Development Board is referred as **development board**.

1. Motor currents are amplified on the MCLV-48V-300W development board; it can also be amplified by the amplifiers internal to the dsPIC33CK256MP508 on the DIM. By default, the firmware and DIM are set to sample and convert the outputs of the internal amplifier (**'internal op-amp configuration'**) to measure motor currents. **Table-1** summarizes the resistors to be populated and removed to convert the DIM from **'internal op-amp configuration'** to **'external op-amp configuration'** or vice versa.

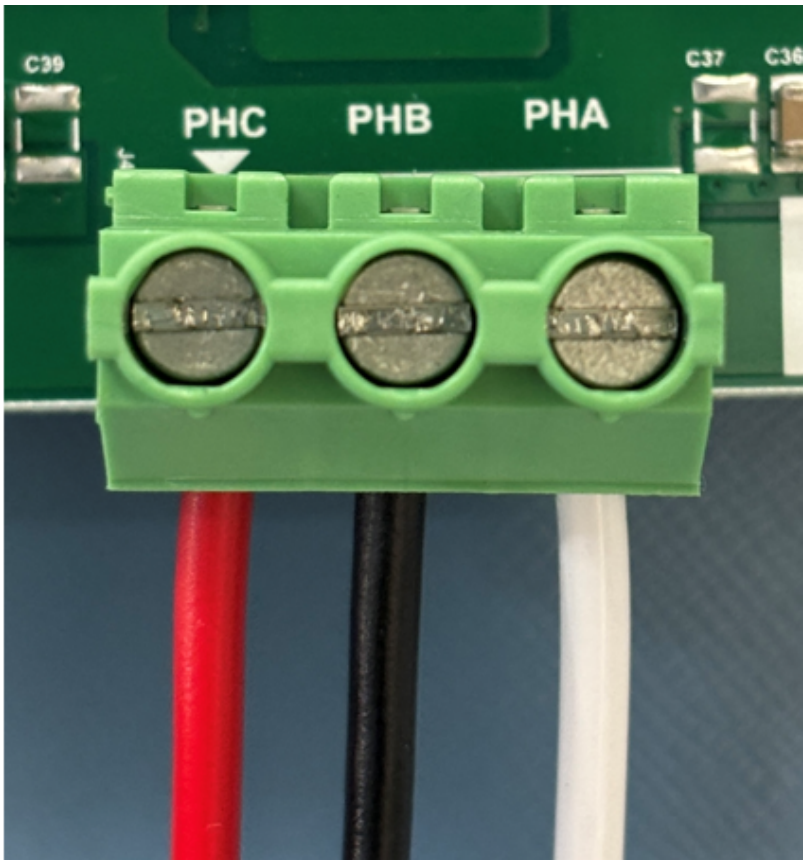
TABLE 1 : SELECTION BETWEEN EXTERNAL AND INTERNAL AMPLIFIER OUTPUTS

Current Signal	Jumper Resistor (0R) settings on the DIM				Firmware setting
	Internal Amplifier Output		External Amplifier Output		
	Populate	Remove	Populate	Remove	
Phase Current **IA or IA_EXT**	R9	R6	R6	R9	• Configure and enable internal amplifiers of the dsPIC DSC in 'internal op-amp configuration' • Ensure that the internal amplifiers are disabled in the 'external op-amp configuration'.
Phase Current **IB or IB_EXT**	R29	R25	R25	R29	
Bus Current **IBUS or IBUS_EXT**	R14	R10, R5 and R7	R10	R14, R5 and R7	

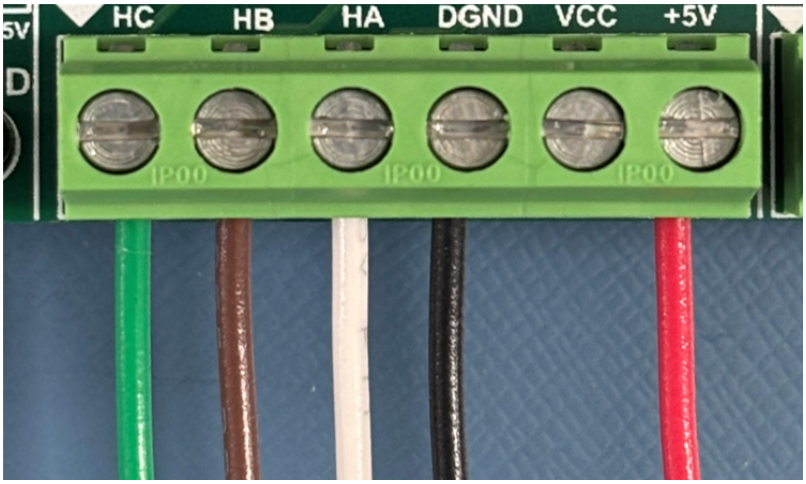
2. Insert the **dsPIC33CK256MP508 Motor Control DIM** into the DIM Interface **connector J8** on the development board. Make sure the DIM is placed correctly and oriented before going ahead.



3. Connect the 3-phase motor wires to PHA, PHB, and PHC of the **connector J4** provided on the development board, **in any order**.

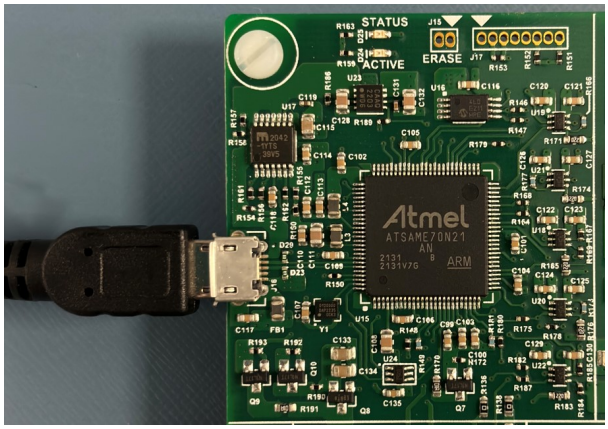


4. Connect the hall sensor's supply and ground wires to the Supply and Ground terminals of **connector J5** on the development board(see the figure). Then, **in any order**, connect the motor's hall sensor wires to the HA, HB, and HC of the **connector J5** provided on the development board.

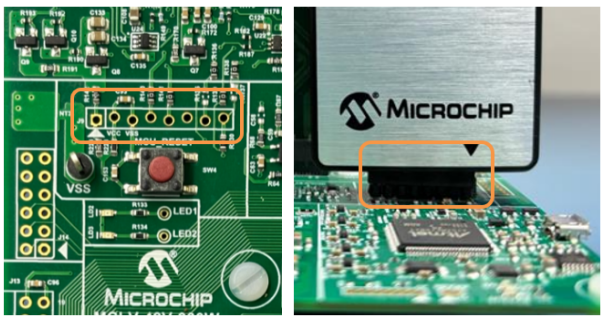


5. Plug the 24V power supply to **connector J1** on the development board. Alternatively, the development board can also be powered through connector J3.

6. The board has an onboard programmer **PICkit™ On Board (PKoBv4)** , which can be used for programming or debugging the microcontroller or dsPIC DSC on the DIM. To use the onboard programmer, connect a micro-USB cable between the Host PC and **connector J16** on the development board.



Alternatively, connect the Microchip programmer/debugger MPLAB® PICkit™ 5 In-Circuit Debugger(PG164150) between the Host PC used for programming the device and the **ICSP header J9** on the development board (as shown). Ensure that PICkit 5 is oriented correctly before proceeding.



4. SOFTWARE SETUP AND RUN

4.1 Setup: MPLAB X IDE and MPLAB XC-DSC Compiler

Install **MPLAB X IDE** and **MPLAB XC-DSC Compiler** versions that support the device **dsPIC33CK256MP508** and **PKoBv4.** The MPLAB X IDE, MPLAB XC-DSC Compiler, and X2C-Scope plug-in used for testing the firmware are mentioned in the Motor Control Application Firmware Required for the Demonstration section.

To get help on

- MPLAB X IDE installation, refer link
- MPLAB XC-DSC Compiler installation steps, refer link

If MPLAB IDE v8 or earlier is already installed on your computer, then run the MPLAB driver switcher (Installed when MPLAB®X IDE is installed) to switch from MPLAB IDE v8 drivers to MPLAB X IDE drivers. If you have Windows 8 or 10, you must run the MPLAB driver switcher in **Administrator Mode**. To run the Device Driver Switcher GUI application as administrator, right-click on the executable (or desktop icon) and select **Run as Administrator**. For more details, refer to the MPLAB X IDE help topic **"Before You Begin: Install the USB Device Drivers (For Hardware Tools): USB Driver Installation for Windows Operating Systems."**

4.2 Setup: X2C-SCOPE

X2C-Scope is a MPLAB X IDE plugin that allows developers to interact with an application while it runs. X2C-Scope enables you to read, write, and plot global variables (for motor control) in real-time. It communicates with the target using the UART. To use X2C-Scope, the plugin must be installed. To set up and use X2C-Scope, refer to the instructions provided on the web page.

5. BASIC DEMONSTRATION

5.1 Firmware Description

The firmware version needed for the demonstration is mentioned in the section Motor Control Application Firmware Required for the Demonstration. This firmware is implemented to work on Microchip's Digital signal controller (dsPIC® DSC) **dsPIC33CK256MP508**. For more information, see the **dsPIC33CK256MP508 Family datasheet (DS70005349)**.

The Motor Control Demo application uses a push buttons to start or stop the motor and for direction reversal. Also, a potentiometer is used to vary the speed of the motor. This Motor Control Demo Application configures and uses peripherals like PWM, ADC, UART, OP-AMP, CMP, DAC etc. For more details, refer to Microchip Application note **AN957, "Sensored BLDC Motor Control Using dsPIC30F2010"** available on the Microchip website.

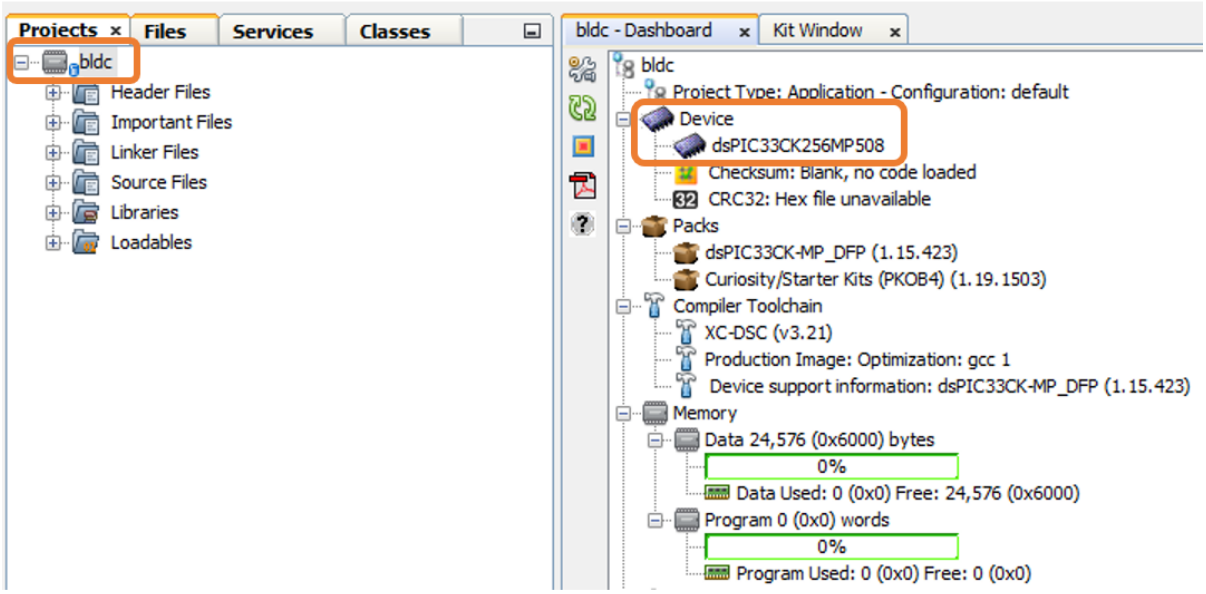
> **Note:**
> The project may not build correctly in Windows OS if the Maximum path length of any source file in the project is more than 260 characters. In case the absolute path exceeds or nears the maximum length, do any (or both) of the following:

> - Shorten the directory name containing the firmware used in this demonstration. If you renamed the directory, consider the new name while reading the instructions provided in the upcoming sections of the document.
> - Place firmware in a location such that the total path length of each file included in the projects does not exceed the Maximum Path length specified.
>
> Refer to MPLAB X IDE help topic **"Path, File, and Folder Name Restrictions"** for details.

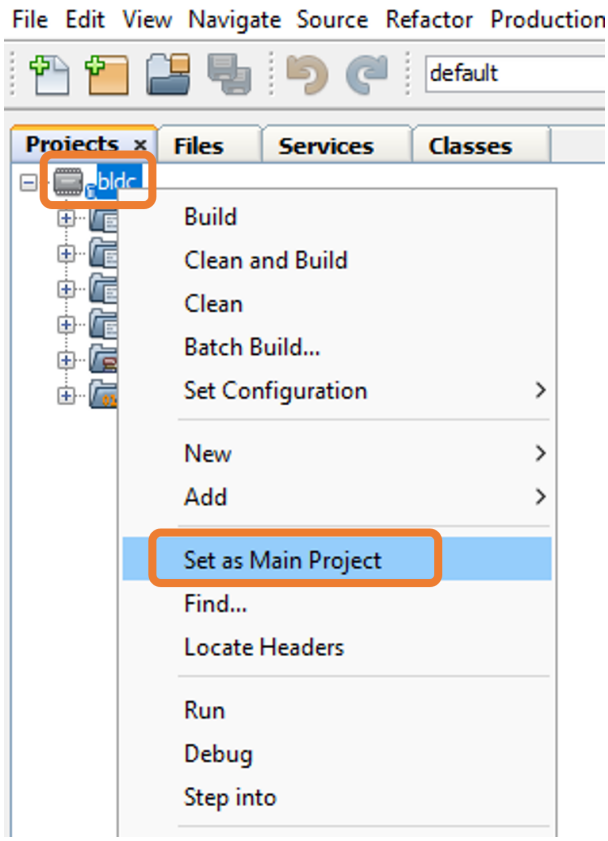
5.2 Basic Demonstration

Follow the below instructions, step by step, to set up and run the motor control demo application:

1. Start **MPLAB X IDE** and open the project **bldc.X (File > Open Project)** with device selection **dsPIC33CK256MP508.**



2. Set the project **bldc.X** as the main project by right-clicking on the project name and selecting **Set as Main Project** as shown. The project **bldc.X** will then appear in **bold.**



3. Open `hall_identifier.h` (**bldc.X > Header Files > hallsensor**) in the project **bldc.X.**

   - The macro **HALLSEQ_CURRENT_LIMIT_AMPS** is used to limit the current (in Amps) applied to the motor during the Hall sequence identification process.

```
/** Set the CURRENT LIMIT in Amps
*
* Current limit is set in Ampere to limit the current applied to the motor
* during the hall sequence identification process.
* The value for current limit is set to 10% to 30% of the motor rated current.
*
*/
#define HALLSEQ_CURRENT_LIMIT_AMPS 0.5
```

   - The macro **VECTOR_COMMUTATION_INTERVAL** is the interval between two different voltage vectors applied to the motor to determine hall sequence of the motor.

```
/* Set the Voltage VECTOR COMMUTATION INTERVAL
*
* Specify the VECTOR COMMUTATION INTERVAL in terms ADC ISR cycles (counts).
* This sets the interval between two different voltage vectors applied to the
* motor to determine hall sequence of the motor.
* This interval must be varied depending on the application and motor inertia.
* In this code the detection function is called in the ADC interrupt which
* occurs every 50 microseconds.
* e.g. VECTOR_COMMUTATION_INTERVAL(in seconds) = 20,000 * 50 usec = 1 second
*/
#define VECTOR_COMMUTATION_INTERVAL 20000
```

> **Note:**

> - It is assumed that the macros `HALLSEQ_CURRENT_LIMIT_AMPS` and `VECTOR_COMMUTATION_INTERVAL` are set appropriately for the sequence identification, these values must be varied depending on the application and motor inertia.
> - If a failure is detected during the test, the program will enter fault mode and **faultStatus** will display `MCAPP_HALLSEQ_IDENT_FAILURE`.
> - The sequence identification for all the Motors are tested under no load conditions. To achieve optimal performance under loaded conditions, the control parameters in the firmware may need additional tuning.

4. Open `mc1_user_params.h` (**bldc.X > Header Files**) in the project **bldc.X.**

   - Firmware is by default configured to run in closed-loop speed control using a PI controller.
   - **define** the macro to **CLOSED_LOOP 0** to enable open-loop duty control.
   - **define** the macro to **CLOSED_LOOP 1** to enable closed-loop speed control using a PI controller.
   - **define** the macro to **CLOSED_LOOP 2** to enable closed-loop current control using a PI controller.

```
/* Define macros for Operational Modes */

/*Control Loop Selection :
                        0 = Open-loop duty control
                        1 = Closed-loop speed control using a PI controller
                        2 = Closed-loop current control using a PI controller */
#define CLOSED_LOOP 1
```

   - Firmware is configured to run with Hurst DMA0204024B101 Motor(Hurst300 or Long Hurst-AC300022) by default.
   - **define** the macro to **MOTOR 2** to run with Hurst DMB0224C10002 Motor(Hurst075 or Short Hurst-AC300020).
   - **define** the macro to **MOTOR 3** to run with ACT 57BLF02 Brushless DC Motor (57BLF02).
   - **define** the macro to **MOTOR 4** to run with Leadshine Servo Motor (ELVM6020V24FH-B25-HD).
   - All the Motors are tested under no load conditions. To achieve optimal performance under loaded conditions, the control parameters in the firmware may need additional tuning.

```
/*Motor Selection : 1 = Hurst DMA0204024B101(AC300022: Hurst300 or Long Hurst)
                    2 = Hurst DMB0224C10002(AC300020: Hurst075 or Short Hurst)
                    3 = ACT 24V 3-Phase Brushless DC Motor - ACT 57BLF02
                    4 = Leadshine 24V Servo Motor ELVM6020V24FH-B25-HD (200W) */
#define MOTOR  1
```

   - When internal amplifiers are used for current amplification (referred to as **internal op-amp configuration**), **define** the macro `INTERNAL_OPAMP_CONFIG`

```
#define INTERNAL_OPAMP_CONFIG
```

   - Otherwise, if external amplifiers are used for current amplification (referred to as **external op-amp configuration**), **undefine** the macro `INTERNAL_OPAMP_CONFIG`

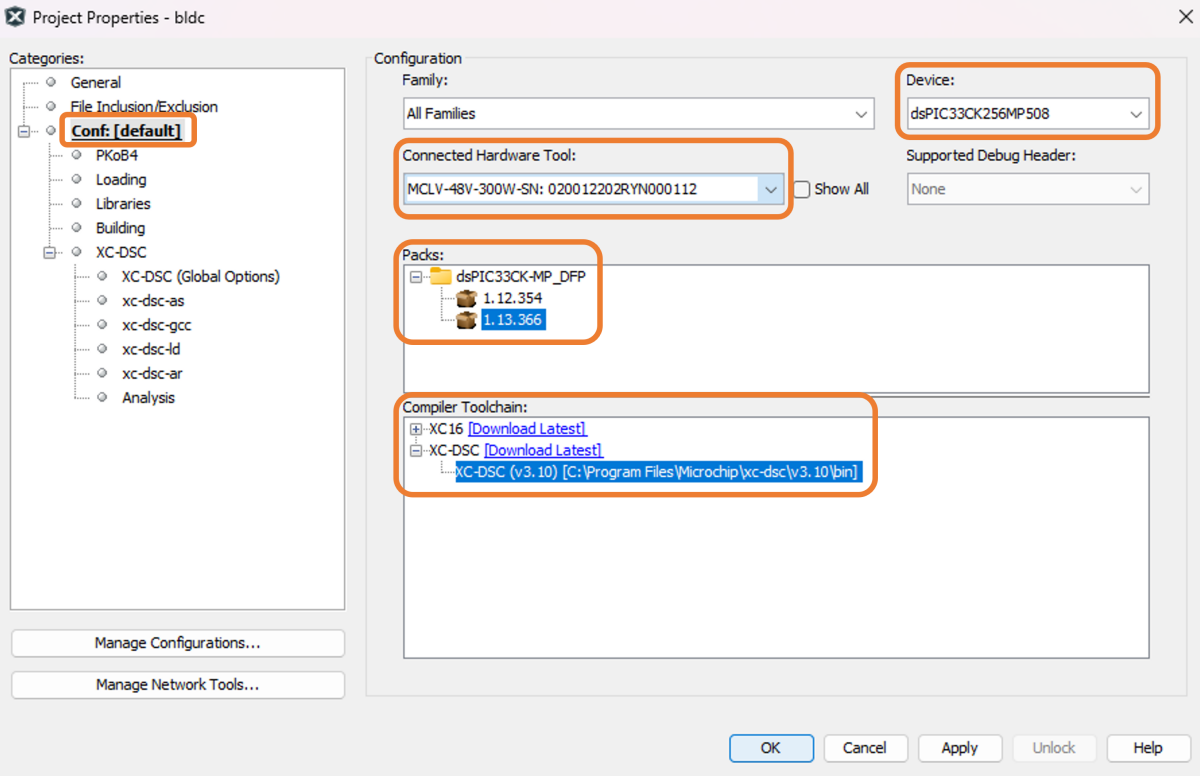
```
#undef INTERNAL_OPAMP_CONFIG
```

5. Right-click on the project **bldc.X** and select **Properties** to open its **Project Properties** Dialog. Click the **Conf:[default]** category to reveal the general project configuration information. The development tools used for testing the firmware are listed in section 2.2 Software Tools Used for Testing the firmware.

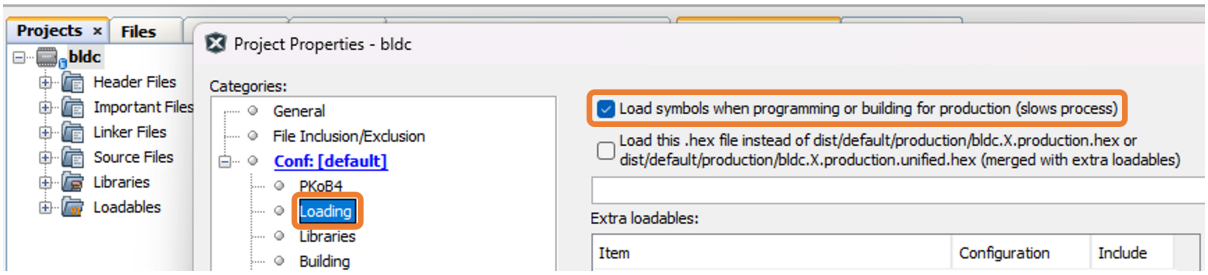
   In the **Conf:[default]** category window:

   - Ensure the selected **Device** is **dsPIC33CK256MP508.**
   - Select the **Connected Hardware Tool** to be used for programming and debugging.
   - Select the specific Device Family Pack (DFP) from the available list of **Packs.** In this case, **dsPIC33CK-MP_DFP 1.15.423** is selected.
   - Select the specific **Compiler Toolchain** from the available list of **XC-DSC** compilers. In this case, **XC-DSC(v3.21)** is selected.
   - After selecting Hardware Tool and Compiler Toolchain, Device Pack, click the button **Apply**

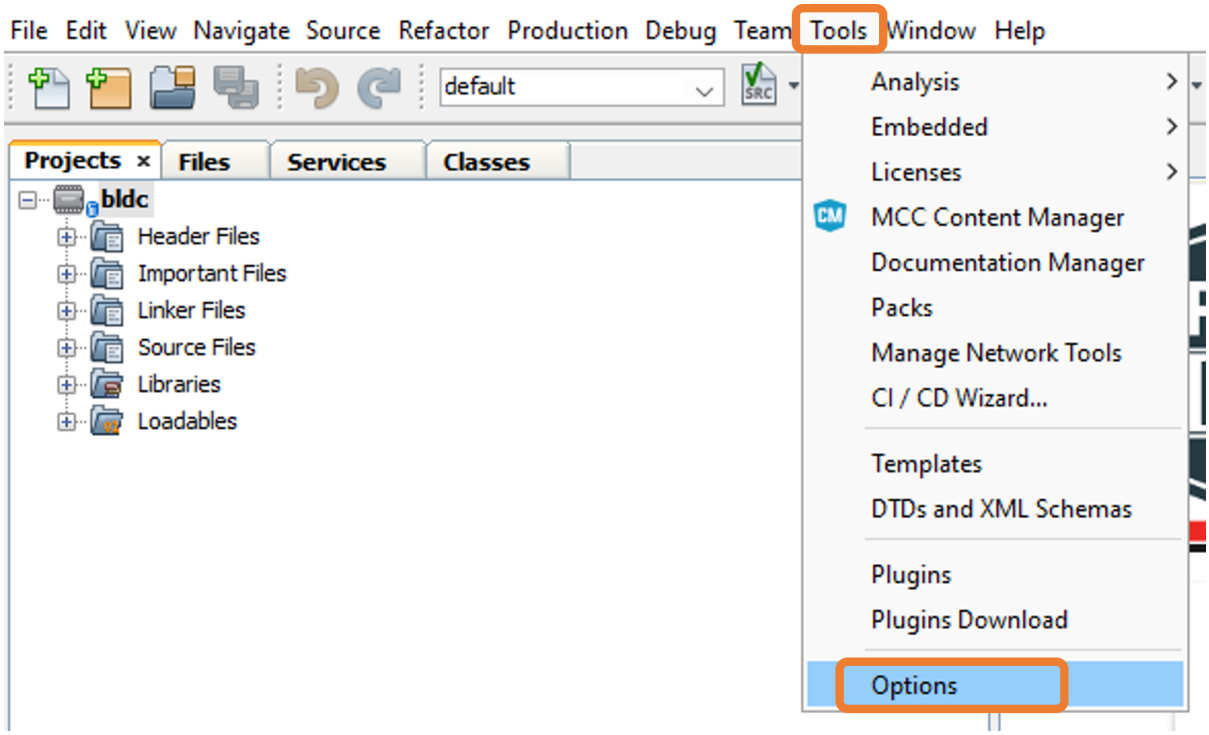
   Please ensure that the selected MPLAB® XC-DSC Compiler and Device Pack support the device configured in the firmware



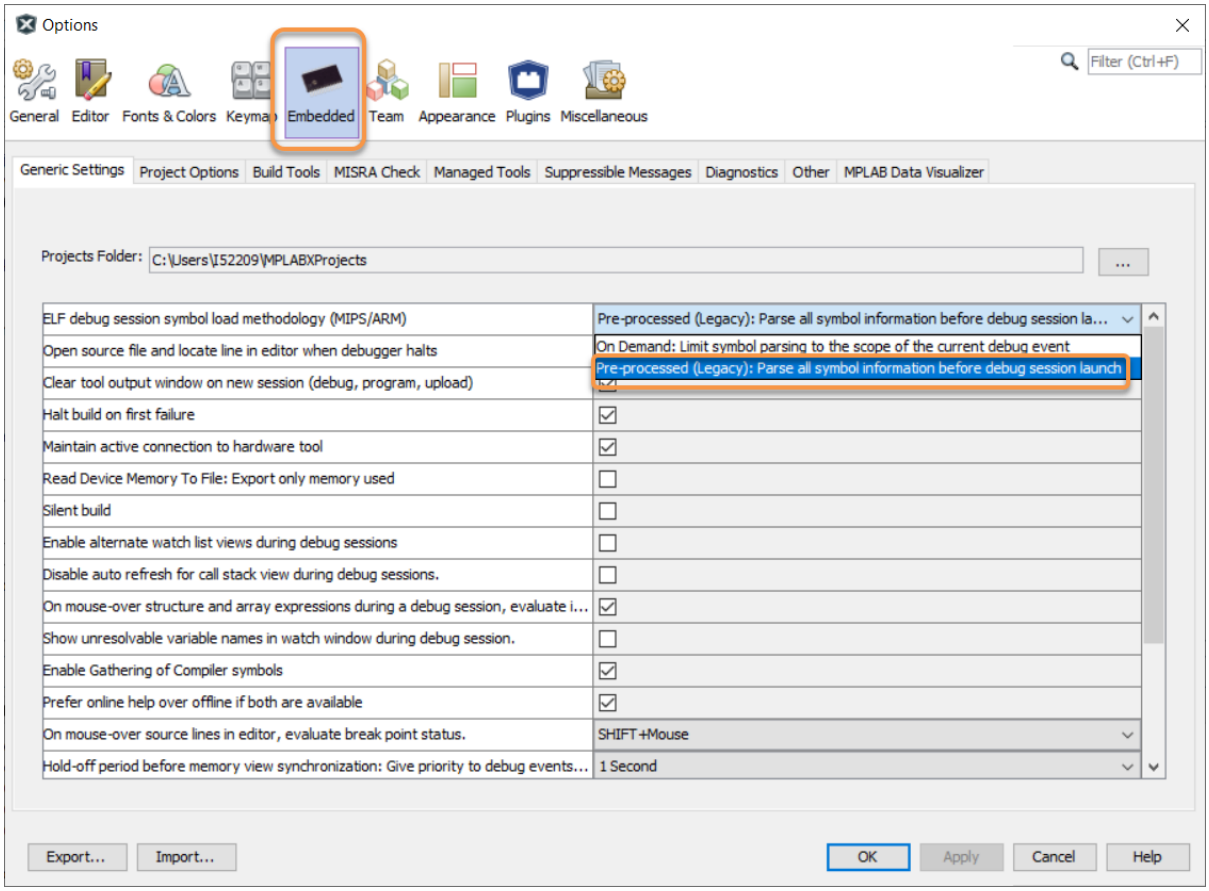
6. Ensure that the checkbox **Load symbols when programming or building for production (slows process)** is checked under the **Loading** category of the **Project Properties** window.



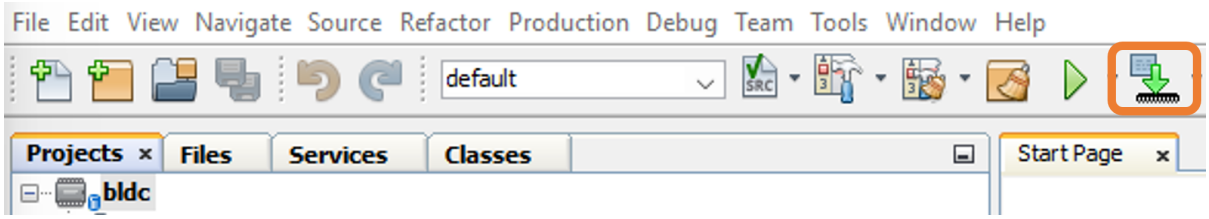
Also, go to **Tools > Options** , and



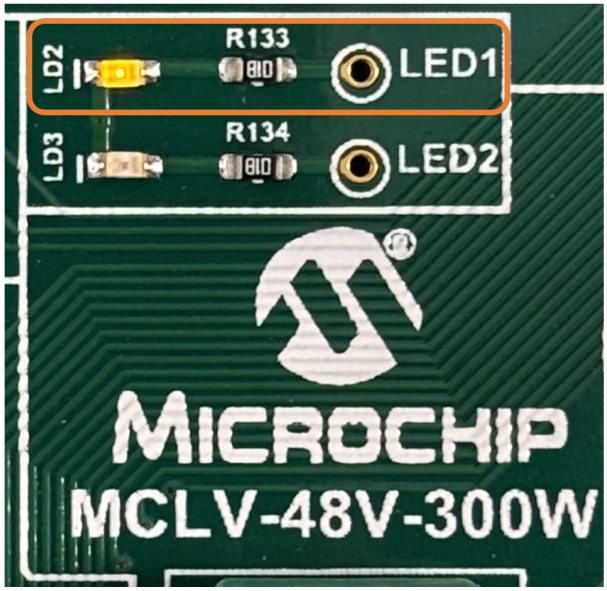
Open the **Embedded > Generic Settings** tab and ensure that the **ELF debug session symbol load methodology (MIPS/ARM)** is selected as **Pre-procesed (Legacy)** from the drop down.



7. To build the project (in this case, **bldc.X**) and program the device dsPIC33CK256MP508, click **Make and Program Device Main project** on the toolbar

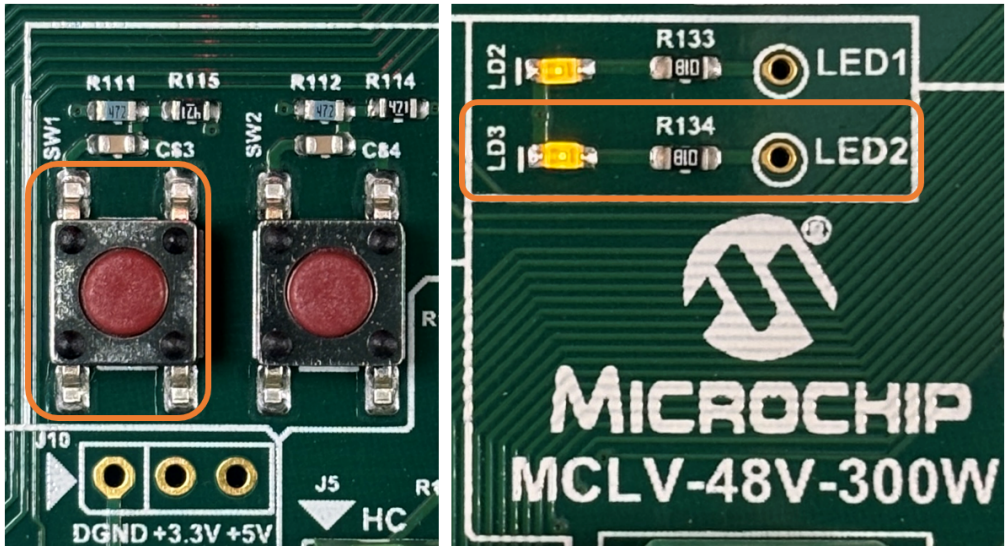


8. If the device is successfully programmed, **LED1 (LD2)** will be turned **ON**, indicating that the dsPIC® DSC is enabled.



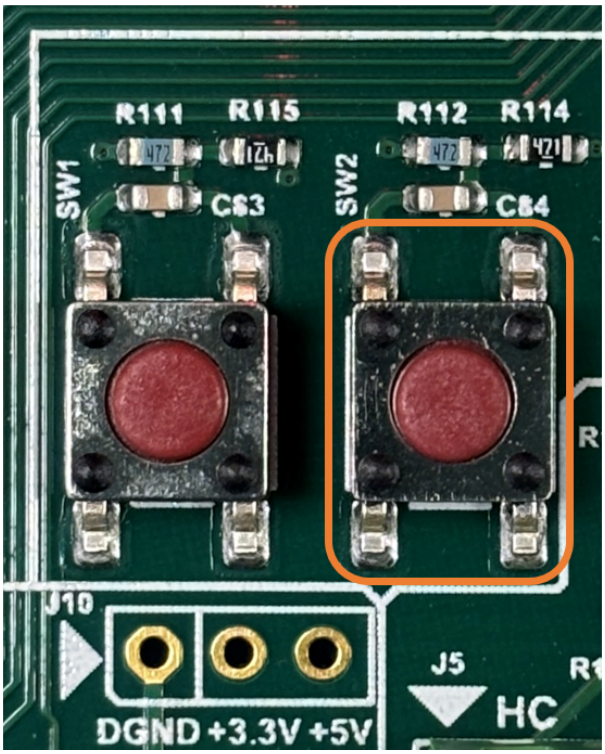
9. The Hall sequence identifier executes to determine the correct hall sensor pattern and load the corresponding inverter switching sequence. The motor starts running only after the test is complete, and this process occurs only once during power-on.

10. Run or stop the motor by pressing the push button **SW1.** The motor should start spinning smoothly in one direction in the nominal speed range. Ensure that the motor is spinning smoothly without any vibration. The **LED2 (LD3)** is turned **ON** to show that the button has been pressed to start the motor. The specific motor was tested under no load conditions. To achieve optimal performance under loaded conditions, the control parameters in the firmware may need additional tuning.



11. The motor speed can be varied using the potentiometer **(POT1).**

12. Press the push button **SW2** to change the direction of rotation of the motor.



13. Press the push button **SW1** to stop the motor.

> **Note:**
> The macros `POLE_PAIRS`, `MINIMUM_SPEED_RPM`, `MAXIMUM_SPEED_RPM`, `DIRECTION_CHANGE_SPEED_RPM`, and `NOMINAL_CURRENT_BUS_RMS` are defined in the respective motor header files. Exceeding manufacture's specifications may damage the motor or the board or both.

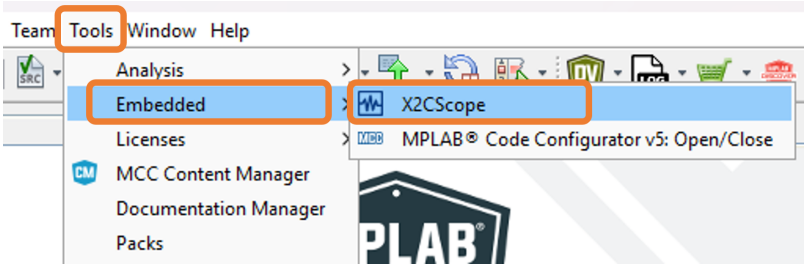
5.3 Data visualization through X2C-Scope Plug-in of MPLAB X

X2C-Scope is a third-party plug-in in MPLAB X, which helps in real-time diagnostics. The application firmware comes with the initialization needed to interface the controller with the host PC to enable data visualization through the X2C-Scope plug-in. Ensure the X2C-Scope plug-in is installed. For more information on how to set up a plug-in, refer to either the Microchip Developer Help page or the web page.

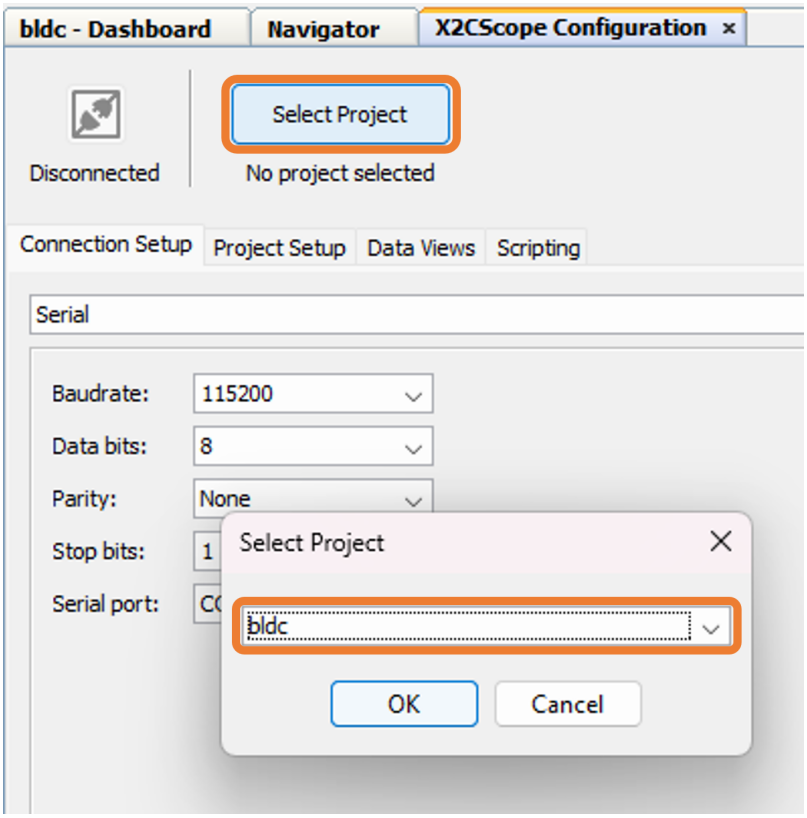
1. To establish serial communication with the host PC, connect a micro-USB cable between it and **connector J16** on the development board. The same interface is also used for programming.

2. Ensure the application is configured and running as described under section 5.2 Basic Demonstration by following steps 1 through 12.

3. Open the **X2C-Scope** window by selecting **Tools>Embedded>X2CScope.**

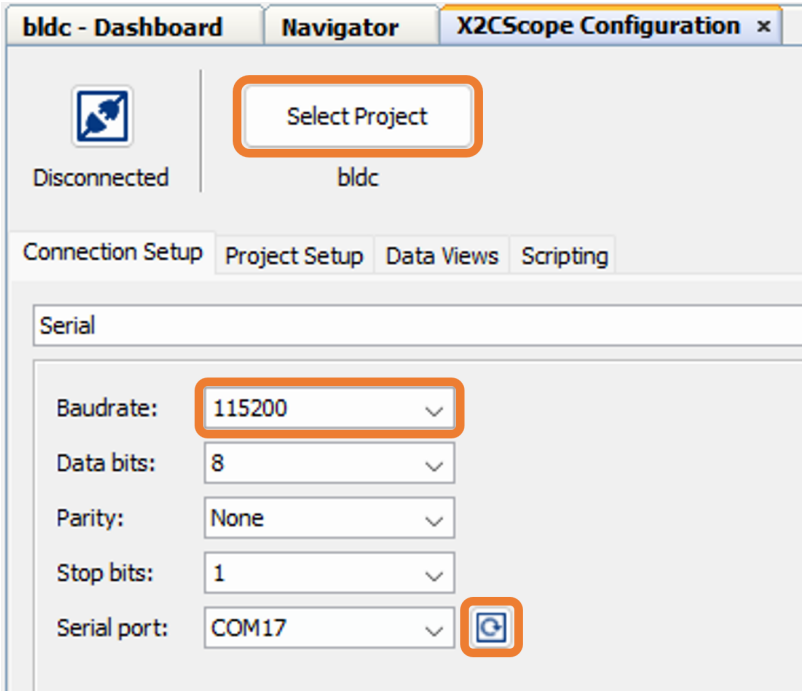


4. **In the X2C-Scope Configuration** window, open the **Connection Setup** tab and click **Select Project.** This opens the drop-down menu **Select Project** with a list of opened projects. Select the specific project **pmsm** from the list of projects and click **OK.**

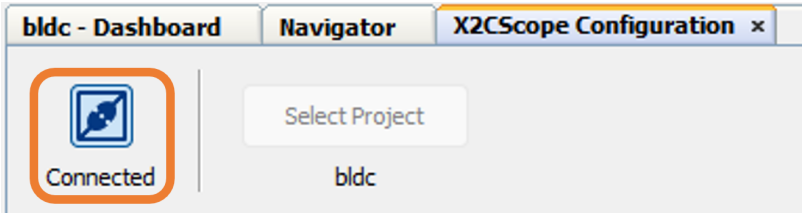


5. To configure and establish the serial communication for **X2C-Scope**, open the **X2CScope Configuration** window, click on the **Connection Setup** tab and:

   - Set **Baudrate** as **115200**, which is configured in the application firmware.
   - Click on the **Refresh** button to refresh and update the list of the available Serial COM ports connected to the Host PC.
   - Select the specific **Serial port** detected when interfaced with the development board. The **Serial port** depends on the system settings

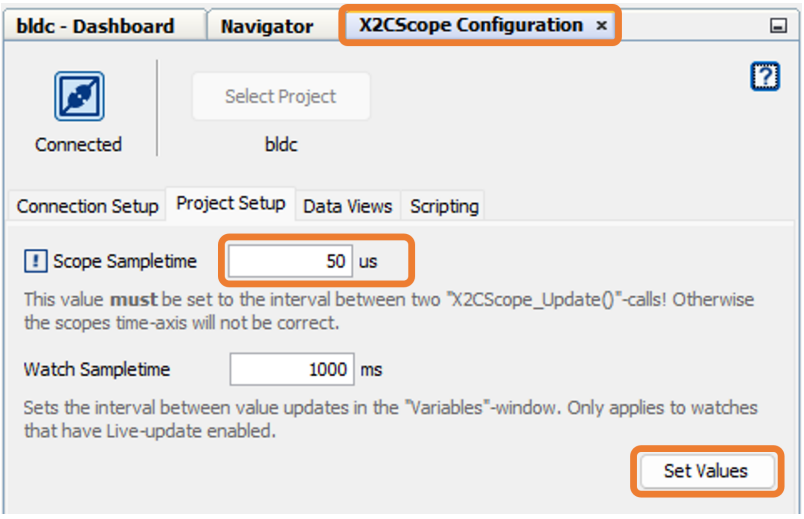


6. Once the **Serial port** is detected, click on **Disconnected** and turn to **Connected**, to establish serial communication between the Host PC and the board.

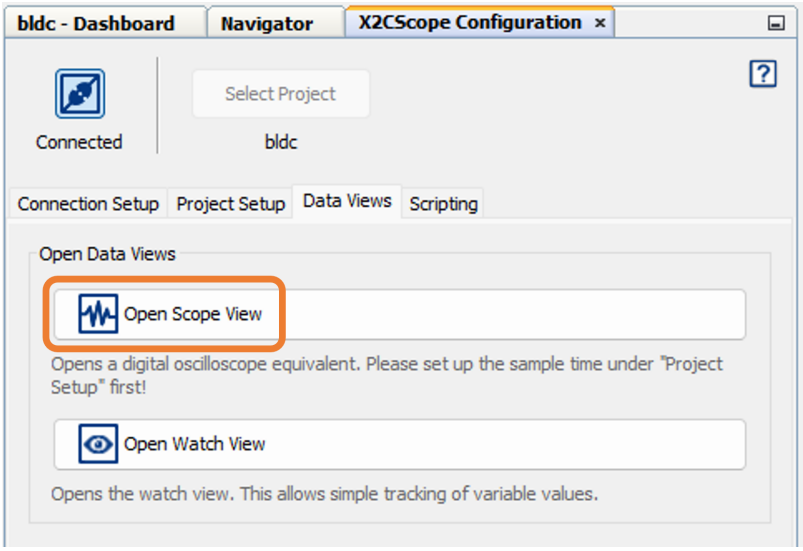


7. Open the **Project Setup** tab in the **X2CScope Configuration** window and,

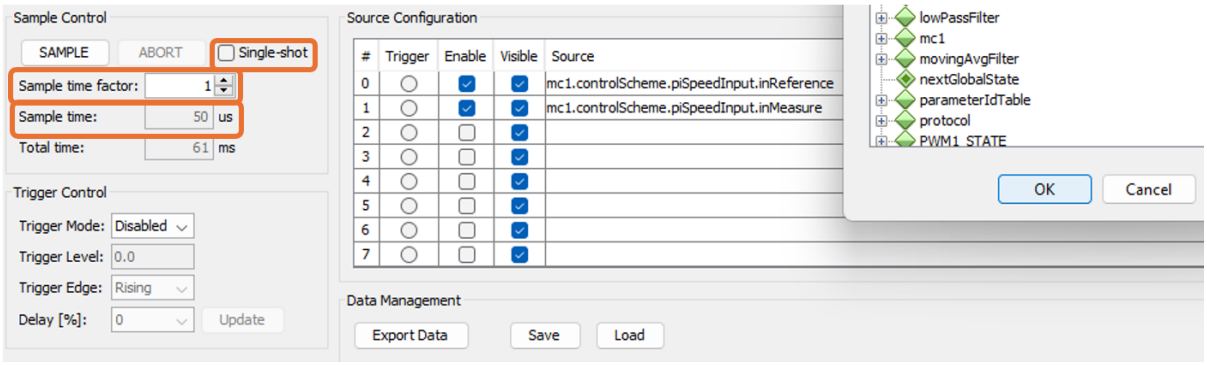
    - Set **Scope Sampletime** as the interval at which `X2CScopeUpdate()` is called. In this application, it is every `50µs`.
    - Then, click **Set Values** to save the configuration.



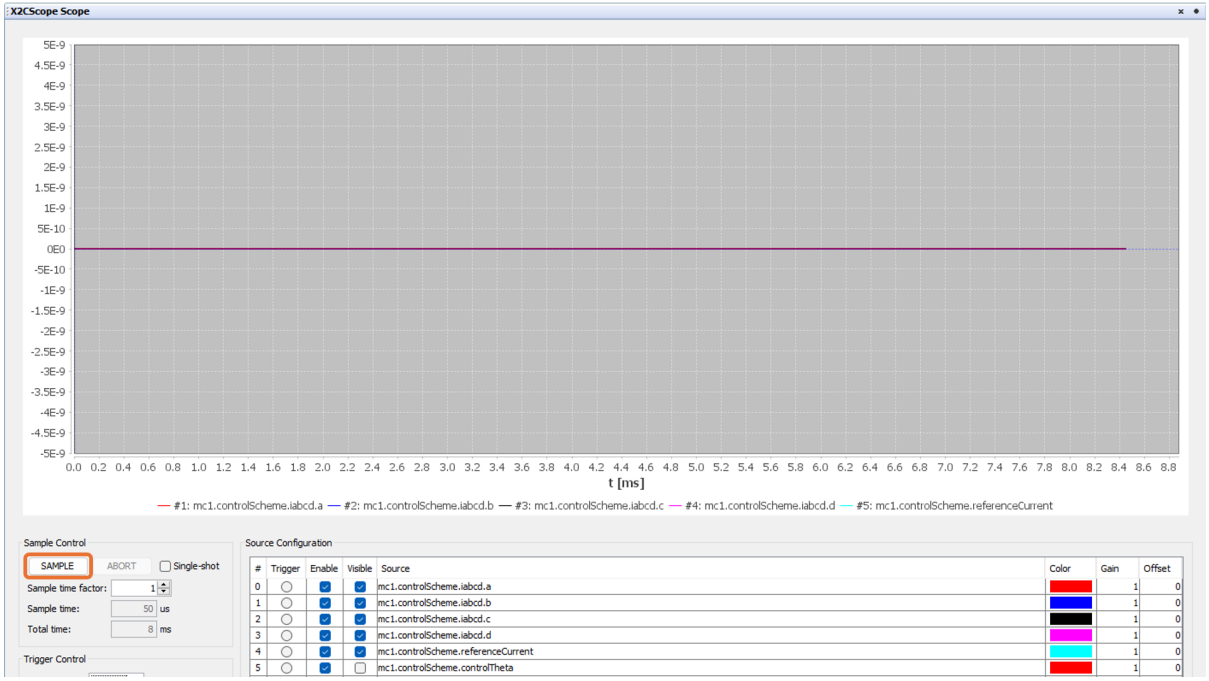
8. Click on **Open Scope View** (in the **Data Views** tab of the **X2CScope Configuration** Window); this opens **Scope Window.**



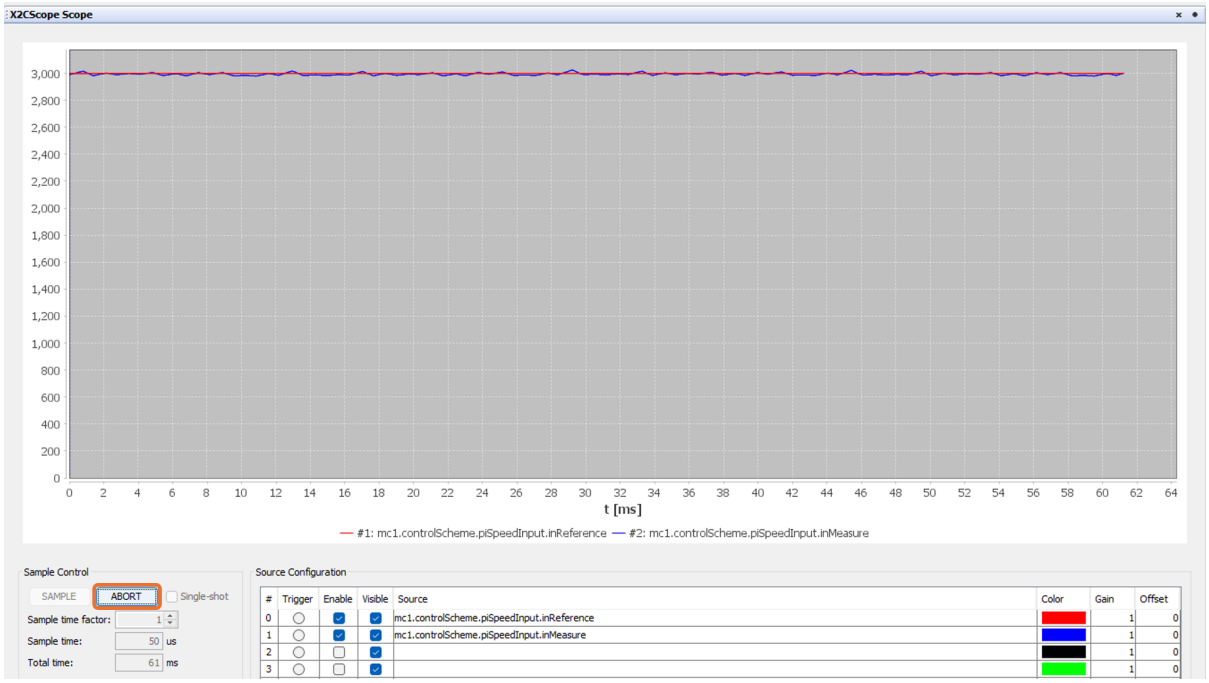
9. In the **Scope Window**, select the variables that must be watched. To do this, click on the **Source** against each channel, and a window **Select Variables** opens on the screen. From the available list, the required variable can be chosen. Ensure checkboxes **Enable** and **Visible** are checked for the variables to be plotted. To view data plots continuously, uncheck **Single-shot.** When **Single-shot** is checked, it captures the data once and stops. The **Sample time factor** value multiplied by **Sample time** decides the time difference between any two consecutive data points on the plot.



10. Click on **SAMPLE**, then the X2C-Scope window plots variables in real-time, which updates automatically.



11. Click on **ABORT** to stop.



6. REFERENCES:

For additional information, refer following documents or links.

1. AN957 Application Note "Sensored BLDC Motor Control Using dsPIC30F2010"
2. MCLV-48V-300W Development Board User's Guide (DS50003297)
3. dsPIC33CK256MP508 Motor Control Dual In-Line Module (DIM) Information Sheet (DS50003063)
4. dsPIC33CK256MP508 Family datasheet (DS70005349)
5. Family Reference Manuals (FRM) of dsPIC33CK256MP508 family
6. MPLAB® X IDE User's Guide (DS50002027) or MPLAB® X IDE help
7. MPLAB® X IDE installation
8. MPLAB® XC-DSC Compiler installation
9. Installation and setup of X2Cscope plugin for MPLAB X

10. Microchip Packs Repository