
Sensorless BLDC Motor Control for AVR® Microcontrollers

Introduction

Authors: Alexandru Zîrnea, Leona Pop, Microchip Technology Inc.

This application note describes a sensorless Brushless Direct Current (BLDC) motor control method using a Plug-In Module (PIM) based on the AVR128DA48 or AVR128DB48 microcontrollers from the AVR® DA and AVR DB families.

The first part of the document focuses on the motor control theory from which the software implementation is derived. It describes the sensed and sensorless control methods, details about trapezoidal commutation, and other important information about how to drive the motor.

The second part is dedicated to the software and hardware implementation of the system.

The example application uses the AVR128DA48 or the AVR128DB48-based PIM connected to a dsPICDEM™ MCLV-2 Development Board (Motor Control Low-Voltage) (MCLV-2). The MCLV-2 board offers a potentiometer for adjusting the speed of rotation, while the input protection and Back Electromotive Force (BEMF) circuitry are found on the PIM.

The system is capable of regenerative braking, which is inherent to the driving method used.

The driving waveforms for the motor are generated independently from the core, using the hardware capabilities of the microcontroller's peripherals.

Additionally, the application employs several protection mechanisms such as stall detection, false input detection, and overcurrent protection in both directions.

The system can drive a wide range of BLDC motors with an adequate tuning of the firmware. Guidelines for tuning the parameters needed for different types of motors are outlined at the end of the document.

BLDC motors are used in a wide range of applications such as cordless power tools, computers, hard drives, multimedia equipment, cooling fans in small electronic equipment, electric vehicles and are a growing market for the electric future. The presented solution creates a cost-effective motor driver by using an 8-bit microcontroller when traditionally, only 16-bit microcontrollers or more were used in creating this device.



View the Motor Control PIM Code on GitHub

[Click to browse repository](#)

Table of Contents

Introduction.....	1
1. Overview.....	3
2. Relevant Devices.....	4
3. Motor Control Theory.....	5
4. Firmware Implementation.....	14
5. Plug-In Module (PIM) Board Description.....	24
6. Tuning.....	34
7. Conclusion.....	37
8. References.....	38
9. Revision History.....	39
The Microchip Website.....	40
Product Change Notification Service.....	40
Customer Support.....	40
Microchip Devices Code Protection Feature.....	40
Legal Notice.....	41
Trademarks.....	41
Quality Management System.....	42
Worldwide Sales and Service.....	43

1. Overview

This application note presents the motor control theory needed to drive a BLDC motor and contains a comparison between the sensed and sensorless motor control, a typical implementation of a power stage, modulation techniques as well as a detailed description of the custom PIM.

Moreover, the document covers a detailed presentation of bipolar switching, which is the focus of this example.

To obtain the position of the rotor, the zero-cross point detection method is covered, using a comparator with a variable voltage reference consisting of a virtual neutral of the motor.

The motor starts at an initial position after an alignment routine. This is done so that at start-up the motor follows the generated open-loop waveform.

At compile-time, the user can select the mode the motor must run in, open- or closed-loop.

A potentiometer on the MCLV-2 board provides a value read by the ADC that is used to compute a duty reference for the waveform timer, and this reading, along with the current, is done asynchronously from the motor logic drive of the software. This part runs in the main loop, while the motor control runs on interrupts to get the best response time possible.

At the end of this document, the user can find a brief introduction to tuning the parameters of the firmware to obtain the desired results.

This application example uses the following peripherals:

- Analog Comparator (AC)
- Analog-to-Digital Converter (ADC)
- I/O Pin Controller (PORT)
- Timer/Counter Type A (TCA)
- Timer/Counter Type B (TCB)
- Timer/Counter Type D (TCD)
- Custom Configurable Logic (CCL)
- Voltage Reference (V_{REF})
- Universal Synchronous Asynchronous Receiver Transmitter (USART)

Prerequisites:

1. Software:
 - Microchip Studio 7 Integrated Development Environment (IDE) with AVR DX Device Family Pack (DFP) 1.7.85 or newer)
2. Hardware:
 - AVR128DA48 Motor Control PIM or AVR128DB48 Motor Control PIM with internal op amp configuration board
 - [dsPICDEM™ MCLV-2 Development Board \(Motor Control Low-Voltage\)](#)
 - [AC300020 - 24V 3-Phase Brushless DC Motor](#)
 - [AC002013 - 24V Power Supply](#)
 - [MPLAB® PICKIT™ 4 In-Circuit Debugger](#)

The MCLV-2 Development Board is targeted to control either a brushless motor or a permanent magnet synchronous motor in sensed or sensorless operation. In this demo, it will be used in conjunction with the AVR128DA48 or AVR128DB48 Motor Control PIMs to drive a brushless motor in sensorless operation. The board offers a driver for the three phases, a way to measure feedback signals, and an on-board op amp. An additional configuration board is used with the AVR128DB48 PIM because the internal operational amplifier signals are routed differently.

This demo will use some other features of the MCLV-2 board like the programming interface and the serial interface to receive information about the working parameters of the motor. Additionally, there are LEDs, buttons and a potentiometer on the board that provide a user interface to show which PWM outputs are active, to set the direction of the motor and to start and stop it, and to change the speed of the motor.

2. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 2-1. AVR® DA Family Overview

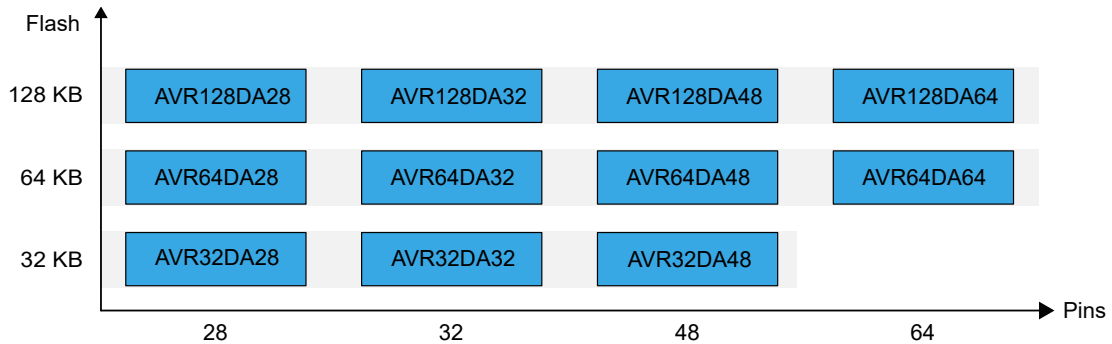
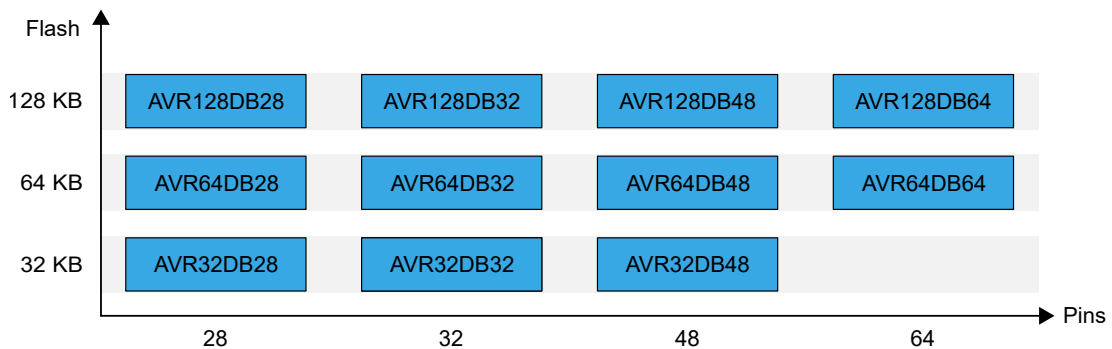


Figure 2-2. AVR® DB Family Overview



3. Motor Control Theory

This section explains the motor control concepts and how they apply in software and hardware implementations.

3.1 Sensored vs. Sensorless Control

BLDC motors are synchronous and compact, provide high efficiency and eliminate the need for any maintenance. They have a big advantage over DC brushed motors, being suitable for consumer and industrial applications that require very stringent standards.

DC brushed motors are mechanically commutated. They have brushes and a commutator that applies voltage to the coils according to the specific rotor position.

Unlike DC brushed motors, the commutation in BLDC motors is done depending on the position of the rotor, which has to be known.

To determine the position of the rotor, either sensor or sensorless algorithms can be used.

3.1.1 Sensored Control

One method is to use sensors on the outside of the stator which will react to the magnets on the rotor accordingly and give a signal that can be used to determine the position. Typically, three Hall-effect sensors will give a 60 degrees difference between two transitions.

The advantage of using sensors is that by knowing the state of each hall-effect sensor, the position of the rotor is known even at zero speed, on system power-up.

The main downside of using sensors is that they add cost and complexity to the motor, while also needing auxiliary power.

3.1.2 Sensorless Control

Another method is to use sensorless control, which only requires a few passive components on the control board.

This method works by sensing the voltage on a floating phase induced by the movement of the magnets in the rotor, called Back Electromotive Force (BEMF). BEMF gives the microcontroller data about the current position of the rotor.

As the voltage induced in one coil is proportional to the speed of the rotor, at very low speeds, the amplitude of the BEMF is very small, noisy and difficult to filter. As such, there is an open-loop starting procedure in which the rotor follows the magnetic field generated by the microcontroller asynchronously. The speed grows continuously until the BEMF has a detectable amplitude.

The upside of sensorless control is that any sensed motor can be driven in Sensorless mode and can be used as a fail-safe control method, where it is possible. It is also more reliable as the system does not depend on external inputs, other than the inherent BEMF of the motor.

Unlike sensed control, where the hall effect sensors provide a logical signal, without any noise, sensorless control needs signal filtering and conditioning due to motor construction, power supply noise and other factors.

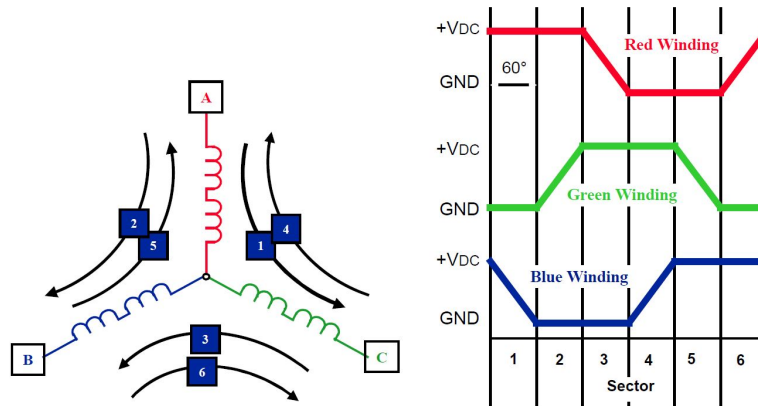
Because of the BEMF filtering, abrupt changes in motor speed cannot be detected precisely and may get the system out of lock. The sensed control does not suffer from this problem, as the Hall sensors provide the signal regardless of the motor operating conditions.

3.2 Six-Step (Trapezoidal) Commutation

The trapezoidal control consists of three phases (A, B, and C in this case), from which two phases are driven, and the third is left floating, thus, giving the ability to detect the zero-cross of the BEMF generated by the moving rotor.

Each step or sector represents 60 degrees from a total of 360 degrees or one full electrical revolution.

Figure 3-1. Ideal Trapezoidal Waveform



Step commutation is done as follows:

Step 1: <ul style="list-style-type: none"> • Red winding is driven positive • Green winding is driven negative • Blue winding is not driven 	Step 4 <ul style="list-style-type: none"> • Green winding is driven positive • Red winding is driven negative • Blue winding is not driven
Step 2: <ul style="list-style-type: none"> • Red winding is driven positive • Blue winding is driven negative • Green winding is not driven 	Step 5: <ul style="list-style-type: none"> • Blue winding is driven positive • Red winding is driven negative • Green winding is not driven
Step 3: <ul style="list-style-type: none"> • Green winding is driven positive • Blue winding is driven negative • Red winding is not driven 	Step 6: <ul style="list-style-type: none"> • Blue winding is driven positive • Green winding is driven negative • Red winding is not driven

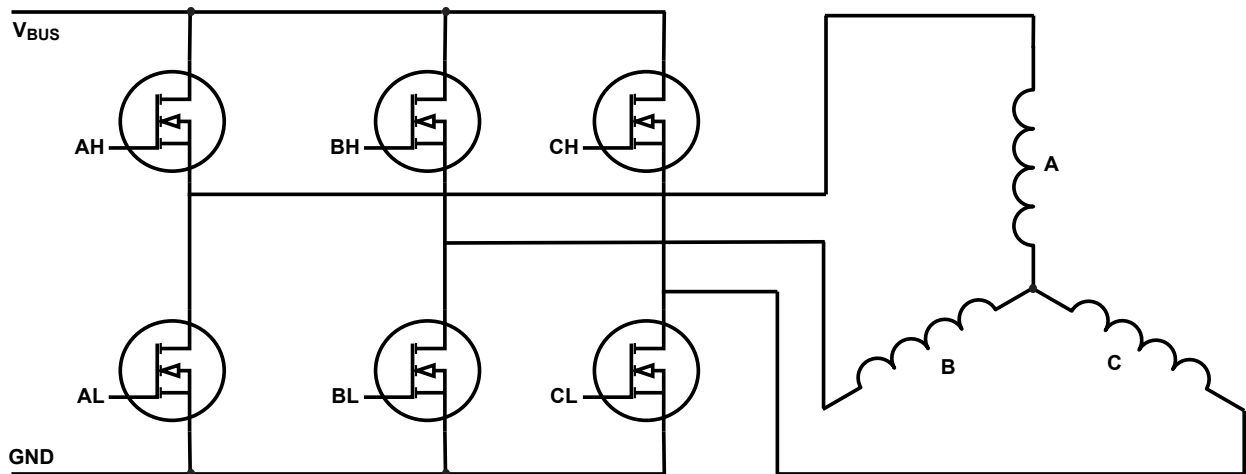
At constant speed, or for small speed variations, the period between two commutations is equal. This is the estimation for controlling the motor in Closed Loop.

This implementation uses trapezoidal commutation due to its simplicity and ability to be very easily implemented in 8-bit microcontrollers, as it needs little memory resources and processing power.

3.3 Power Stage

The power stage is comprised of an inverter, which consists of three half-bridges that can either tie each phase to the supply voltage or ground. The switching element is typically a MOSFET, for low-voltage applications, or an IGBT, for high-voltage applications.

Figure 3-2. Typical Power Stage Schematic



The AH, BH and CH represent the high-side command signals from the microcontroller, while AL, BL and CL represent the low-side signals.

In addition to the power transistors, MOSFET/IGBT drivers are used to ensure good rise and fall times and provide the gate voltage needed for the high-side and low-side transistors.

A combinational logic circuit is used to avoid shoot-through by adding dead time between the high-side and low-side command signals.

This dead time ensures the MOSFETs or IGBTs on the same branch are completely turned off and do not conduct current at the same time.

3.4 Modulation Techniques

To control the speed of a BLDC motor, the voltage level applied to the motor's phases needs to be adjusted. This can be easily done in the same manner as in DC brushed motors, and that is to modulate the voltage with the help of Pulse-Width Modulation (PWM). Thus, the motor speed is directly proportional to the duty cycle of the PWM signal.

The following sections present two switching methods along with their advantages and disadvantages.

3.4.1 Unipolar Switching

Unipolar switching requires two transistors to be energized during one trapezoidal step.

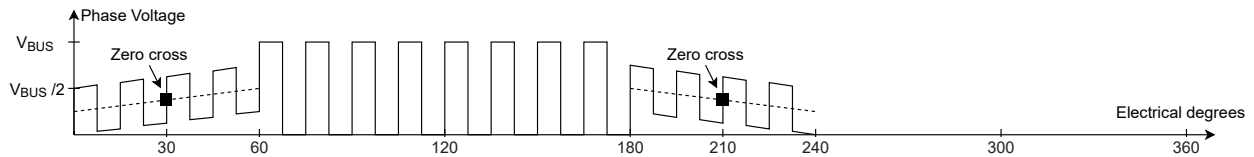
The low-side transistor is conducting all the time, while the upper one is receiving a PWM signal. In most applications, the high-side transistor command is handled by a MOSFET driver. The MOSFET driver has a bootstrap capacitor that charges when the low side is conducting. If the method used applies the PWM on the low side, the bootstrap capacitor will be charged depending on the duty cycle, resulting in a low charge at a low percentage of the duty cycle, which will result in an increase of the $R_{DS(on)}$ of the transistors and will decrease the overall efficiency, or under-voltage lock-out of the driver IC chip.

In sensorless control, this method poses a few challenges:

- During the time in which the BEMF is exposed, the waveform will be chopped due to magnetic coupling between all of the coils, as the PWM is induced in the floating phase. Thus, a synchronizing method is needed to 'sample' the BEMF at specific times. This sampling time is ideally at the middle of a high pulse, where the BEMF stabilizes and does not vary.
- Any noise that overlays the waveform will be hard to filter and will trigger the zero-cross at the wrong time, which will decrease the efficiency and increase both acoustic and electrical noise. In this method, the traditional RC filter method will prove inefficient at low speeds as it will require high values and introduce a high amount of delay at high speeds.

A typical waveform using unipolar switching can be seen in [Figure 3-3](#).

Figure 3-3. Unipolar Switching Phase Voltage (50% Duty Cycle)



This is a classic scenario used in most simple designs.

3.4.2 Bipolar Switching

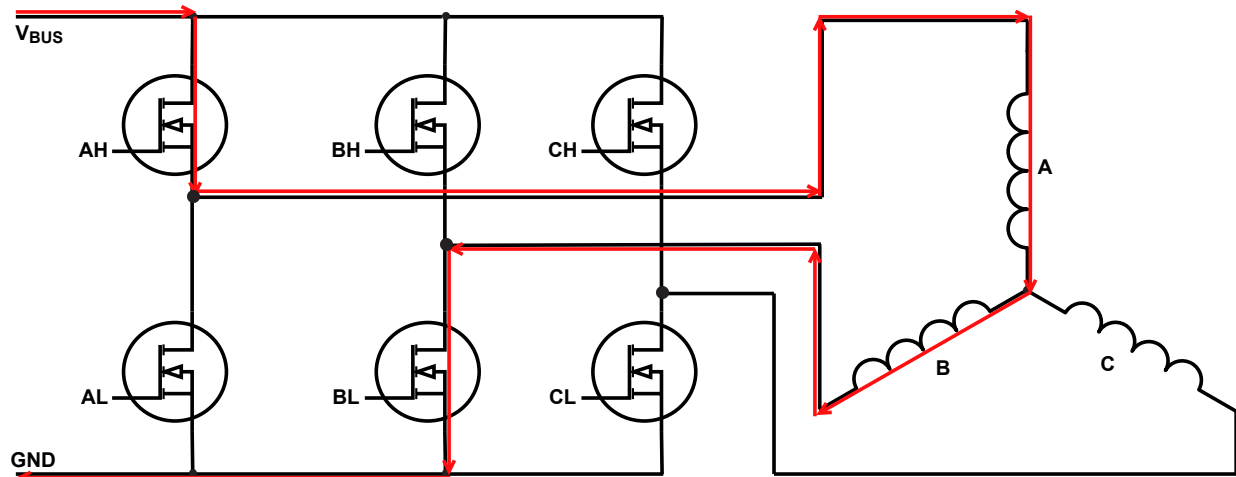
Unlike unipolar switching, bipolar switching requires four transistors to be energized during one trapezoidal step.

In contrast to the unipolar switching, in this case, the other two transistors to be energized are complementary to those characteristic to each step.

This behavior is expressed in the following situation:

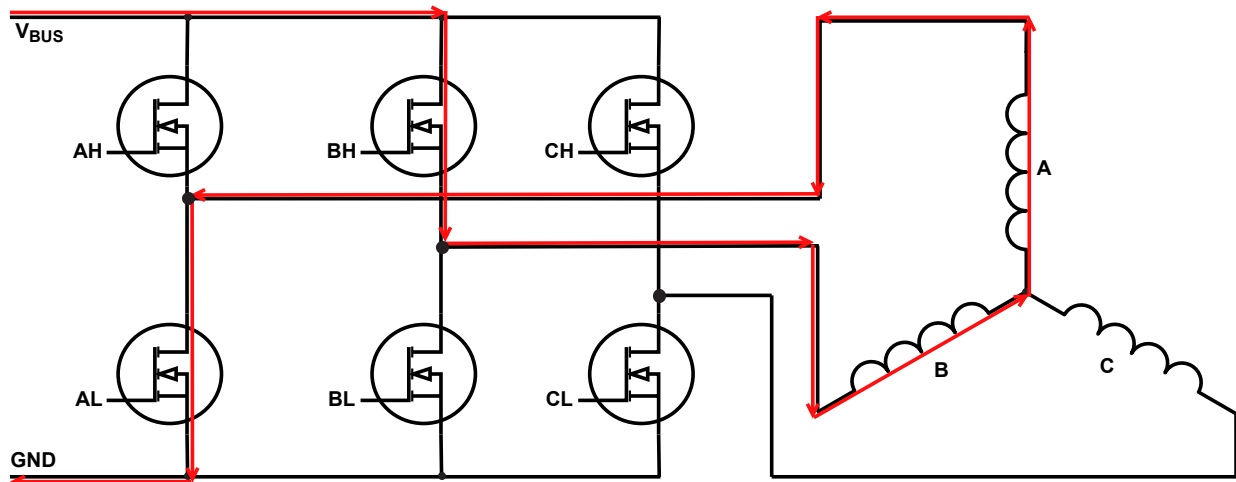
The characteristic state is AH-BL, which means that phase A is tied HIGH and phase B is tied LOW. This is done by energizing the high-side transistor of the phase A half-bridge, and the low-side transistor of phase B.

Figure 3-4. Current Flow During AH-BL State



The complementary state of AH-BL will be AL-BH and the current will flow in this way. See [Figure 3-5](#) below.

Figure 3-5. Current Flow During AL-BH State



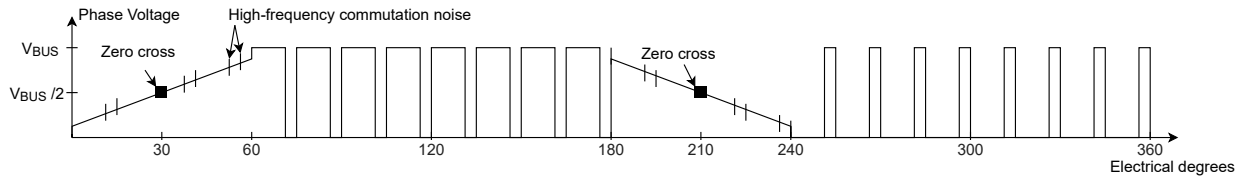
Considering zero speed at 0% duty cycle for unipolar drive, in this case, the zero speed is at 50% duty cycle, where the complementary state time is equal to the characteristic state.

This method has the advantage of a very clean BEMF because all the voltage induced in the floating phase is canceled by the complementary state, thus, obtaining a waveform that will contain only high-frequency spikes, which can be filtered using an RC filter with a very small time constant.

The main disadvantage of this method is that the motor emits more electromagnetic and acoustic noise and increases the overall system loss due to another transistor pair needed to switch.

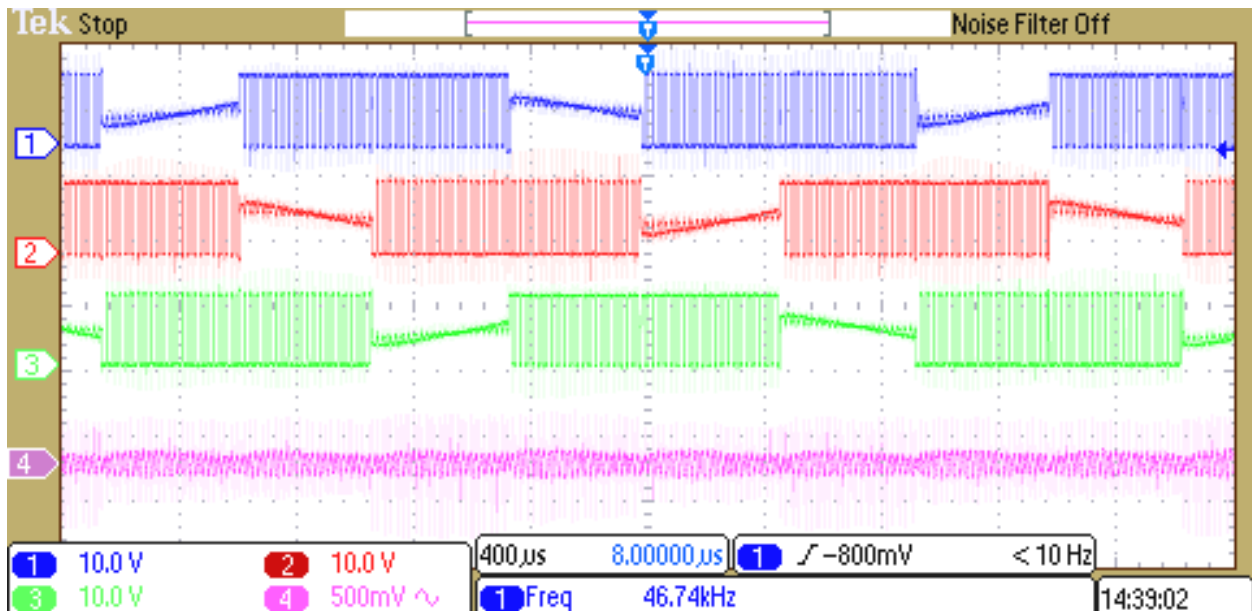
The voltage on one phase can be seen in [Figure 3-6](#).

Figure 3-6. Bipolar Switching Phase Voltage (75% Duty Cycle)



The high-frequency spikes appear at the transition between the characteristic state and the complementary state. These can be observed in low-inductance motors where the amplitude of the spikes gets higher near the commutation zones.

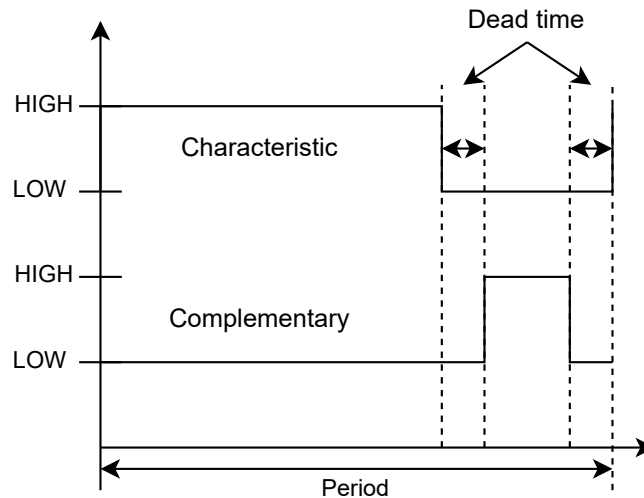
Figure 3-7. BEMF Noise on Low-Inductance Motor



In [Figure 3-7](#), the first three waveforms represent the PWM signals for the three phases of the motor while the fourth waveform represents the sensed current through the transistors.

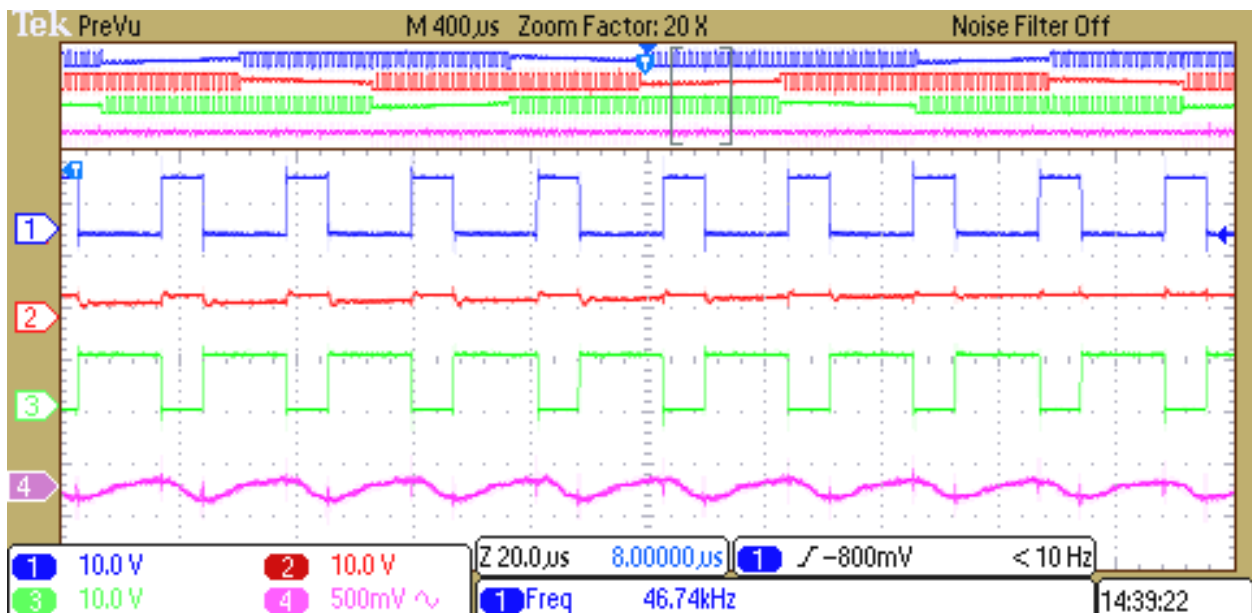
To avoid shoot-through, dead time will be added between the command signals of the drivers.

Figure 3-8. Dead-Time Insertion



A dead time of approximately 42 ns was added (one clock cycle of the PWM timer), and the result can be seen in [Figure 3-9](#). However, it is recommended to have a higher dead time if the power MOSFETs/IGBTs have high gate capacitance.

Figure 3-9. Motor Waveform Close-Up



In [Figure 3-9](#), the first three waveforms represent the PWM signal for the three phases while the fourth waveform is the sensed current through the transistors. The small spikes in the current waveform correspond to the transistors switching and represent a small amount of shoot-through. If no dead time was added, then the current would rise considerably and the transistors will warm up and burn up.

Switching Noise

Similar to unipolar switching, bipolar switching suffers from a specific problem: During commutation, when the motor has a certain amount of load applied to it, there will be a demagnetization sequence that will influence the detection of the zero-cross and potentially give a false input, which, in turn, will get the motor out of sync.

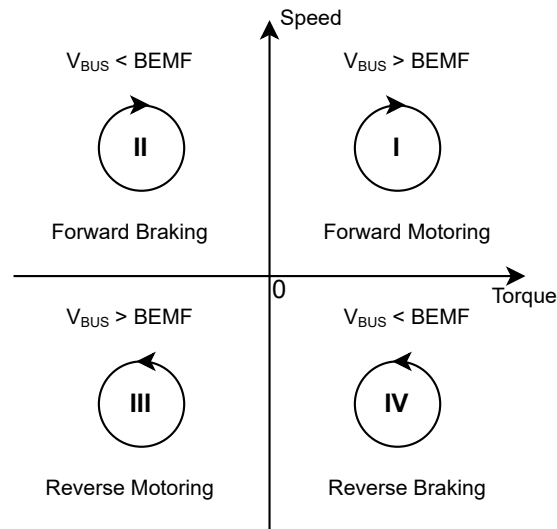
This can be particularly challenging as the demagnetization sequence can almost get to 30 electrical degrees, which will, in turn, limit the area in which the zero cross point can be detected, and potentially suppress it.

The solution to this behavior is to add a blanking time, during which the zero cross is ignored completely. This parameter will be dynamically set in the control algorithm and self-adapt during the entire range of RPM.

Regenerative Braking

A brushless motor can run in four different regions. These regions are called 'quadrants' and are described in [Figure 3-10](#).

Figure 3-10. Four Quadrant Operation



For this application note, only quadrant numbers one and two are of interest and the motor can run in either one, without the specific intervention of the control algorithm.

In quadrant one, the motor draws current from the bus and is free-running or has a load applied to it. The measured speed of the rotor is equal to or less than the set speed.

On the other side, in quadrant number two, the set speed of the motor is lower than the measured speed due to system inertia or other external forces.

The two-quadrant operation is directly related to the duty cycle, which will impose a specific RPM, given in the equation below.

$RPM = K_v \times U$, where K_v is a parameter of the motor and U is the voltage applied to the coils.

The following concepts: RPM_{SET} and $RPM_{MEASURED}$ provide information about the current RPM of the motor and the set RPM at any given time.

It is, thus, necessary to express the difference in the RPM with the formula:

$$\Delta RPM = RPM_{SET} - RPM_{MEASURED}$$

If ΔRPM is greater than zero, then the motor is in quadrant one and draws current from the power supply.

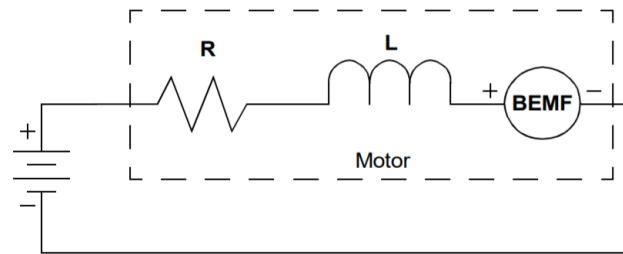
Otherwise, if ΔRPM is less than zero, the motor is in quadrant two and pushes current to the power supply.

This is a rough approximation of how the motor behaves and does not account for variables such as friction losses or load.

In quadrant two, the system behaves like a boost converter and the maximum current that can be generated depends on the parameters of the motor, as well as the ΔRPM mentioned earlier and the ESR of the battery.

An equivalent circuit can be obtained for the motor:

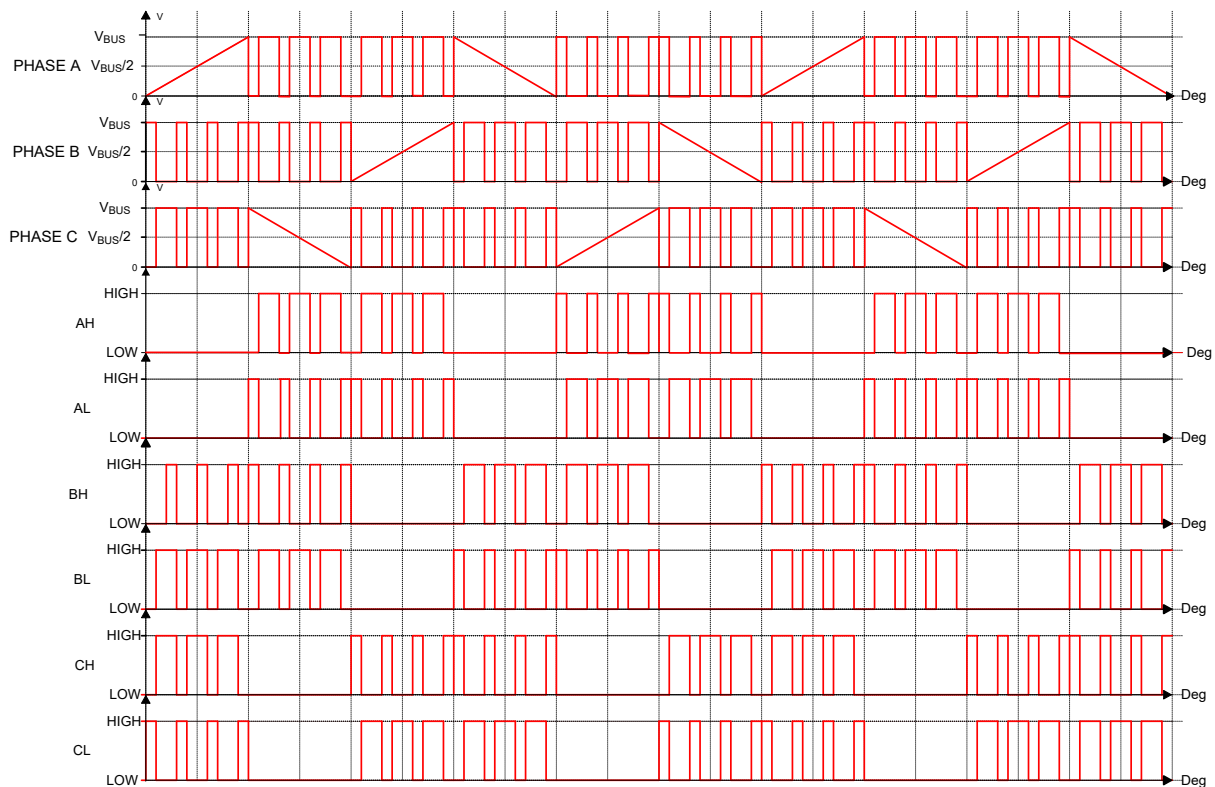
Figure 3-11. Equivalent Circuit of a Motor



The coil resistance and inductance are static throughout the RPM range but the BEMF is directly proportional to the RPM: $BEMF = RPM/K_v$. The BEMF amplitude is equal to the supply voltage multiplied by the PWM duty factor (for bipolar operation, 50% duty means 0V). If the motor does not have any load, the BEMF amplitude will approach the supply voltage at the full duty cycle.

To protect the system and the power supply from an Overvoltage condition, additional circuitry is needed to monitor the voltage on the bus.

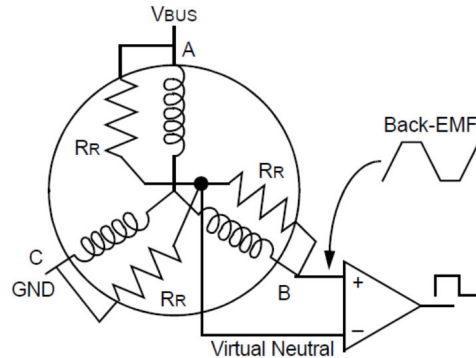
Figure 3-12. Phase Voltages and Command Signals of Bipolar Switching



3.5 Zero-Cross Point Detection

In this implementation, the method employed to measure the zero-cross point is by using a comparator with a variable reference consisting of the sum of all the phase voltages.

Figure 3-13. Zero-Cross Point Detection Using Comparator

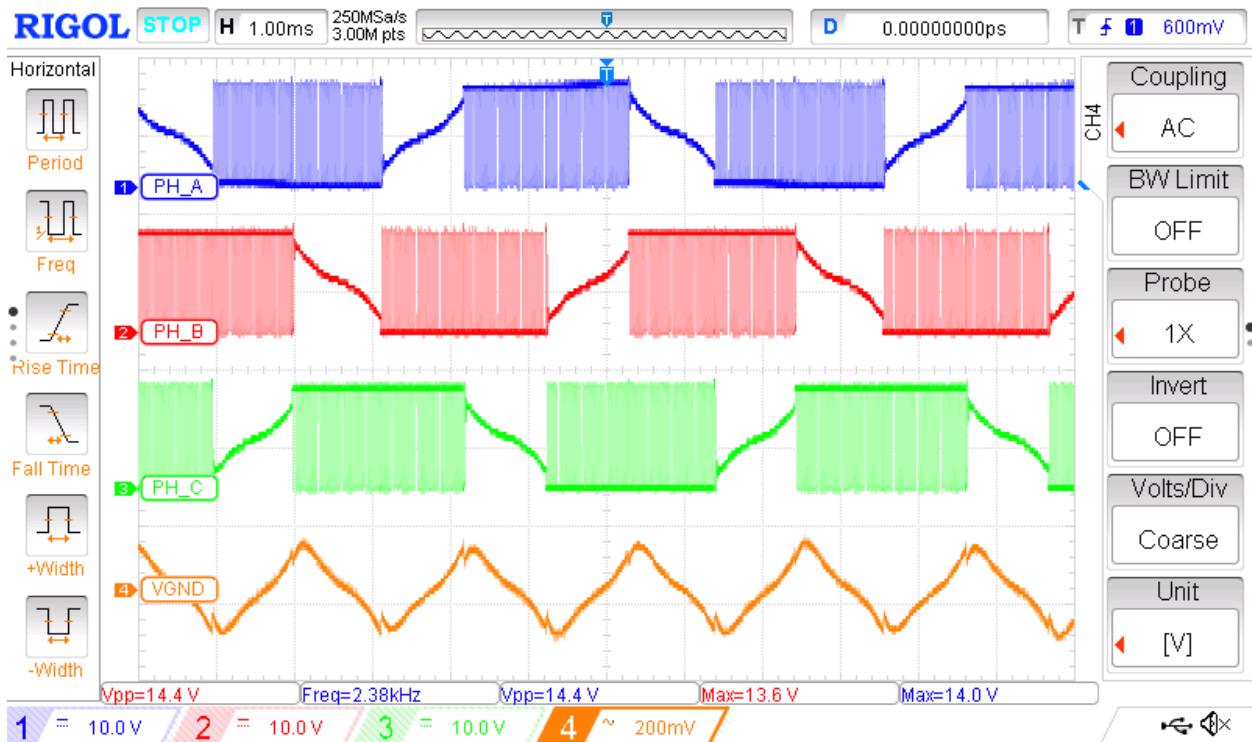


The neutral voltage can be expressed as $V_n = \frac{BEMF A + BEMF B + BEMF C}{3}$, where V_n is the virtual neutral voltage, $BEMF A$ is the BEMF voltage in phase A, $BEMF B$ is the BEMF in phase B, and $BEMF C$ is the voltage in phase C.

Due to the asymmetry in the coil impedance, the virtual neutral signal will not be constant, but one that will vary based on the driving cycle. To avoid any phase shifts in the commutation point, the virtual neutral point must not be heavily filtered. Only high-frequency noise must be suppressed with a low RC time constant low-pass filters.

The irregularity of the virtual ground can be seen in Figure 3-14, where the first three waveforms represent the PWM signal for each of the three phases and the fourth waveform is the virtual ground.

Figure 3-14. Virtual Ground Waveform



4. Firmware Implementation

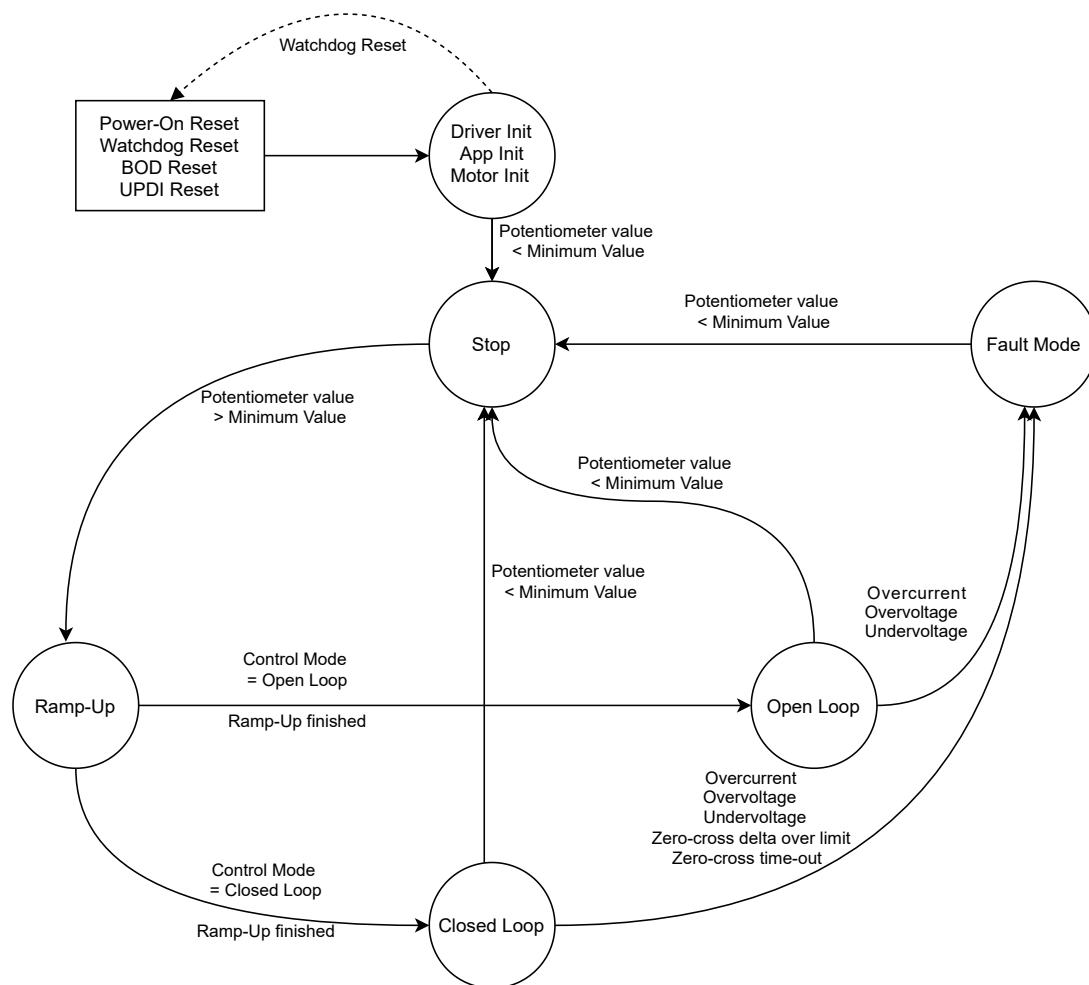
This section presents the use of the internal AVR peripherals to achieve a specific task and describes how the firmware works as a motor control system.

4.1 System Overview

The System State Machine

The firmware is based on a state machine that controls how the system behaves. It provides Fault handling, Reset information, and enables debugging by utilizing error messages transmitted via UART.

Figure 4-1. System Control State Machine



On system power-up, the System Control state is set to Driver Init, App Init state. This is the default state that the system has after any Reset. To identify the Reset cause, a message is printed to notify the user about the specific event. The potentiometer value is checked beforehand to make sure that the system does not start right up (for example, if the system had an external Reset while the motor was at full speed, it would not have started

right up protecting the motor and power stage). If the condition is not met, the watchdog will trigger and reset the microcontroller.

After this, the motor enters the Stop state. If the potentiometer value increases above a set threshold, then the rotor will be aligned, and the system state will be set to Ramp-Up. Here, the motor is accelerated to the defined speed in software, without knowing the position of the rotor.

Upon finishing the ramp-up sequence, the system goes in the Defined Control state, which can be either Open Loop or Closed Loop.

When in Open Loop, the motor runs and the microcontroller does not have any information about its rotor position. The speed of the motor can be adjusted through the potentiometer. If its value is smaller than a set threshold, the motor stops and the system goes into the Stop state. If an overcurrent is detected, the system goes into the Error Handling state.

When in Closed Loop, the running of the motor is based on the feedback received by the microcontroller about the rotor position. The speed of the motor is constant and only the duty cycle can be adjusted through the potentiometer. If its value is smaller than a set threshold, the motor stops and the system goes into the Stop state.

Besides overcurrent, zero-cross delta over limit, zero-cross time-out, undervoltage and overvoltage cause the system to enter the Error Handling state.

If the system gets into Error state, then the only way to reset it is to get the potentiometer under a set threshold. Then the system gets back to the initial Stop state.

Overview of the Utilized Device Modules and Signals

Figure 4-2 depicts the relationship between the hardware, microcontroller and software modules. The term 'control' is used to signal any modifications on the peripheral's registers.

Figure 4-2. Overview of the System Modules

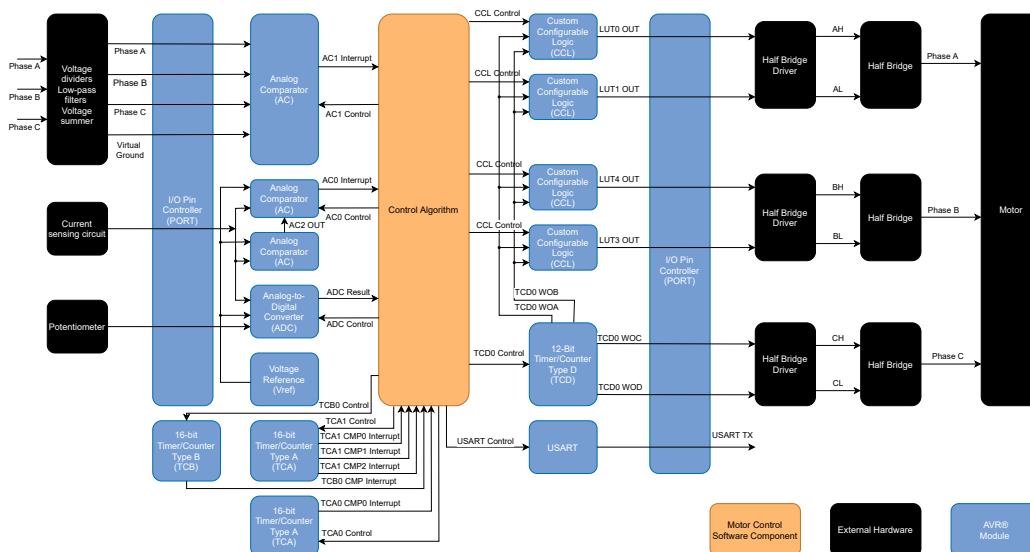
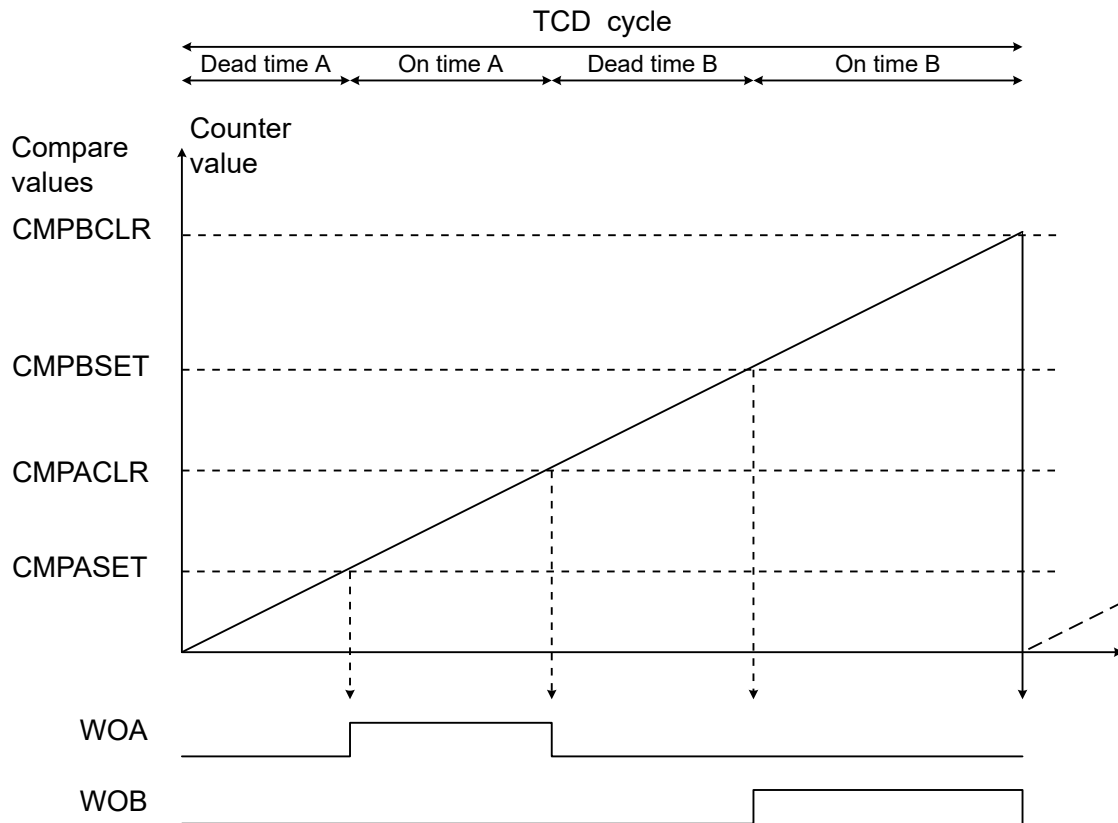


Figure 4-3. TCD One Ramp Mode Waveform



In One Ramp mode, the TCD cycle period is: $T_{TCD_cycle} = \frac{(CMPBCLR + 1)}{f_{CLK_TCD_CNT}}$

For example, using an input clock of 24 MHz, a prescaler of 1:1, and a CMPBCLR value of 512, will give a period of 21.375 μ s, or a frequency of approximately 46.783 kHz.

The dead time is set in software during external changes on the timer's duty cycle.

For this application, the operation region is above 50% duty cycle to operate in quadrant one and two.

PWM Signal Routing

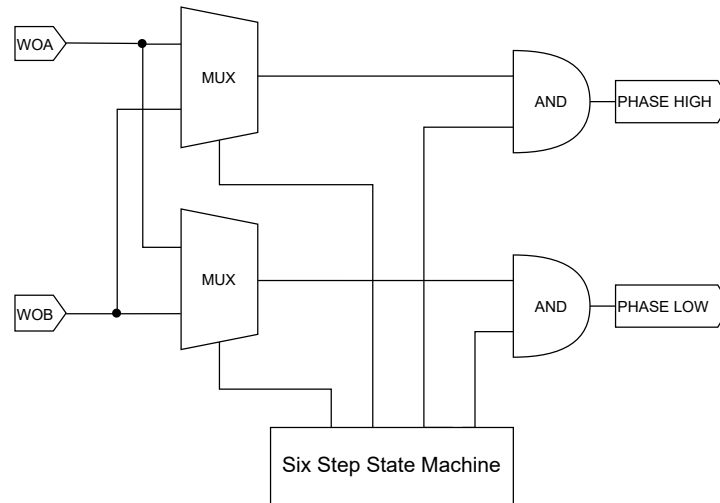
For six steps trapezoidal control, three half-bridge drivers must be controlled by a microcontroller. The TCD has only four waveform outputs that allow direct control of only two half-bridge drivers. The other two phases are created with the help of Custom Configurable Logic (CCL).

The WOC and WOD outputs of the TCD peripheral are available to the physical pins on the microcontroller and they are used to drive phase C.

The WOA and WOB outputs of the TCD peripheral are used to control the other two phases. With the help of CCL logic, the WOA and WOB signals are available on two different pairs of pins.

The equivalent logic for one phase, based on the behavior of the system, is configured through the TRUTH registers:

Figure 4-4. Equivalent Logic Routing Schematic



The CCL logic implements smart routing of PWM signals to the half-bridge drivers, allowing control of two motor phases using only one complimentary PWM pair (WOA and WOB), as described in the use cases below.

If the characteristic state of the phase is LOW, WOA is routed to the low-side transistor, while the WOB will be routed to the high-side transistor.

If the characteristic state of the phase is HIGH, the WOA is routed to the high-side transistor and the WOB to the low-side transistor.

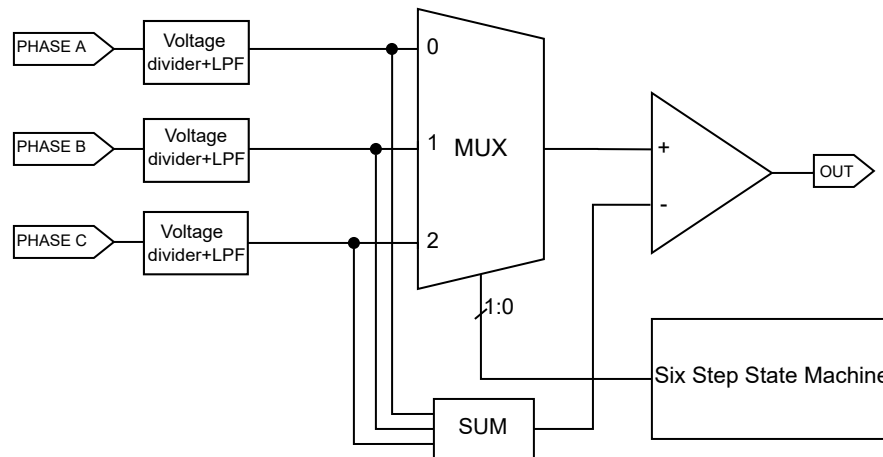
4.3 Zero-Cross Detection

The zero-cross point must be detected with high precision to ensure an accurate switching point. The amplitude of the voltage present on each phase is reduced using a voltage divider and is then filtered with a low-pass RC filter.

One internal comparator (AC1) is used to compare the filtered voltages with the virtual neutral of the motor, which is provided as a sum of all the phase voltages.

The AC1 comparator used here has multiple inverting and non-inverting inputs selectable with the help of a built-in multiplexer. This feature is used to switch the AC1 inputs function by a state machine, allowing the use of one AC for all three phases and thus, reducing the number of peripherals required. Figure 4-5 describes the equivalent circuit used to detect the zero-cross point.

Figure 4-5. Zero-Cross Detection Equivalent Circuit



4.4 Zero-Cross Filtering

In some cases, the zero-cross point might be detected earlier or later than the ideal case due to noise or other factors such as motor particularities.

A filtering method will be used to avoid an unwanted behavior that can get the motor out of lock.

The basic constraints of this filter need to follow a few simple rules:

- The value of the filter must not be too aggressive in a way that would affect the reaction time of the motor in the case of a moderately fast speed change and not too soft, as it defeats its intended purpose
- The filter must not be computationally intensive, as it may take a long time to obtain the desired result and delay the commutation point
- A minimum pass-through delay is required to obtain the best results

Based on the above requirements, a first-order IIR filter is chosen under the form of a moving average low-pass filter using the formula $y(n) = \frac{y(n-1) \times (a-1)}{a} + \frac{x(n)}{a}$, where y is the output of the filter, x is the input, and a is the filter coefficient.

```
timerValue = (uint32_t)((uint32_t)previous_zero_cross_time * (MOTOR_DIVISION_FACTOR - 1) +
(uint32_t)current_zero_cross_time) / MOTOR_DIVISION_FACTOR;
```

Ideally, for faster computation, the filter coefficient (`MOTOR_DIVISION_FACTOR`) must be a power of two, as it can be implemented using bit shifting with specific core instructions.

The input and output values are unsigned 16-bit values, thus, to prevent an overflow during the multiplication of $y(n-1)$, as well as the total sum, the operations are done using casting to 32-bit unsigned values.

For the scope of this application, a value of 4 for the filter coefficient was considered the best fit.

4.5 Thirty-Degree Timing

The thirty degree wait time is defined as the time between the commutation point and the next zero-cross point. In trapezoidal control, when the zero cross is found, the system must wait for a thirty-degree electrical angle for the commutation to occur.

Any alteration of this time is called advance or retard, when the commutation is done earlier or later than it is supposed to.

Advance is done to achieve a greater torque during acceleration and, similarly, the braking force is increased during the retard.

A fixed advance value is implemented by subtracting the value from the newly computed thirty-degree time to compensate for any pass-through delay and add an advance time to the commutation point.

This is done using the following line of code: `tempValue = timerValue - MOTOR_ADVANCE_TIME;`

4.6 BEMF Commutation Noise Blanking

In trapezoidal control, when the system moves to the next step, the BEMF of the floating phase specific to the new step is affected by the commutation noise. It is then necessary to avoid capturing that sequence, as it can trigger a false zero-cross point.

This is called blanking time and it must be set based on the speed of the motor. For example, if a blanking time is good for low speeds in a high-speed scenario, the blanking time might be more than enough and the zero cross is not captured. It is calculated by using half of the previous value utilized in the thirty-degree timing.

4.7 Handling Time-Related Events

The driving of the motor is highly time-dependent. Two instances of the TCA peripheral and one instance of the TCB peripheral are used in this system to handle the time-related events.

The first instance of the TCA peripheral (TCA0) controls the speed of the motor when the system is in Ramp-Up/Open Loop mode. It triggers a periodical interrupt at an interval determined by the value from the CMP0 compare register.

By decreasing the value from the CMP0 register, the interrupt period decreases, and the motor is accelerated. If the decreasing of the register stops, the motor is commutated with a fixed period, and the Control mode is switched to Open Loop mode.

Upon reaching the interrupt, the algorithm moves to the next trapezoidal step and, after that, it triggers it.

This is done using the following lines of code:

```
tmr_ramp_up_compare_value -= MOTOR_TIMER_RAMP_UP_DECREMENT;
TCA0_setCMP0Value(tmr_ramp_up_compare_value);
```

```
#define MOTOR_TIMER_RAMP_UP_DECREMENT    800
```

The second instance of the TCA peripheral (TCA1) is used for driving/handling time-related motor events during each step.

During one trapezoidal step, the counter serves four purposes:

- It handles the blanking time
- It counts the time from the commutation point to the zero-cross point
- It counts the new calculated time from the zero-cross point to the new commutation point
- It handles the stall detection (zero-cross time-out)

The counter has three compare channels, each of them configured to trigger a specific interrupt.

The method of operation is described below:

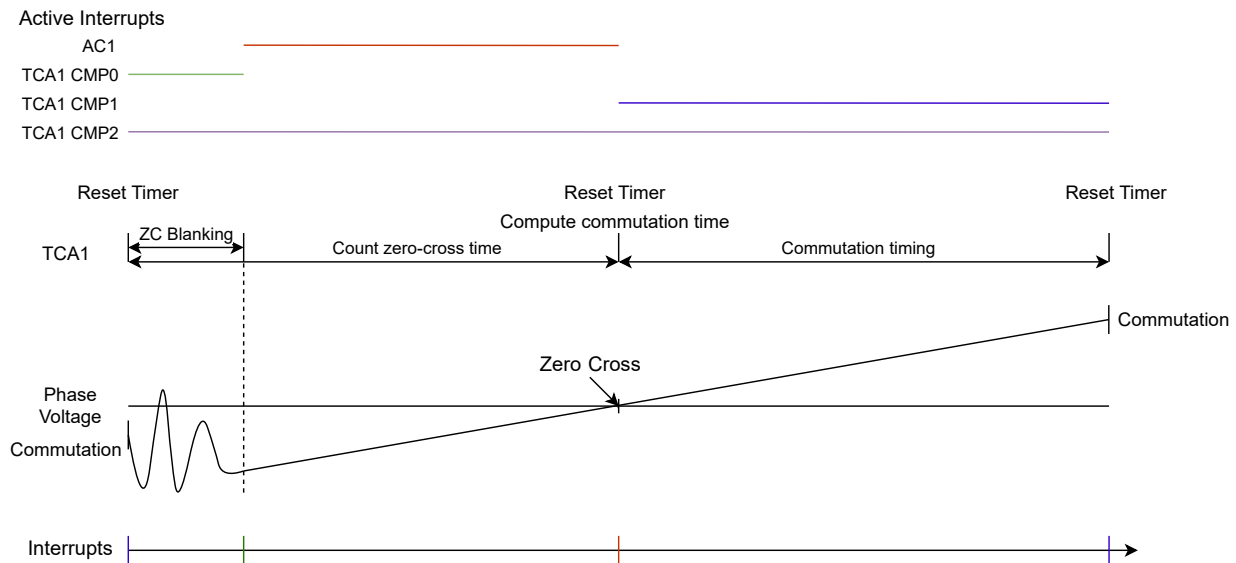
After a new commutation, the CNT register value is cleared, and the CMP0 interrupt is enabled.

Upon reaching the CMP0 interrupt, the AC1 interrupt is enabled, and the CMP0 interrupt is disabled.

If the zero cross is found within the CMP2 time interval, then the timer is stopped, the CNT register value is processed with the zero-cross filter and set to the CMP1 register. The counter register is cleared once again and the CMP1 interrupt is enabled.

When the timer reaches the CMP1 value, it triggers an interrupt, commutates to the next step, disables the CMP1 interrupt, and the cycle is repeated. A detailed, time-domain representation of the functionality of this block can be seen in [Figure 4-6](#).

Figure 4-6. TCA1 Time-Domain Representation



The TCB0 timer/counter acts like a very basic scheduler to run tasks that are non-time critical. It uses a fixed value for a compare register which triggers an interrupt and sets the `SCHEDULER_mainLoopFlag` checked in the main loop.

4.8 Overcurrent Protection

To protect the motor, the power stage, and the power supply, an overcurrent protection method is needed. This is accomplished with the help of a current shunt which gives out an analog signal with a DC offset of $V_{DD}/2$, so that the voltage can swing in both directions. Thus, at a value of $V_{DD}/2$ the system draws zero amps.

For faster response time, two internal Analog Comparators in Window mode are used to detect the overcurrent.

The upper and lower limits of the window are generated using two DACs, one for each comparator. An interrupt generated by this current window will force the system into a Fault state.

4.9 Stall Detection

This implementation employs three methods of stall detection:

- Zero-cross time-out using the TCA1 CMP2 interrupt
- Using a zero-cross delta limit. This means that if the motor has any abrupt changes on the time in which the zero cross is detected, based on the previous one, then the motor is considered stalled.
- Most likely, during a stall, the current through the windings will rise considerably and get over the overcurrent limit

The most sensitive part of stall detection is at start-up. When the motor is accelerated in Open Loop mode and switching to Closed Loop, the system might get a false response that the zero cross is valid, even though the BEMF

is a result of the magnetic coupling between the driven and the undriven coils, and not the voltage induced in the coils by the moving magnet.

4.10 RPM Measurement

RPM measurement is done using the filtered time of the zero-cross. This means that the RPM is calculated using only half of a trapezoidal step (the point in time when the motor commutates to the point where the zero cross is found). As such, the precision of the motor RPM detection decreases.

The RPM is then computed using the zero-cross timer value, multiplying it with two to get the duration of a trapezoidal step, multiply it with six to obtain the electrical period. After this, it is just a matter of plugging the value into the formula $RPM = 120 / (T_e \times P)$, where T_e is the electrical frequency, and P is the number of the pole pairs specific to the motor.

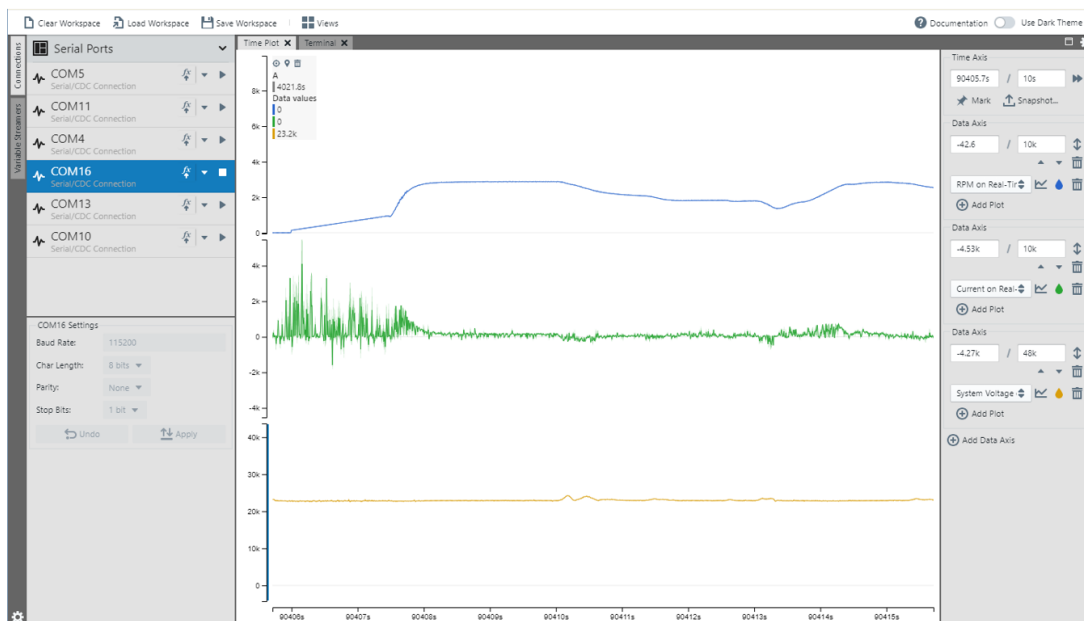
To compensate for this effect, the TCA1 timer/counter prescaler was adjusted in such a way that at low speeds, it does not overflow, while also obtaining the highest value possible.

The process of establishing the best prescaler was done by printing the zero-cross timer value and changing the prescaler to get the maximum value possible at the minimum RPM (that is, the maximum time interval between commutation and zero-cross signal).

4.11 Debugging

The MPLAB® Data Visualizer is used to view live data and for debugging purposes.

Figure 4-7. MPLAB® Data Visualizer Example



It can display the following variables:

- Motor speed in Revolutions per Minute (RPM)
- Motor current in mA
- System voltage in mV

Moreover, debug messages, which display any faults or state changes, can be enabled and viewed in the MPLAB Data Visualizer terminal section. The following lines of code are contained in the application area of the project, in the `app.c` file. It is strongly recommended that only a single functionality is enabled at once, either the debug messages

or the Data Visualizer. Debug messages are disabled by default. To enable them, comment lines 31 and 33 and uncomment line 30. Any changes to the code require recompiling and reuploading the project.

Figure 4-8. Debug Options

```

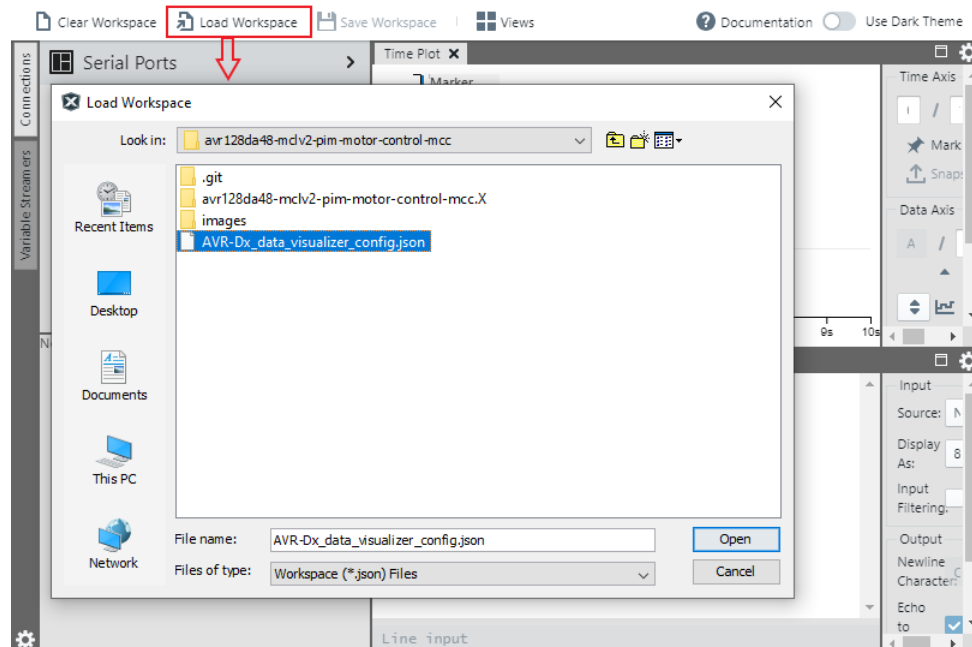
30  // #define COMM_debugPrint(x) printf(x) // comment to disable debug messages
31  #define COMM_debugPrint(x) // uncomment to disable debug messages
32
33  #define ENABLE_DATA_VISUALIZER () // comment to disable data visualizer

```

Configuration

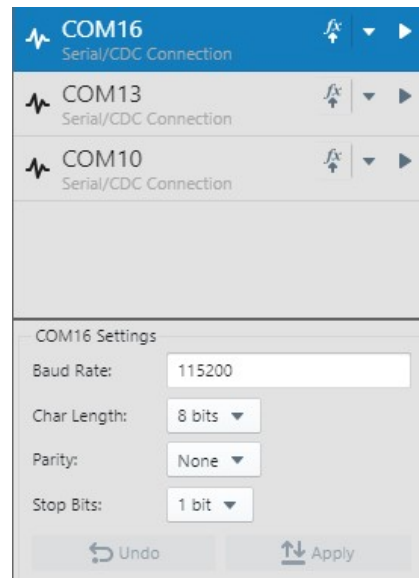
1. Connect the MCLV-2 board to the PC using a Mini-USB cable.
2. Make sure the MPLAB Data Visualizer plug-in is installed.
 - 2.1. Go to **Tools** → **Plug-ins** → **Available** plug-ins and search for MPLAB Data Visualizer.
3. Launch the MPLAB Data Visualizer and click on **Load Workspace** from the top bar. Select the **.json** file in the root of the repository.

Figure 4-9. Loading the Workspace



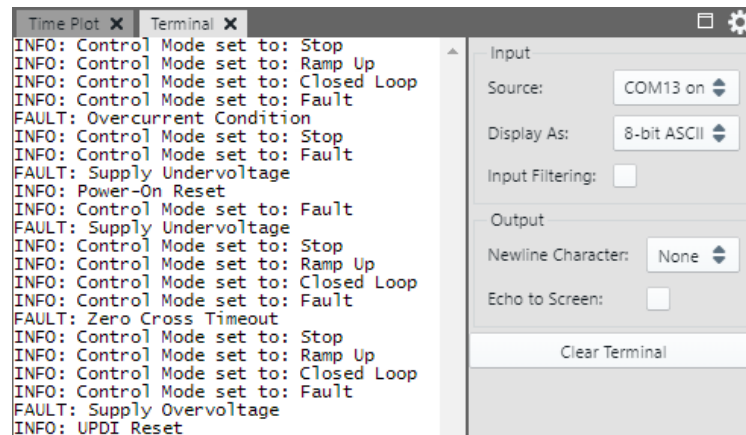
4. On the left **Connections** tab, select the **COM port** and set the Baud Rate to 115200. Leave the rest of the configuration like in [Figure 4-10](#):

Figure 4-10. COM Port Configuration



5. Press the **Start streaming** button for the selected COM port. To view the traces on the Time Plot, the Show Live Data mode needs to be activated.
 - If debug messages are enabled (see the `app.c` source file macros), they can be viewed in the **Terminal** tab, with the Display As mode set to 8-bit ASCII

Figure 4-11. Debug Messages for Various States and Faults



5. Plug-In Module (PIM) Board Description

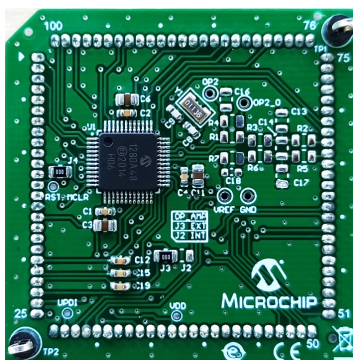
The AVR DX PIM comes in two variants:

- Internal Operational Amplifier variant: Based on the AVR128DB48 microcontroller which includes an internal op amp
- External Operational Amplifier variant: Based on the AVR128DA48, it represents the plain version and uses the external MCLV-2 op amp

Figure 5-1. AVR® DB PIM with Internal Op Amp



Figure 5-2. AVR® DA PIM with External Op Amp



The usage of the internal op amps relies mainly on microcontrollers that contain the op amp module such as the AVR DB family, eliminating the need for an external op amp. The amplified current is then fed through a comparator window with software-adjustable limits leaving the user the possibility to select between internal or external references by changing a single selection jumper on the PIM.

A special configuration board is needed to gain access to both signals from the current shunts and the back-EMF of the motor. It comes in the same package as the PIM. This is only needed when using the internal op amp current reference. However, if the PIM is configured to run with an external op amp, it can also work with the provided MCLV-2 EXTERNAL OP_AMP configuration board.

5.1 Pin Mapping

Table 5-1 provides the mapping between the 100-pin PIM and the 48-pin device.

Table 5-1. Pin Mapping Table (Sorted by PIM Pin Number)

PIM Pin #	Signal Name	Pinout Description	AVR® DX PIN
1	DBG_LED2	Debug LED 2	PA5
2	V _{DD}	Digital supply	VDD
3	PWM1H3	PWM output - 3H	PA6

.....continued

PIM Pin #	Signal Name	Pinout Description	AVR® DX PIN
13	MCLR	Device Master Clear	PF6
15	V _{SS}	Digital supply	GND
16	V _{DD}	Digital supply	VDD
18	FAULT	DC BUS current fault (active-low logic)	PC2
19	TX	UART Transmit	N/A
20	PIM_V_M3	Voltage feedback signal	PD4
21	PIM_V_M2	Voltage feedback signal	PD6
22	PIM_V_M1	Voltage feedback signal	PD3
23	PIM_IMOTOR_SUM	DC bus current signal	PD2
24	PIM_IMOTOR2	Phase current signal	PE0
25	PIM_IMOTOR1	Phase current signal	PF2
26	PGC	Device programming clock line	N/A
27	PGD	Device programming data line	UPDI
28	V _{REF}	Reference voltage (half of AV _{DD} voltage)	PD7
29	PIM_REC_NEUTR	Reconstructed motor neutral line voltage	PD5
30	AV _{DD}	Analog supply	AVDD
31	AV _{SS}	Analog supply	GND
32	PIM_POT	Potentiometer signal	PD0
34	PIM_GEN2	General I/O	PA4
35	PIM_VBUS	DC bus voltage (downscaled)	PD1
36	V _{SS}	Digital supply	GND
37	V _{DD}	Digital supply	VDD
41	PIM_MONITOR_1	Hall sensor/Current sense/Voltage feedback signal	N/A
42	PIM_MONITOR_2	Hall sensor/Current sense/Voltage feedback signal	N/A
43	PIM_MONITOR_3	Hall sensor/Current sense/Voltage feedback signal	N/A
45	V _{SS}	Digital supply	GND
46	V _{DD}	Digital supply	VDD
47	HALLB	Hall sensor/QEI input	PB5
48	HALLC	Hall sensor/QEI input	PB4
49	RX	UART Receive	PA0
50	TX	UART Transmit	PA1
51	USB_TX	UART Transmit (connected directly to U7)	PB1
52	USB_RX	UART Receive (connected directly to U7)	PB0
58	PIM_FLT_OUT2	General I/O	PC6
59	PIM_FLT_OUT1	General I/O	PC5

.....continued

PIM Pin #	Signal Name	Pinout Description	AVR® DX PIN
60	DBG_LED1	Debug LED 1	PC0
61	HOME	Home signal for QE1	PC7
62	V _{DD}	Digital supply	VDD
63	OSC1/CLKO	Crystal oscillator in	N/A
64	OSC2/CLKI	Crystal oscillator out	N/A
65	V _{SS}	Digital supply	GND
66	PIM_IBUS+	BUS current shunt signal	N/A
67	PIM_IBUS-	BUS current shunt signal	N/A
68	LIN_CS	LIN Chip Select signal	PA2
69	LIN_FAULT	LIN Fault signal	PC1
70	RX	UART Receive	N/A
72	USB_RX	UART Receive (connected directly to U7)	N/A
73	PIM_IB+	IMOTOR1 shunt signal	N/A
74	PIM_IA+	IMOTOR2 shunt signal	N/A
75	V _{SS}	Digital supply	GND
76	USB_TX	UART Transmit (connected directly to U7)	N/A
77	CAN_TX	CAN Transmit	N/A
78	CAN_RX	CAN Receive	N/A
80	HALLA	Hall sensor/QEI input	PB2
82	PIM_GEN1	General I/O	N/A
83	BTN_1	Push-button S2 input	PF5
84	BTN_2	Push-button S3 input	PF4
86	V _{DD}	Digital supply	VDD
87	CAN_RX	CAN Receive	N/A
88	CAN_TX	CAN Transmit	N/A
93	PWM1L1	PWM output - 1L	PC3
94	PWM1H1	PWM output - 1H	PA3
98	PWM1L2	PWM output - 2L	PF3
99	PWM1H2	PWM output - 2H	PB3
100	PWM1L3	PWM output - 3L	PA7

Notes:

1. Digital Power (V_{DD}) pins are connected to the PIM.
2. Digital Ground (V_{SS}) and Analog Ground (AV_{SS}) pins are connected to the PIM.

Table 5-2 provides the mapping between the 48-pin device and the 100-pin PIM.

Table 5-2. Pinout Table (Sorted by AVR® DX Pin Name)

PIM Pin #	Pin Name	Pinout Description	AVR® DX PIN
30	AV _{DD}	Analog supply	AVDD
31	AV _{SS}	Analog supply	GND
15	V _{SS}	N/A	GND
36	V _{SS}	N/A	GND
45	V _{SS}	N/A	GND
65	V _{SS}	N/A	GND
75	V _{SS}	N/A	GND
19	TX	UART Transmit	N/A
26	PGC	Device programming clock line	N/A
41	PIM_MONITOR_1	Hall sensor/Current sense/Voltage feedback signal	N/A
42	PIM_MONITOR_2	Hall sensor/Current sense/Voltage feedback signal	N/A
43	PIM_MONITOR_3	Hall sensor/Current sense/Voltage feedback signal	N/A
63	OSC1/CLKO	Crystal oscillator in	N/A
64	OSC2/CLKI	Crystal oscillator out	N/A
66	PIM_IBUS+	BUS current shunt signal	N/A
67	PIM_IBUS-	BUS current shunt signal	N/A
70	RX	UART Receive	N/A
72	USB_RX	UART Receive (connected directly to U7)	N/A
73	PIM_IB+	IMOTOR1 shunt signal	N/A
74	PIM_IA+	IMOTOR2 shunt signal	N/A
76	USB_TX	UART Transmit (connected directly to U7)	N/A
77	CAN_TX	CAN Transmit	N/A
78	CAN_RX	CAN Receive	N/A
82	PIM_GEN1	General I/O	N/A
87	CAN_RX	CAN Receive	N/A
88	CAN_TX	CAN Transmit	N/A
49	RX	UART Receive	PA0
50	TX	UART Transmit	PA1
68	LIN_CS	LIN Chip Select signal	PA2
94	PWM1H1	PWM output - 1H	PA3
34	PIM_GEN2	General I/O	PA4
1	DBG_LED2	Debug LED 2	PA5
3	PWM1H3	PWM output - 3H	PA6
100	PWM1L3	PWM output - 3L	PA7
52	USB_RX	UART Receive (connected directly to U7)	PB0

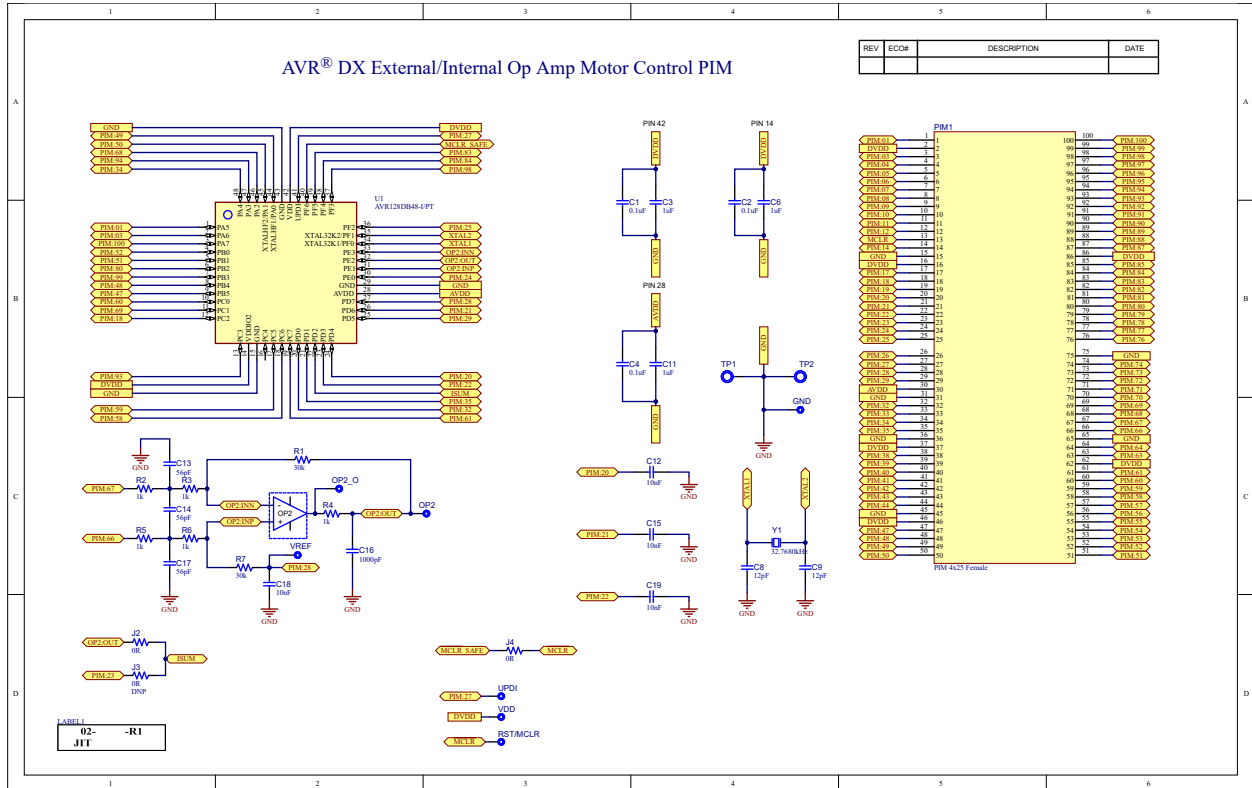
.....continued			
PIM Pin #	Pin Name	Pinout Description	AVR® DX PIN
51	USB_TX	UART Transmit (connected directly to U7)	PB1
80	HALLA	Hall sensor/QEI input	PB2
99	PWM1H2	PWM output - 2H	PB3
48	HALLC	Hall sensor/QEI input	PB4
47	HALLB	Hall sensor/QEI input	PB5
60	DBG_LED1	Debug LED 1	PC0
69	LIN_FAULT	LIN Fault signal	PC1
18	FAULT	DC BUS current fault (active-low logic)	PC2
93	PWM1L1	PWM output - 1L	PC3
59	PIM_FLT_OUT1	General I/O	PC5
58	PIM_FLT_OUT2	General I/O	PC6
61	HOME	Home signal for QEI	PC7
32	PIM_POT	Potentiometer signal	PD0
35	PIM_VBUS	DC bus voltage (downscaled)	PD1
23	PIM_IMOTOR_SUM	DC bus current signal	PD2
22	PIM_V_M1	Voltage feedback signal	PD3
20	PIM_V_M3	Voltage feedback signal	PD4
29	PIM_REC_NEUTR	Reconstructed motor neutral line voltage	PD5
21	PIM_V_M2	Voltage feedback signal	PD6
28	V _{REF}	Reference voltage (half of AV _{DD} voltage)	PD7
24	PIM_IMOTOR2	Phase current signal	PE0
25	PIM_IMOTOR1	Phase current signal	PF2
98	PWM1L2	PWM output - 2L	PF3
84	BTN_2	Push-button S3 input	PF4
83	BTN_1	Push-button S2 input	PF5
13	MCLR	Device Master Clear	PF6
27	PGD	Device programming data line	UPDI
2	V _{DD}	N/A	VDD
16	V _{DD}	N/A	VDD
37	V _{DD}	N/A	VDD
46	V _{DD}	N/A	VDD
62	V _{DD}	N/A	VDD
86	V _{DD}	N/A	VDD

Notes:

1. Digital Power (V_{DD}) pins are connected to the PIM.
2. Digital Ground (V_{SS}) and Analog Ground (AV_{SS}) pins are connected to the PIM.

5.2 PIM Schematic

Figure 5-3. PIM Schematic

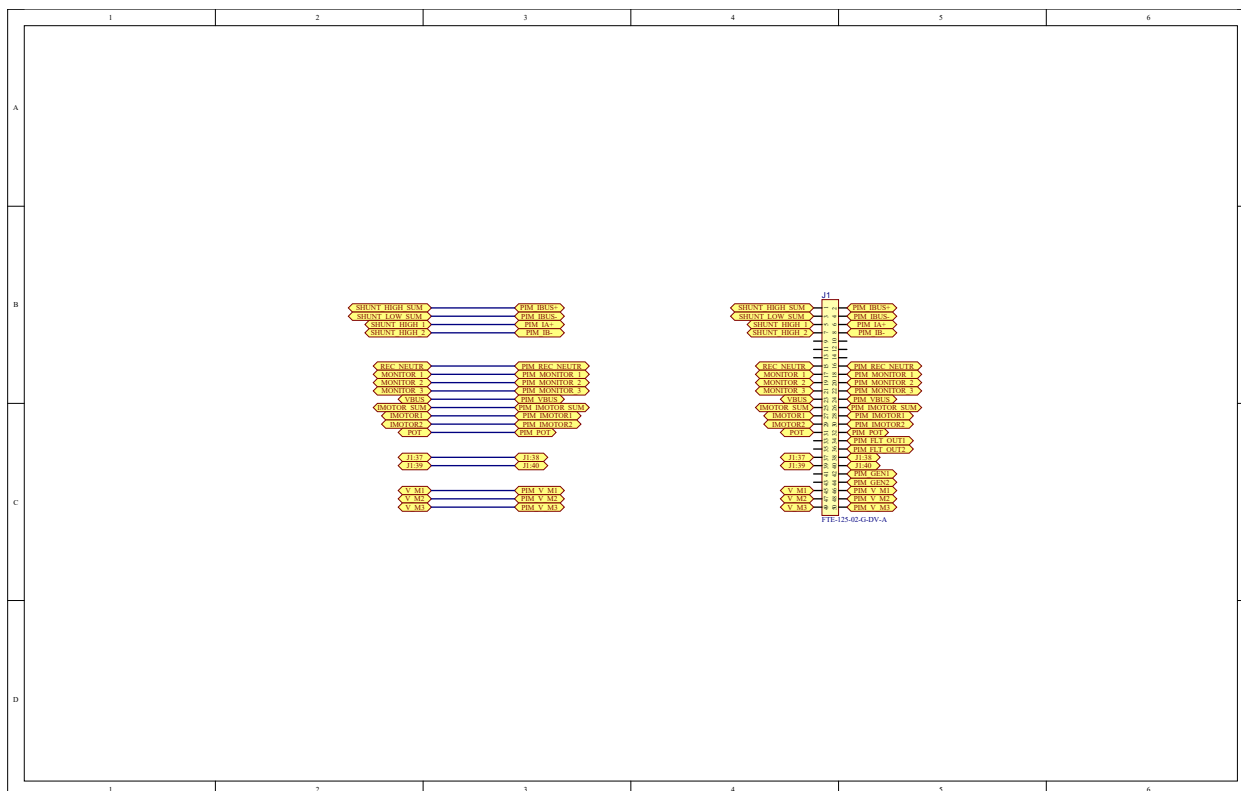


Notes:

1. Jumpers J2 and J3 are used to switch between external and internal op amp.
2. Jumper J4 can be replaced with a resistor to protect the microcontroller against an accidental HV programming mode.

5.3 Configuration Board Schematic

Figure 5-4. PIM Schematic

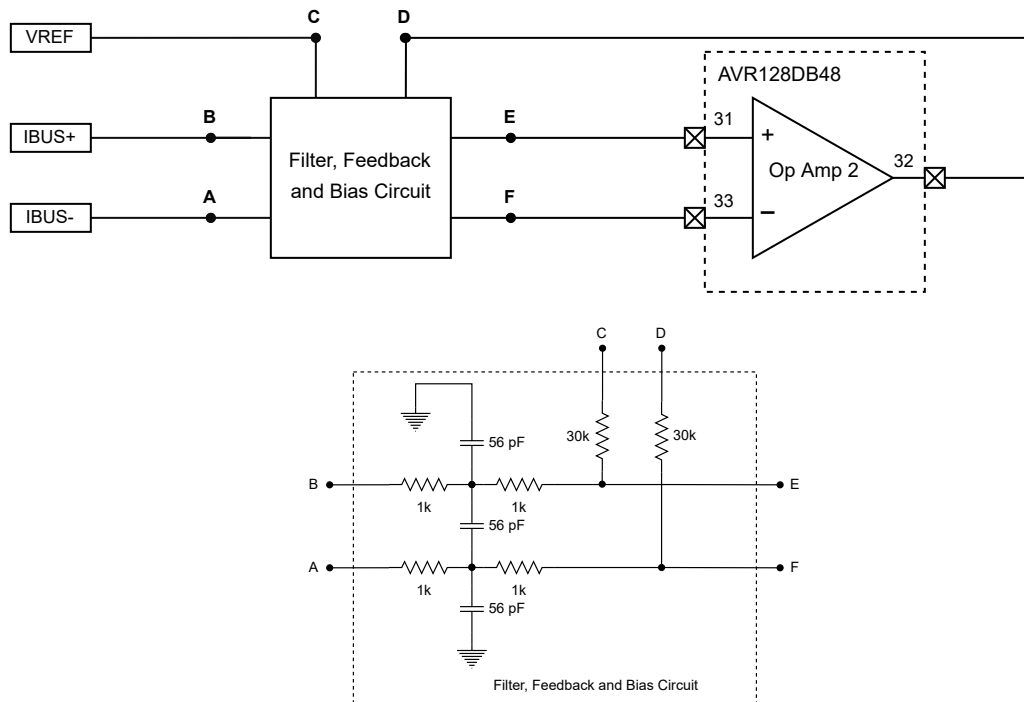


5.4 Analog Functionality

Table 5-3. Internal Op Amp Parameters

Op Amp #	Analog Function	Passive Components	Design Equations
2	Low-pass filter	R2, R3, R5, R6, C13, C14, C17	$R2 = R3 = R5 = R6 = R$ $C13 = C17 = C$
	Reference voltage bias	R1, R7	$R1 = R7$
	Differential amplifier input	R2, R3, R5, R6	Common – mode $f_{-3\text{ dB}} \cong \frac{1}{2\pi RC}$
	Differential amplifier feedback	R1	Differential – mode $f_{-3\text{ dB}} \cong \frac{1}{2\pi(2R)\left(\frac{C}{2} + C14\right)}$ Differential amplifier gain $\cong \frac{R7}{2R}$

Figure 5-5. Internal Op Amp Configuration

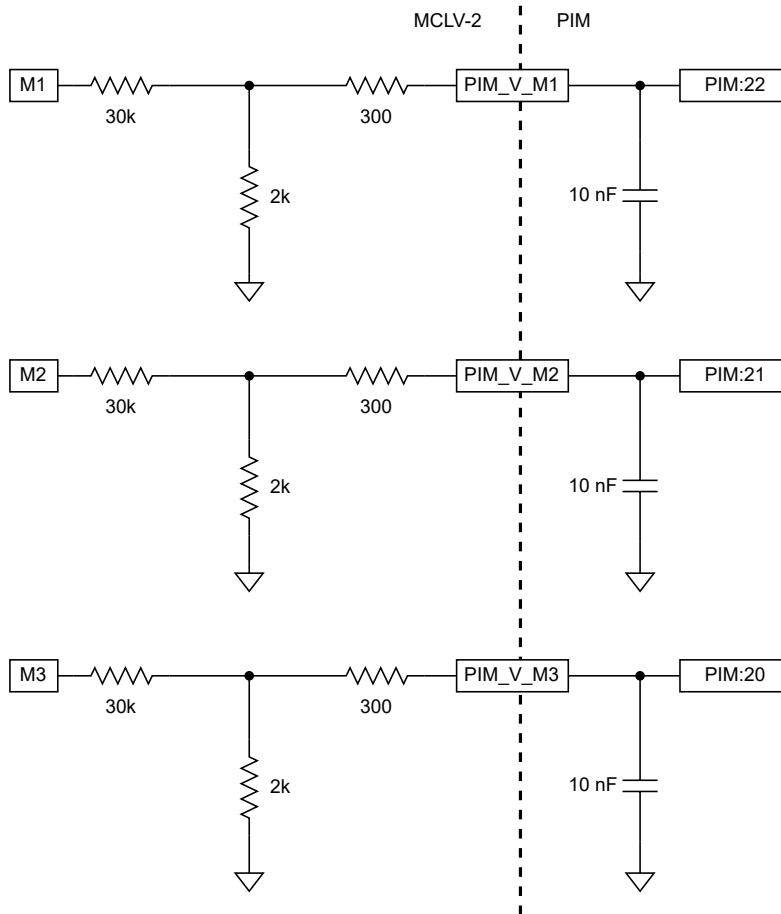


$$\text{Differential amplifier gain} = \frac{30k\Omega}{2 * 1k\Omega} = 15$$

$$\text{Differential mode } f_{-3\text{ dB}} \cong \frac{1}{2\pi(2 * 1k\Omega)\left(\frac{56\text{ pF}}{2} + 56\text{ pF}\right)} \cong 948\text{ KHz}$$

$$\text{Common - mode } f_{-3\text{ dB}} \cong \frac{1}{2\pi(1k\Omega)(56\text{ pF})} \cong 2.8\text{ MHz}$$

Figure 5-6. Voltage Divider and Low-Pass Filter for Phase Back-EMF



$$\text{Cutoff frequency}_{-3 \text{ dB}} = \frac{1}{2\pi(30k \parallel 2k + 300) 10 \text{ nF}} \cong 7321 \text{ Hz}$$

$$30 \text{ degree phase shift frequency} = \frac{\tan(30)}{2\pi(30k \parallel 2k + 300) 10 \text{ nF}} = 4226 \text{ Hz}$$

Thus, the theoretical maximum speed of the motor will be:

$$4226 \text{ Hz} * 60 \cong 253560 \text{ eRPM}$$

$$\text{Mechanical RPM} = \text{eRPM} / \text{Pole pairs}$$

5.5 User Interface

Table 5-4 describes the user interface with the MCLV-2 board using the AVR DX PIM.

Table 5-4. User Interface Description

Interface	Type	Function	Description
S1	Input	Reset	Pressing this button will reset the microcontroller and bring the system to the Driver Init, App Init state
S2	Input	Start/Stop	Starts or stops the motor. If in Error Handling state, a press on this button will bring the system to the Stop state (braking PWM LEDs D10, D12, and D14 ON).
S3	Input	Change direction	Changes the direction of the motor. This only works when the system is in Stop state. A change of the direction also changes the direction LEDs.
POT1	Input	Set duty cycle	Directly sets the duty cycle of the PWM signal
D2, D17	Output	Motor direction indication	Indicates the direction of the motor (CW or CCW, depending on the motor wiring)
D10-D15	Output	PWM outputs	Visual indication if the PWM outputs are active. No LED turned ON means that the system is either turned OFF or is in Error Handling state.

6. Tuning

6.1 First Start-Up

The motor will start without issues if a 24V supply voltage is provided and the default values are used.

However, the number of the pole pairs of the used motor may be different from the one used in the default configuration. If this is the case, normal behavior is exhibited, but the RPM seen in the MPLAB Data Visualizer will have an incorrect value.

To fix this issue, two fields in the `AVR_Dx_tuning_params.xslm` excel file need to be modified:

- Motor pole pairs: Change according to the used motor
- Target RPM: Change according to the formula:
 - $\text{NewRPM} = \text{NewPolePairs} * \text{OldRPM} / \text{OldPolePairs}$
 - Round the number to the nearest integer

If the results are in order and the motor response is correct, then tuning is not necessary.

6.2 Excel File Parameters Tuning

It may be that the motor fails to start, or it does not exhibit good performance.

Figure 6-1. Excel File Editable Parameters

	A	B	C	D
	Parameter	Value	Description	Range
1	Startup PWM Duty [%]	25	Startup duty cycle in percents	1 - 100
2	Rotor align time [ms]	250	Rotor align time in which the duty is raised to the startup PWM duty above	1 - 14000
3	Target RPM [RPM]	800	Final RPM after startup. This becomes the minimum RPM(also see tolerance below)	
4	Initial startup period [ms]	300	From where to start the ramp. Never start from zero.	1 - 1000
5	Ramp duration [ms]	2000	Total ramp time without sustain	100 - 5000
6	Ramp sustain duration [ms]	1	Time in which the motor is kept at a constant speed after acceleration	1 - 5000
7	Holdoff steps	1	How many trapezoidal steps have the PWM disabled, after the sustain zone	1-250
8	Motor pole pairs	4	The number of motor pole pairs	1 - 255
9	Minimum PWM duty [%]	20	Minimum PWM duty value in percents - Indirectly limits the minimum speed	0 - 100
10	Maximum PWM duty[%]	100	Minimum PWM duty value in percents - Indirectly limits the maximum speed	0 - 100
11	Braking PWM duty [%]	90	PWM duty cycle for the non-regenerative braking	0 - 100
12	Motoring current limit[mA]	4420	Maximum current allowed to draw from the supply. Triggers a fault condition	0 - 4420
13	Braking current limit[mA]	-4420	Maximum current allowed to push to the supply. Triggers a fault condition	-4420 - 0
14	Undervoltage protection[mV]	11000	Maximum supply voltage. Triggers a fault condition	11000-48000
15	Overvoltage protection[mV]	25000	Minimum supply voltage. Triggers a fault condition	11000-48000
16	Moving average filter division factor	8	Higher values eliminates glitches but has slower response. Set as power of 2	1 - 128
17	Advance angle[Deg]	0	Advance angle for the next sector timing	0 - 30
18	Passthrough delay compensation[us]	200	Fixed compensation time for any hardware or software delays	1-1000
19	Delta division factor	1	Maximum variation of the current zero cross time expressed as (previous time/factor)	0-8(disabled for 0)
20	Minimum RPM tolerance[%]	40	Minimum RPM for closed loop mode. Subtracts this percentage from the startup RPM	0-100

The following paragraphs will focus on the tuning parameters and their impact on the system behavior:

1. **Start-up PWM Duty [%]** – It represents the start-up duty cycle in percentage. A value that is too low may be insufficient for the ramp-up sequence as the current through the windings is too low and the generated magnetic field is not strong enough for the rotor to follow.
A value that is too high will result in a trigger of the overcurrent protection or excessive heating of the windings. The start-up duty cycle is also strongly dependent on the motor load at start-up. Higher loads demand higher duty cycles.
This parameter applies to the ramp-up sequence only.
2. **Rotor align time [ms]** – The rotor align time in milliseconds. The duty cycle is gradually increased using a linear ramp from 0% to the Start-up PWM Duty value so there is no mechanical shock. This alignment is done for the motor to start from a known position.
This parameter applies to the ramp-up sequence only.
3. **Target RPM [RPM]** – This will be the final RPM after the ramp-up sequence. If it is set too high, then there will be a need for a higher duty cycle so that the motor can follow the magnetic field at high RPMs.
This parameter applies to the ramp-up sequence only, but the closed-loop control also depends on it.

4. **Initial start-up period [ms]** – Start-up point of the ramp. There is generally no need to tune this parameter. This parameter applies to the ramp-up sequence only.
5. **Ramp duration [ms]** – Sets how much the ramp will last. For faster start-up times, lower this value. Beware that strong accelerations also demand higher duty cycles. This parameter applies to the ramp-up sequence only.
6. **Ramp sustain duration [ms]** – After the ramp-up sequence, where the motor is accelerated, there is a sustained period where the motor is spun with the **Target RPM** speed. Unless there is a PI controller used, where it can benefit in the training of the controller, it is recommended for this parameter to be kept to a minimum. This parameter applies to the ramp-up sequence only.
7. **Hold-off steps** – Immediately after the ramp-up sequence, the PWM is disabled and the comparator is enabled. This is done to detect the exact position of the rotor in absence of any phase noise. This parameter needs to be kept to a minimum of one unless there is a very slow variation of the phase voltage of the motor. This parameter applies to the ramp-up sequence only.
8. **Motor pole pairs** – The number of pole pairs of the motor.
9. **Minimum PWM duty [%]** – The minimum PWM duty cycle. Use this parameter to limit the minimum speed of the motor. This parameter applies to the closed and open-loop control.
10. **Maximum PWM duty [%]** – The maximum PWM duty cycle. Use this parameter to limit the maximum speed of the motor. This parameter applies to the closed and open-loop control.
11. **Braking PWM duty [%]** – The duty cycle for the motor in Stop mode. The system is protected from overcurrents caused by braking at high speeds. In this case, the firmware will get to the Fault state and protect the transistors. This parameter applies to the Stop state only.
12. **Motoring current limit [mA]** – The maximum current value that can be drawn from the bus over which the Fault condition is triggered. The maximum value is limited by MCLV-2 to 4420 mA. This parameter applies to all system states except for the Error Handling state.
13. **Braking current limit [mA]** – The maximum current value that can be pushed to the bus over which the Fault condition is triggered. The maximum value is limited by MCLV-2 to -4420 mA. This parameter applies to all system states except for the Error Handling state.
14. **Undervoltage protection [mV]** – The minimum voltage on the bus for the MCLV-2. Do not set under 11V. This parameter applies to all system states except for the Error Handling state.
15. **Overvoltage protection [mV]** – The maximum voltage for the system. Do not set over 24V if using a single supply, or over 48V if using dual supplies (see the [MCLV-2 user guide](#)). This parameter applies to all system states except for the Error Handling state.
16. **Moving average filter division factor** – The division factor for the moving average filter of the zero-cross time needs to be set as a power of two because it can be optimized by the compiler with right shifts. Higher values are recommended because the system has higher immunity to any glitches but limits the maximum variation between 2 zero-cross events, thus limiting the maximum acceleration of the motor. This parameter applies to the closed-loop control only.
17. **Advance angle [Deg]** – The advance angle of the motor. Increasing this value will advance the generation of the waveforms with that specific degrees, concerning the motor back-EMF zero-cross point. Also, it will have an impact on the torque generated by the motor. This parameter applies to the closed-loop control only.
18. **Pass-through delay compensation [us]** – Fixed advance time for the next sector timing. This value is directly subtracted from the next sector timing to compensate for any software or hardware delays. This parameter applies to the closed-loop control only.
19. **Delta division factor** – This parameter controls an additional stall and glitch detection mechanism. In the case of very early or very late detection of the zero-cross point, if the absolute difference between the previous zero-cross time and the currently measured time is more than the previous zero-cross time/Delta division factor, then a Fault condition is triggered. The Delta division factor is ideal to be chosen as a power of two for code efficiency. A value too high will make the system very unstable and will limit the maximum acceleration by triggering the Fault condition. A value too low may not detect a glitch very well but will not encounter any troubles at start-up. This parameter applies to closed-loop control only.

20. **Minimum RPM tolerance [%]** – This parameter controls the time-out and the minimum RPM for the timer. After the start-up, it may be that the system will detect the zero-cross point a bit later than it needs to, and if the tolerance is very low, it will be considered as a time-out event. To avoid that, a given tolerance is needed under the shape of a percentage of the minimum RPM. The minimum RPM thus becomes **Target RPM - Target RPM * Minimum RPM tolerance**.
This parameter applies to the closed-loop control only.

After modifying the parameters, press the **Patch File** button, then recompile and program the AVR DX with the firmware.

6.3 Changing the Control Mode

After the ramp-up, the system will either run in Closed Loop or Open Loop mode. The user is free to choose between each one by configuring a single line of code.

Figure 6-2. Control Mode Code Line

```
35 | /* Control mode after ramp-up */
36 | #define MOTOR_CONTROL_MODE MOTOR_CLOSED_LOOP_MODE
```

The valid options are: `MOTOR_CLOSED_LOOP_MODE` or `MOTOR_OPEN_LOOP_MODE`.

The behavior of these two modes is described below:

1. Closed Loop mode: In the Default Control mode, the commutations happen according to the zero-cross events received from the internal comparator. The active protections are the following:
 - Overcurrent protection
 - Overvoltage/undervoltage protection
 - Zero-cross delta over limit
 - Zero-cross time-out
2. Open Loop mode: In the Alternate control mode, the commutations happen with the help of the timer responsible for the ramp-up sequence. The motor is spun at the RPM set in the Excel file. It can be particularly useful when testing if the ramp-up sequence works correctly and the rotor does not stop following the magnetic field near the end of the ramp. In this mode, the potentiometer is used to set the duty cycle, but the speed of the motor will remain constant. The following protections are active in this mode:
 - Overcurrent protection
 - Overvoltage/undervoltage protection

7. Conclusion

This application note describes a method of driving a BLDC motor using bipolar switching, starting from generating the waveforms and routing them to the required pins.

There is no need for external comparators or other logic circuits, the only exception being a few passive components to condition the input signal going to the microcontroller.

The AVR core running at 24 MHz, providing up to 24 MIPS, is a very powerful choice in designing a system that integrates multiple requirements alongside the Core Independent Peripherals (AC, ADC, TCA, TCB, CCL, USART) used to drive the motor as a complete control system.

Bipolar drive ensures clean BEMF which helps an accurate detection of the zero-cross point, without the need for synchronization with the PWM output signal.

For better performance, the built-in moving average filter adds an extra measure of protection in case of an external disturbance and ensures a smooth run throughout the RPM range of the motor.

8. References

1. [AN857, Brushless DC Motor Control Made Easy \(DS00857\)](#)
2. [AN1160, Sensorless BLDC Control with Back-EMF Filtering Using a Majority Function \(DS01160\)](#)
3. [AVR444: Sensorless control of 3-phase brushless DC motors](#)
4. [AN2522, Core Independent Brushless DC Fan Control Using CCL AVR® microcontrollers \(DS00002522\)](#)

9. Revision History

Doc. Rev.	Date	Comments
B	6/2021	Updated Introduction section with GitHub link.
A	5/2021	Initial document release.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

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

- 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 and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- 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. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like 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.

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.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-8307-6

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com	Australia - Sydney Tel: 61-2-9868-6733 China - Beijing Tel: 86-10-8569-7000 China - Chengdu Tel: 86-28-8665-5511 China - Chongqing Tel: 86-23-8980-9588 China - Dongguan Tel: 86-769-8702-9880 China - Guangzhou Tel: 86-20-8755-8029 China - Hangzhou Tel: 86-571-8792-8115 China - Hong Kong SAR Tel: 852-2943-5100 China - Nanjing Tel: 86-25-8473-2460 China - Qingdao Tel: 86-532-8502-7355 China - Shanghai Tel: 86-21-3326-8000 China - Shenyang Tel: 86-24-2334-2829 China - Shenzhen Tel: 86-755-8864-2200 China - Suzhou Tel: 86-186-6233-1526 China - Wuhan Tel: 86-27-5980-5300 China - Xian Tel: 86-29-8833-7252 China - Xiamen Tel: 86-592-2388138 China - Zhuhai Tel: 86-756-3210040	India - Bangalore Tel: 91-80-3090-4444 India - New Delhi Tel: 91-11-4160-8631 India - Pune Tel: 91-20-4121-0141 Japan - Osaka Tel: 81-6-6152-7160 Japan - Tokyo Tel: 81-3-6880-3770 Korea - Daegu Tel: 82-53-744-4301 Korea - Seoul Tel: 82-2-554-7200 Malaysia - Kuala Lumpur Tel: 60-3-7651-7906 Malaysia - Penang Tel: 60-4-227-8870 Philippines - Manila Tel: 63-2-634-9065 Singapore Tel: 65-6334-8870 Taiwan - Hsin Chu Tel: 886-3-577-8366 Taiwan - Kaohsiung Tel: 886-7-213-7830 Taiwan - Taipei Tel: 886-2-2508-8600 Thailand - Bangkok Tel: 66-2-694-1351 Vietnam - Ho Chi Minh Tel: 84-28-5448-2100	Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 Denmark - Copenhagen Tel: 45-4485-5910 Fax: 45-4485-2829 Finland - Espoo Tel: 358-9-4520-820 France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 Germany - Garching Tel: 49-8931-9700 Germany - Haan Tel: 49-2129-3766400 Germany - Heilbronn Tel: 49-7131-72400 Germany - Karlsruhe Tel: 49-721-625370 Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 Germany - Rosenheim Tel: 49-8031-354-560 Israel - Ra'anana Tel: 972-9-744-7705 Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781 Italy - Padova Tel: 39-049-7625286 Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340 Norway - Trondheim Tel: 47-72884388 Poland - Warsaw Tel: 48-22-3325737 Romania - Bucharest Tel: 40-21-407-87-50 Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 Sweden - Gothenberg Tel: 46-31-704-60-40 Sweden - Stockholm Tel: 46-8-5090-4654 UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820