# Method for Detecting Motor BEMF in Sinusoidal Drive Using the Dead Time Interval

TB3349

## Introduction

Authors: Gheorghe Turcan, Grig Barbulescu, Teodor Lina Microchip Technology Inc.

There are two standardized methods for driving Brushless DC (BLDC) motors or Permanent Magnet Synchronous Motors (PMSM): Trapezoidal Drive (or Block Commutation) and Sinusoidal Drive.

Trapezoidal Drive is simple to implement, even with 8-bit microcontrollers (MCUs), as it is a driving state machine with state switching dictated by the few feedback sources. In a Sensorless solution, the revolution indicator is the motor Back Electromotive Force (BEMF), which provides the position of the rotor every 60° based on the Zero-Cross Detection (ZCD) of each of the three motor's phases.

To measure the BEMF of a spinning motor, the coils must not be driven such that the signal is not distorted. This condition occurs when both the high-side, and this condition is encountered when both the high-side and the low-side MOSFETs of the half-bridge are driven to a logical low level.

BEMF measurement becomes possible in Trapezoidal Drive because two of the six steps of one complete electrical revolution fulfill the necessary condition one phase at a time, and the other phases’ interferences are filtered by hardware and software algorithms.

Sinusoidal Drive has less torque ripple, higher efficiency, and better torque at low speed when compared to Trapezoidal Drive, but with the cost of greater complexity. Moreover, in Sinusoidal Drive, all three phases are driven continuously. Thus, there is no window for the BEMF to be acquired directly.

The usual method used for BEMF acquisition in Sinusoidal Drive is Field Oriented Control (FOC), which requires detailed and fast individual coil current measurements, complex mathematical transformations, and dynamic signal reconstruction and prediction. For FOC, an MCU needs a clock frequency higher than 100 MHz, fast ADCs, and mathematical processing power found only in 16-bit and 32-bit cores.

The method presented in this technical brief creates the conditions necessary for a direct BEMF measurement while the drive is still sinusoidal and continuous for all three phases, thus eliminating the need for a complex acquisition and control algorithm.

The solution provides an easy way to obtain the ZCD from the motor for each 60° rotation, thus creating the possibility for drive synchronization to be simplified to a degree where even 8-bit MCUs can keep up and implement a simple Sensorless Sinusoidal Motor Control solution.

# Table of Contents

# 1. Overview

When an inductor moves through a magnetic field, a voltage is induced in it. A brushless motor is a collection of three inductors spinning in a circular arranged magnetic field, and an electrical voltage is induced in the motor's coils. BEMF is the induced voltage that always occurs when the motor spins. Ideally, the BEMF occurs at any non-zero velocity, but in practice, due to noise and limited measurement resolution, a minimum rotation per minute (RPM) is required to have a measurable BEMF which can provide the rotor's position through ZCD detection.

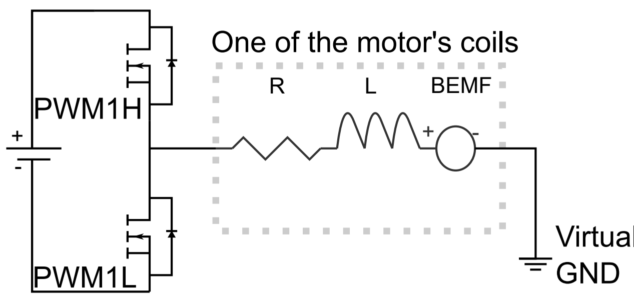
The BEMF voltage has three main parameters: Amplitude, frequency, and phase.

The amplitude is proportional to the rotor's speed and is motor-specific.

The frequency is proportional to the rotor's speed and the number of pole pairs. The motor's RPM can be calculated using the BEMF frequency and equations that capture the motor characteristics.

The phase is important for synchronization between the stator's rotating field and the rotor. The phase and frequency are used as feedback for drive synchronization (phase and speed alignment with the motor) and for implementing control loops.

**Figure 1-1.** One Phase of a Three Phase Brushless Motor




For proper BEMF measurement and ZCD detection, the drive signals for all three phases must be disabled simultaneously to not influence the measurement. Even in Trapezoidal Drive, this is not achieved, as only one phase is not driven. The other two are still being driven and will generate noise. The ZCD is determined using analog and digital filters, along with a Majority detection algorithm. For more information, see Majority Detection algorithm as a reference. These filters are required to reduce the noise and have a close to accurate ZCD identification, but they introduce a delay between the actual ZCD and the measured ZCD.

For Sensorless Sinusoidal drive, all three phases are conducting continuously and there is no undisturbed measurement window that can be used for direct motor BEMF measurement. Thus, the go-to FOC method measures each coil current and recreates the BEMF through a mathematical model, which requires performant hardware and is not applicable for 8-bit MCUs. The dominant method for Sinusoidal Drive on 8-bit MCUs is given by a sensored solution with a Phase-Locked Loop (PLL) synchronization approach.

This technical brief introduces a new Sinusoidal Drive method, which creates a direct BEMF acquisition window that complies with the requirement for the no phase measurement point disturbance from the drive signals, allowing for a simple and unfiltered ZCD detection that is closer to the actual motor coil current ZCD.

The new Sinusoidal Drive method requires Pulse-Width Modulation (PWM) rising event alignment and dead time control, achieved with pure software or a combination of software and hardware-capable peripherals. Thus, the method presented can be used in any 8-bit MCU that can generate complementary PWM signals. However, with dedicated hardware peripherals, as found in Microchip's AVR® EB Family, the code size is drastically reduced, and interrupt time is shortened.

# 2. Sinusoidal Drive Measurement Window Insertion

To remove the influence from the driving signal, the BEMF must be sampled when both of the transistors in the respective half-bridge are driven low.

The complementary PWM dead-time, as depicted in Figure 2-1, provides such a window for sampling the BEMF. The dead-time is required to prevent current shoot-through in the half-bridge.

That said, the dead-time of each phase does not necessarily align. Thus, the proposed method increases one dead-time period on all six PWM signals and the dead-times are aligned, creating a sufficiently long window to measure the BEMF, as depicted in Figure 2-2.

**Figure 2-1.** Dead time and PWM signals illustration

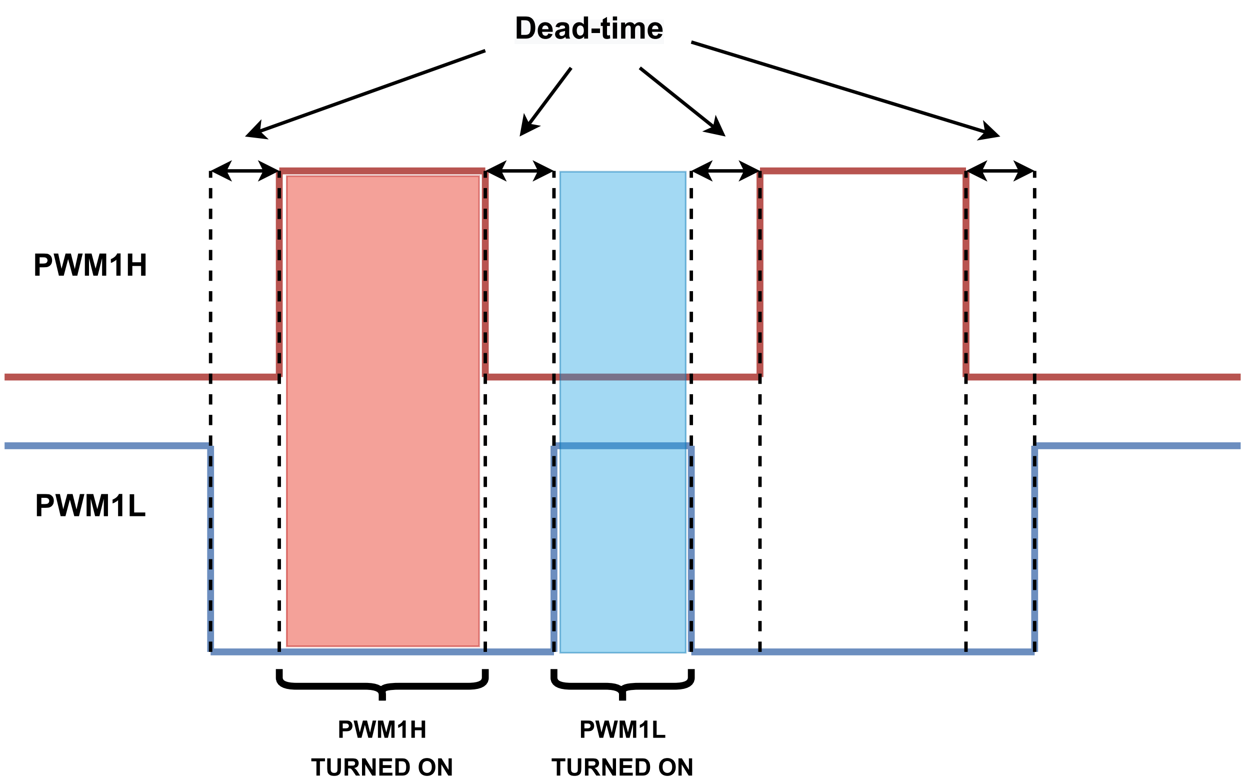
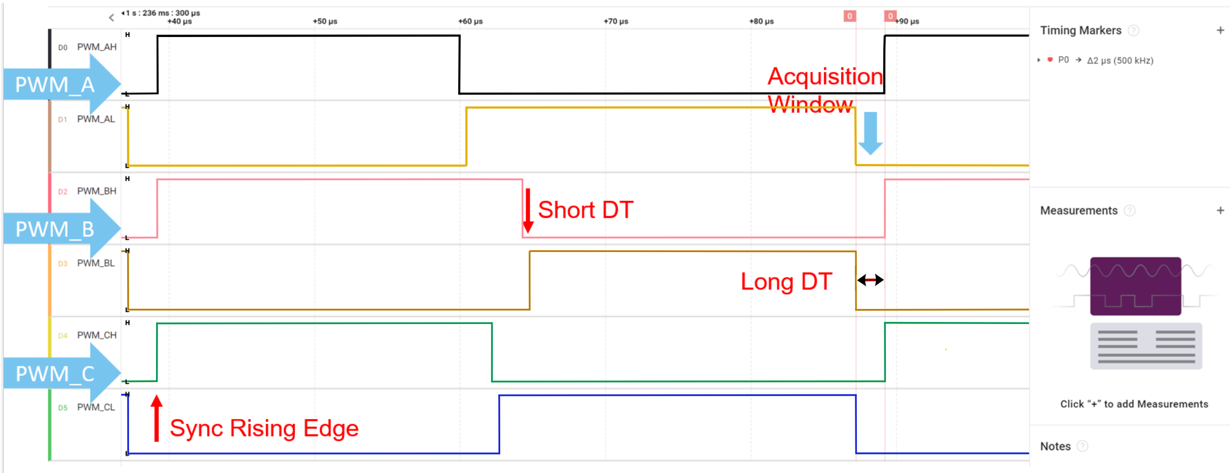


**Figure 2-2.** Logic Analyzer Capture to Visualize the Long Dead Time




Insertion of the BEMF acquisition window brings minor drawbacks when compared to the ordinary sinusoidal drive:

1. Because of the increased dead-time duration, the PWM duty cycle range is decreased by a proportion, dependent on the PWM frequency and the time it takes to correctly assess the BEMF.
2. The half-bridge body diodes get hotter because of the longer dead time. When the BEMF is positive, there are more power losses.

Apart from these disadvantages, this method is a cheap alternative for Sinusoidal Sensorless Control of three-phase motors (BLDC or PMSM) and can be implemented on 8-bit MCUs as it does not require a lot of computational power in one complete PWM cycle. Usually for motor control 20 kHz PWM switching speed is used, so a complete PWM cycle duration is 50 µs.

# 3. BEMF Measurement and Interpretation

Multiple peripherals like the AD converter (ADC) or the Analog Comparator (AC) can be used to sample the BEMF and detect Zero Crosses. In this method, the Analog Comparator samples the BEMF. All the PWM driving signals must synchronously be logically low, the coils are not driven and the signal measured after all the propagation delays is the motor's BEMF.

The Analog Comparator is connected to the divided motor's phase signals on the positive input and the motor summing neutral to the negative input, as depicted in Figure 4-1. The neutral can be either reconstructed in software, or in hardware using resistors in a star configuration. The comparator output acquisition is timed to coincide with the dead time period and possible delays. The comparator output provides the BEMF position equivalent to the current position through the coils. Using this signal, it is possible to find the Zero Cross Detection and estimate the rotor's position. Figure 3-1 shows the Analog Comparator Output, and Figure 3-2 an overview of a Zero Cross point.

**Figure 3-1.** BEMF Interpreted Signal by the Analog Comparator

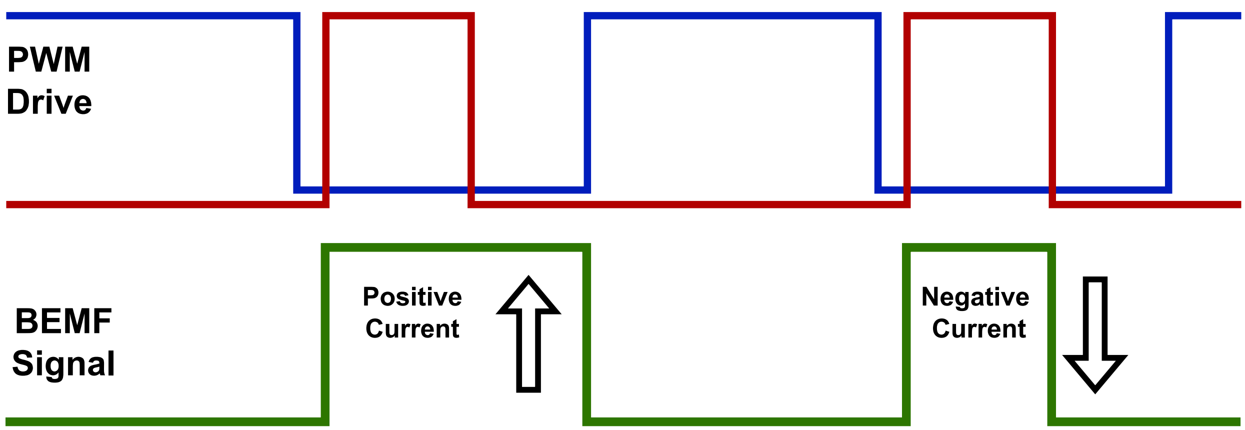
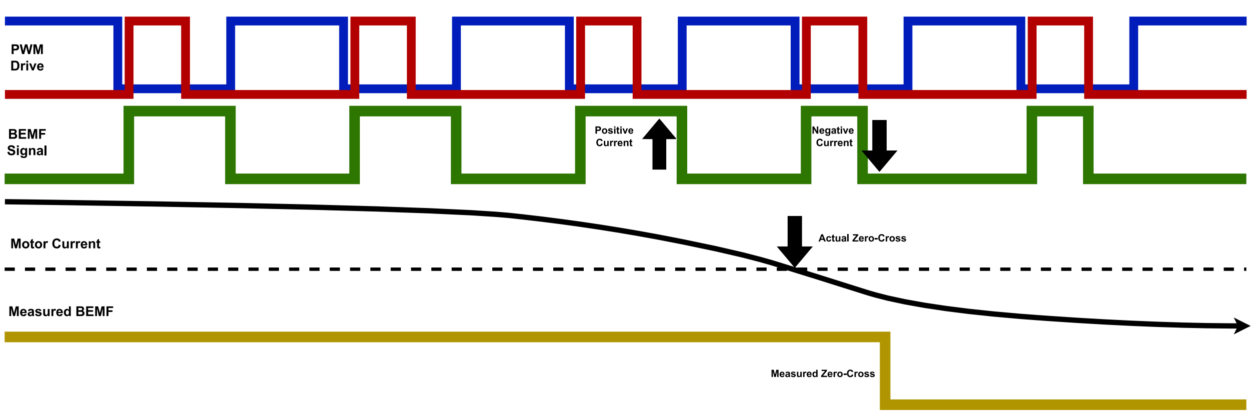



**Figure 3-2.** BEMF Zero Cross Point

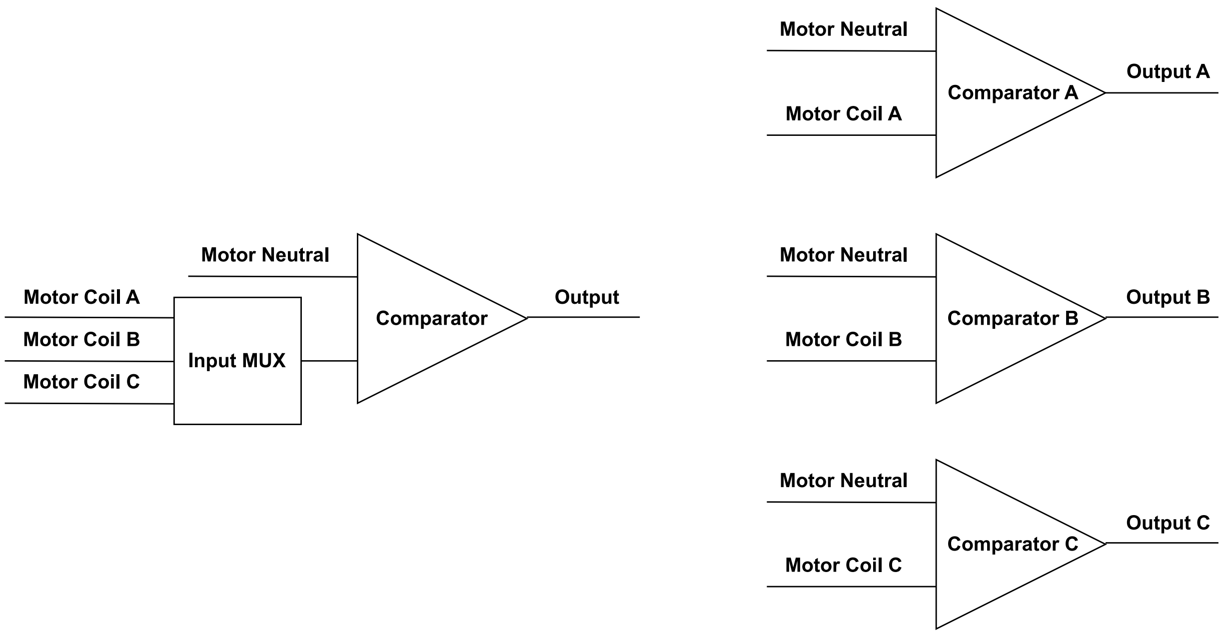


# 4. Drive Synchronization Using BEMF

As the ZCD only happens for one phase at a time, there is no need to sample all three phases simultaneously. Thus, the required components to sample the BEMF can be given by either three comparators, or one comparator with a multiplexed input selection, as shown in Figure 4-1.

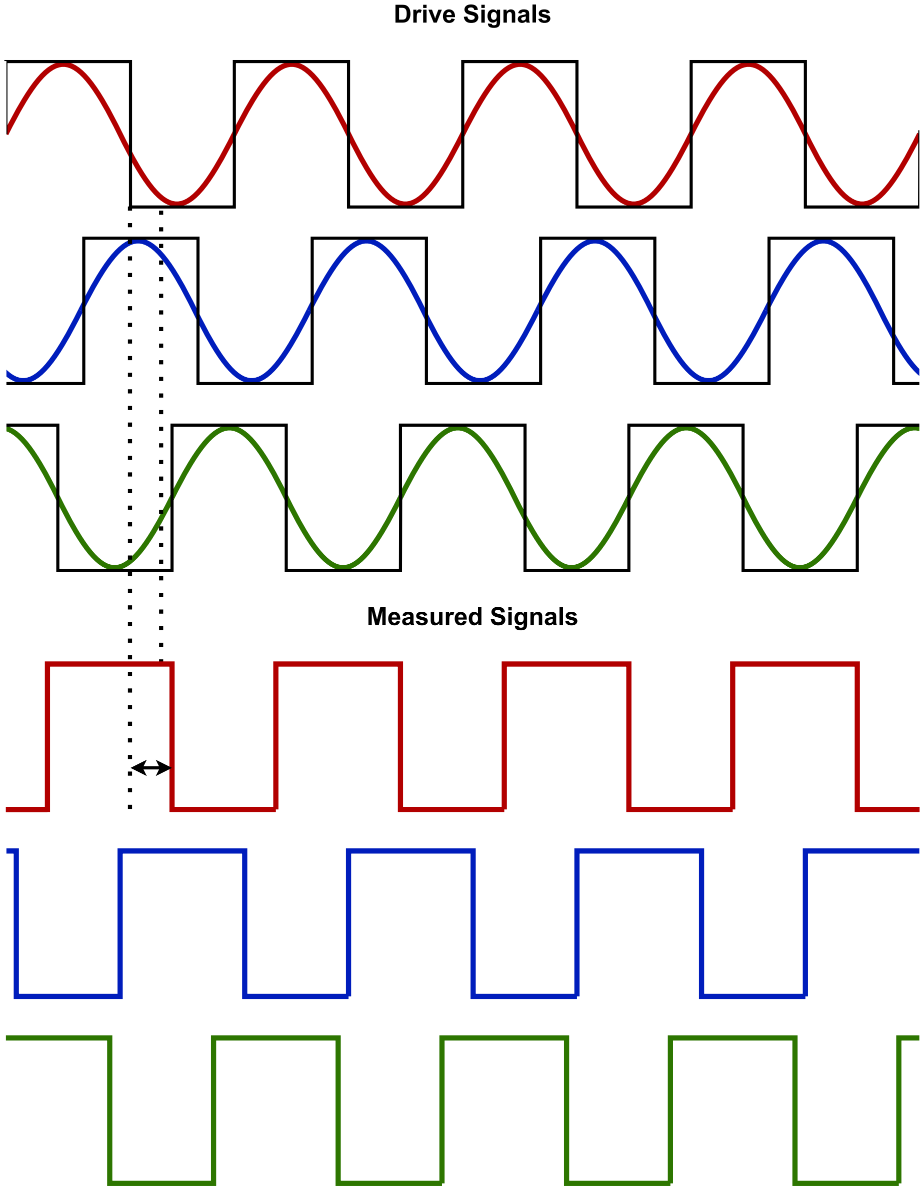
The comparator delays and MUX stabilization must be taken into consideration. To avoid delays, a priority selection method can be used: Measure the first phase of a motor until a zero cross is reached, then measure the second phase until another zero cross is reached, and so on.

**Figure 4-1.** Analog Comparator Possible Configurations




Based on the values of the sampled BEMF, the motor can be ahead or behind the drive field. To reduce the error between the drive field and the motor, a control algorithm can be introduced to improve the synchronization of the motor commutation. Figure 4-2 shows a rendering of all three drive channels with all three filtered BEMF channels when the motor can be behind the drive field.

**Figure 4-2.** Motor's Phases and Measured BEMF for each Phase




Based on the information obtained from the BEMF ZCD, there are three possible cases:

1. The BEMF Signal is ahead of the Drive Signal, so the Drive speed must be reduced to keep synchronization.
2. The BEMF Signal is behind of the Drive Signal, so the Drive speed must be increased to keep synchronization.
3. The BEMF Signal and Drive Signal are in phase with one another, within a range.

Figures 4-3 and 4-4 show 1 and 2 for one motor phase and its BEMF signal. The first figure shows the case when BEMF is ahead of Drive. The second figure shows the case when BEMF is behind the Drive.

**Figure 4-3.** BEMF Signal Ahead of Sine Drive Signal

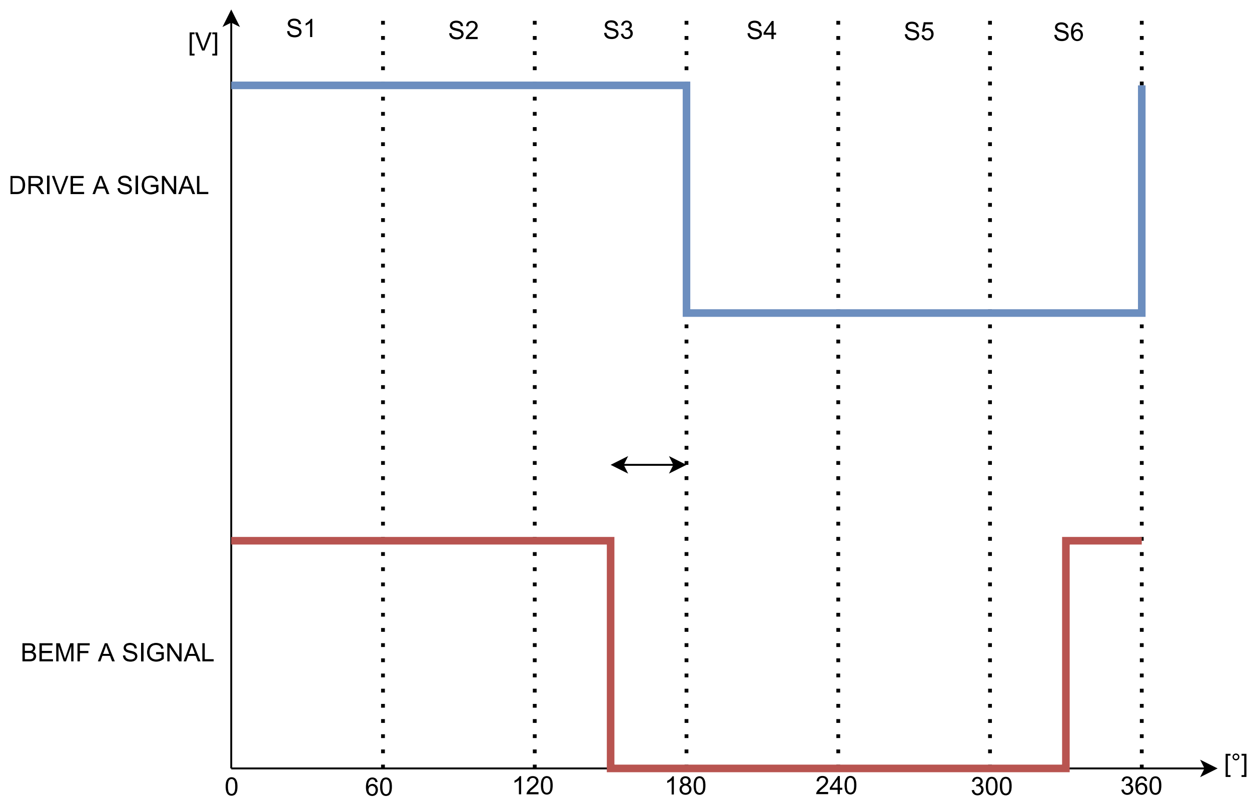
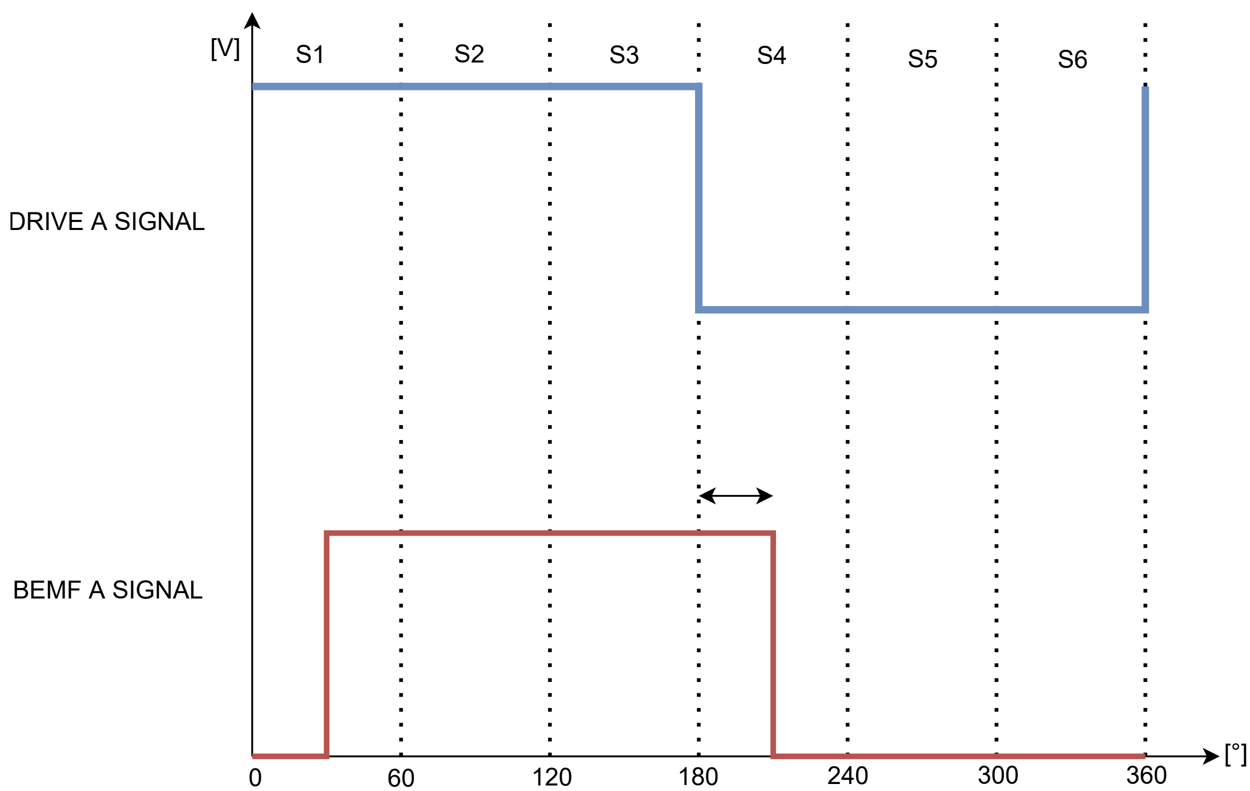



**Figure 4-4.** BEMF Signal Behind of Sine Drive Signal



# 5. BEMF Acquisition Demo

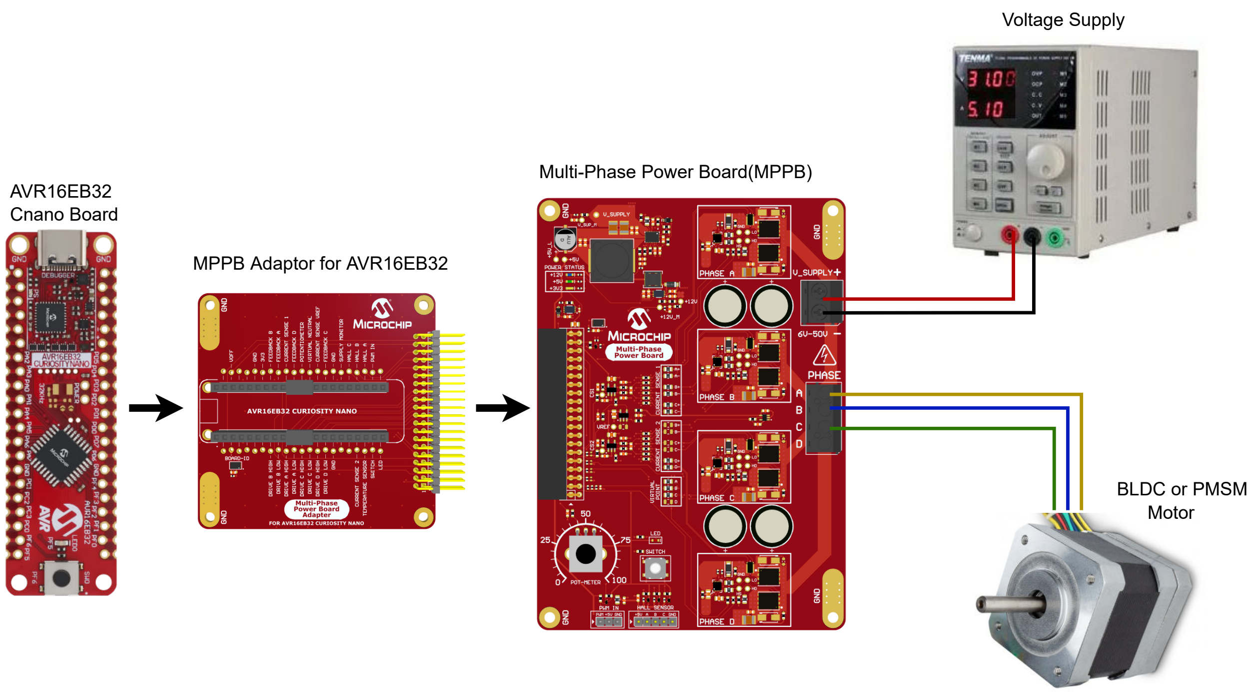
This example shows how to setup the Timer/Counter type E (TCE) and Waveform Extension (WEX) peripherals to generate six complementary PWM signals to spin a three-phase Brushless Direct Current Motor (BLDC) or a Permanent Magnet Synchronous Motor (PMSM) with Sinusoidal Drive. This example also configures the Analog Comparator (AC) peripheral to measure the Back Electromotive Force (BEMF) when the motor is spinning.

The measurement window is one of the two dead time periods during a PWM cycle. Increasing the dead time period to 2 μs gives a big enough acquisition window to detect the current Zero-Cross, uninfluenced by the driving signals. The other dead time period is 500 ns. A complete PWM cycle has 50 μs (the classic 20 kHz frequency of MOSFET switching, commonly used in motor control applications).

In this example, the motor spins using forced commutation without synchronization or closed loop algorithms. This code example highlights the new BEMF measurement method during dead time. The motor is spinning at a constant speed and amplitude that is not modified at run time as it is not the main objective of this application.

The image below shows the setup used in this application:

**Figure 5-1.** Hardware Setup Needed for the Application




To generate this project using MPLAB® Code Configurator (MCC) Melody, MCC Melody (MCC Classic is not supported), follow the next steps:

1. Create a new MPLAB® X IDE project for AVR16EB32.
2. Open MCC from the toolbar, find more information on installing the MCC plug-in here.
3. In MCC Content Manager Wizard, select **MCC Melody**, then click **Finish**.
4. Click **Project Resources**, go to **System**, select CLKCTRL, and disable the Prescaler enable button
5. From the **Device Resources**, go to the **Drivers** and **Timer** window, add the TCE module, then do the following configuration:
   – Module Enable: Must be enabled by default. If not, toggle the button (it turns blue if enabled)

– Clock Selection: System clock (by default, the divider must be 1 - System clock)

- Counter Direction: UP

– Waveform Generation Mode: Single-Slope PWM mode with overflow on TOP

– Requested Period [s]: 0.00005

– Duty Cycle 0 [%]: 0

– Duty Cycle 1 [%]: 0

– Duty Cycle 2 [%]: 0

– Duty Cycle 3 [%]: 98.5

– Waveform Output n: Check the boxes from the Enable column for Waveform Output 0, 1, 2, 3

– Scale mode: CMP values are scaled from the Center, 50% DC

– Scaled Writing to registers: Normal

– Amplitude Control Enable: Toggle the button (it turns blue if enabled)

– Amplitude Value: 0.1

– Generate ISR: Toggle the button (it turns blue if enabled)

– Compare 3 Interrupt Enable: Toggle the button (it turns blue if enabled)

6. From the **Device Resources**, go to **Drivers** and add the WEX module, then do the following configuration:

   – Input Matrix: Direct

   – Update Source: TCE (the update condition for the output signals will be dictated by TCE)

   - Override Settings: Check all the boxes from the Output Enable column for the Waveform Output [0-5]

   – Dead-time Insertion Channel 0 Enable: Toggle the button (it turns blue if enabled)

   – Dead-time Insertion Channel 1 Enable: Toggle the button (it turns blue if enabled)

   – Dead-time Insertion Channel 2 Enable: Toggle the button (it turns blue if enabled)

   – Requested Dead-time Low Side (μs): 2

   – Requested Dead-time High Side (μs): 0.5

7. From the **Device Resources**, go to **Drivers** and add the AC0 module, then do the following configuration:

   – Enable: Toggle the button (it turns blue if enabled)

   – Positive Input MUX Selection: Positive Pin 5

   – Negative Input MUX Selection: Negative Pin 1

   – Output Pad Enable: Toggle the button (it turns blue if enabled)

8. In the **Pin Grid View** window - check if the WEX_WO[0-5] pins are locked as outputs on PORTA. The pins are locked when checking the boxes from the Enable column from Waveform Output n. To change the PORT, click on a pin from another PORT in **Pin Grid View**. Check if PA7 pin is set as AC output. Check if PD4 and PD0 pins are set as inputs for AC. Also, another two pins are necessary as inputs for AC for the other two motor phases. Click on PD5 and PD6 and set them as inputs from Pins -> GPIO. Last, the pins that show the sampled BEMF and the sampling moment must be set as outputs from Pins -> GPIO. These pins are PF1, PF2, PF3 and PF4.
9. In the **Project Resources** window, click the **Generate** button so that MCC will generate all the specified drivers and configurations.

10. After the MCC Melody generates the project files with the configuration explained above, overwrite the content from the main.c file with the following:

    Add macro definitions, variables and Look-up Table (LUT) used to drive the motor:

```
/* Number of pole pairs of a BLDC motor */
#define MOTOR_PAIR_POLES                4

/* MOSFET switching frequency in Hz */
#define F_SAMPLING                      20000.0

/* uint16_t range mapping: 0 - 359.99 electrical degrees -> 0 - 65535 */
#define DEGREES_TO_U16(DEG)             (uint16_t)( (float)(DEG) * 65536.0 / 360.0 + 0.5)

/* Speed conversion from RPM to LUT scrolling speed */
#define RPM_TO_U16(RPM)                 (uint16_t)(((float)(RPM) * 65536.0 * (float)
(MOTOR_PAIR_POLES)) / ((float)(F_SAMPLING) * 60.0) + 0.5)

/* Sets the amplitude of the sine wave signals, and thus the scaling values of duty cycle
 * in U.Q.1.15 format, ranging from 0 to 1.00. Duty cycle scaling is done in hardware
 * using the hardware accelerator of TCE.*/
#define AMP_TO_U16(X)                   (uint16_t)(32768.0*(X) + 0.5)

/* Speed of the motor - 120 RPM */
#define SPEED                           RPM_TO_U16(120)

/* Amplitude of the sine wave - 10% */
#define AMPLITUDE                       AMP_TO_U16(0.1)


#include "mcc_generated_files/system/system.h"

typedef enum
{
    MUX_PHASE_A = (AC_MUXPOS_AINP5_gc | AC_MUXNEG_AINN1_gc),
    MUX_PHASE_B = (AC_MUXPOS_AINP6_gc | AC_MUXNEG_AINN1_gc),
    MUX_PHASE_C = (AC_MUXPOS_AINP3_gc | AC_MUXNEG_AINN1_gc),
} mux_t;

/* LUT that is used to generate a sinusoidal drive */
static const uint16_t sine_lookup_table[] =
{
  16384, 16786, 17187, 17589, 17989, 18389, 18788, 19185, 19580, 19973, 20364,
  20753, 21140, 21523, 21903, 22280, 22653, 23023, 23389, 23750, 24107, 24459, 24807,
  25149, 25486, 25818, 26143, 26463, 26777, 27085, 27386, 27681, 27969, 28250, 28523,
  28790, 29049, 29300, 29543, 29779, 30006, 30226, 30437, 30639, 30833, 31018, 31194,
  31362, 31520, 31670, 31810, 31941, 32062, 32174, 32276, 32369, 32453, 32526, 32590,
  32644, 32689, 32723, 32748, 32763, 32768, 32763, 32748, 32723, 32689, 32644, 32590,
  32526, 32453, 32369, 32276, 32174, 32062, 31941, 31810, 31670, 31520, 31362, 31194,
  31018, 30833, 30639, 30437, 30226, 30006, 29779, 29543, 29300, 29049, 28790, 28523,
  28250, 27969, 27681, 27386, 27085, 26777, 26463, 26143, 25818, 25486, 25149, 24807,
  24459, 24107, 23750, 23389, 23023, 22653, 22280, 21903, 21523, 21140, 20753, 20364,
  19973, 19580, 19185, 18788, 18389, 17989, 17589, 17187, 16786, 16384, 15981, 15580,
  15178, 14778, 14378, 13979, 13582, 13187, 12794, 12403, 12014, 11627, 11244, 10864,
  10487, 10114, 9744, 9378, 9017, 8660, 8308, 7960, 7618, 7281, 6949, 6624, 6304, 5990,
  5682, 5381, 5086, 4798, 4517, 4244, 3977, 3718, 3467, 3224, 2988, 2761, 2541, 2330,
  2128, 1934, 1749, 1573, 1405, 1247, 1097, 957, 826, 705, 593, 491, 398, 314, 241,
  177, 123, 78, 44, 19, 4, 0, 4, 19, 44, 78, 123, 177, 241, 314, 398, 491, 593, 705,
  826, 957, 1097, 1247, 1405, 1573, 1749, 1934, 2128, 2330, 2541, 2761, 2988, 3224,
  3467, 3718, 3977, 4244, 4517, 4798, 5086, 5381, 5682, 5990, 6304, 6624, 6949, 7281,
  7618, 7960, 8308, 8660, 9017, 9378, 9744, 10114, 10487, 10864, 11244, 11627, 12014,
  12403, 12794, 13187, 13582, 13979, 14378, 14778, 15178, 15580, 15981
};
```

Add the `Motor_Drive` function. This function updates the driving signals and generates the sinusoidal drive at a given speed.

```
/* Function that is called every 50us to update the drive */
void Motor_Drive(void)
{
    /* Counters that scroll through the LUT at runtime. These counter are used to create
     * the 120 degrees phase shift between each of the motor's phases */
    static uint16_t     phase_a = DEGREES_TO_U16(0.0);
    static uint16_t     phase_b = DEGREES_TO_U16(120.0);
    static uint16_t     phase_c = DEGREES_TO_U16(240.0);
    static const uint16_t speed = SPEED;

    /* Values that will be written in the CMP channels of TCE */
    uint16_t drive_a, drive_b, drive_c;

    /* Updating the counters */
    phase_a += speed;
    phase_b += speed;
    phase_c += speed;

    /* Select new variables from the LUT for each CMP channel */
    drive_a = sine_lookup_table[(phase_a >> 8)];
    drive_b = sine_lookup_table[(phase_b >> 8)];
    drive_c = sine_lookup_table[(phase_c >> 8)];

    /* Update the values from CMP channels with new ones */
    TCE0_CompareChannels012BufferedSet(drive_a, drive_b, drive_c);
}
```

Add the `Mux_Set` function. This function switches the BEMF monitor from one phase to another because this example uses only one Analog Comparator.

```
/* Functions that switches the MUX to monitor all 3 phases at runtime */
void Mux_Set(uint8_t mode)
{
    uint8_t temp;
    temp = AC0.MUXCTRL;
    temp &= ~(AC_MUXPOS_gm | AC_MUXNEG_gm);
    temp |= mode;
    AC0.MUXCTRL = temp;
}
```

Add the `BEMF_Read` function. This function samples the BEMF signals and displays them on General Purpose Input/Output (GPIO) pins. This function also calls the `Mux_Set` and `Motor_Drive` functions.

```
/* Function that is called during the enlarged dead time to read the BEMF state */
void BEMF_Read(void)
{
    bool  bemf_state;
    static mux_t mux = MUX_PHASE_A;

    /* BEMF sampling point marked by an IO toggling */
    IO_PF4_SetHigh();
    bemf_state = ((AC0.STATUS & AC_CMPSTATE_bm) != 0 );
    IO_PF4_SetLow();

    /* Switching AC0 MUX from one phase to another one */
    switch(mux)
    {
        case MUX_PHASE_A: if(bemf_state) {IO_PF1_SetHigh();} else {IO_PF1_SetLow();}
                                          mux = MUX_PHASE_B; break;
        case MUX_PHASE_B: if(bemf_state) {IO_PF2_SetHigh();} else {IO_PF2_SetLow();}
                                          mux = MUX_PHASE_C; break;
        case MUX_PHASE_C: if(bemf_state) {IO_PF3_SetHigh();} else {IO_PF3_SetLow();}
                                          mux = MUX_PHASE_A; break;
        default: mux = MUX_PHASE_A; break;
    }

    /* Update Analog Comparator MUX to monitor another phase of the motor */
    Mux_Set(mux);

    /* Update drive to keep the motor spinning */
    Motor_Drive();
}
```

Edit the main.c file. Register the `BEMF_Read` function as a callback for the TCE CMP3 channel, enable the TCE hardware scaling accelerator, and set the desired amplitude level.

```
int main(void)
{
    SYSTEM_Initialize();

    /* Register the BEMF sampling function as a callback */
    TCE0_Compare3CallbackRegister(BEMF_Read);

    /* Enable hardware scaling accelerator after initialization to avoid messing up
     * the timing for reading BEMF with CMP3 channel of TCE */
    TCE0_ScaleEnable(true);

    TCE0_AmplitudeSet(AMPLITUDE);

    while(1)
    {
    }
}
```

11. Now, the project can be built and run from MPLAB X IDE. At run time, while the motor is spinning, the BEMF from each motor phase is measured with the AC and shown using some output GPIO pins. It can be observed that the measured BEMF follows the sinusoidal driving signals of the motor's phases.

# 6. BEMF Acquisition Demo Results

Below are illustrations of logic analyzer captures that show the sampling moment of BEMF during the dead time and how the sampled BEMF follows the driving signals of the motor's phases:

**Figure 6-1.** Sampling Point of BEMF During Dead Time

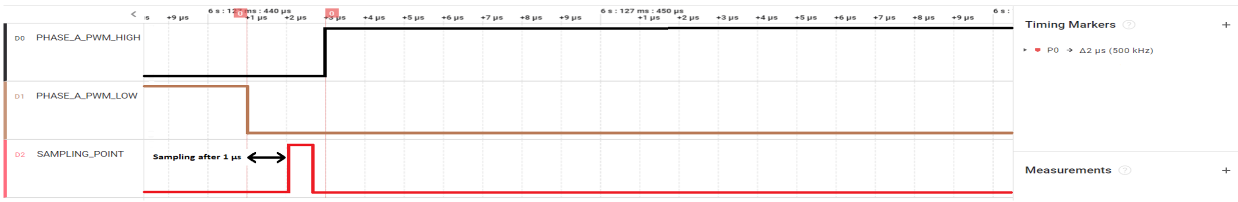



**Figure 6-2.** Sampled BEMF for One Phase of the Motor. Analog Comparator Output Vs. Sampled BEMF

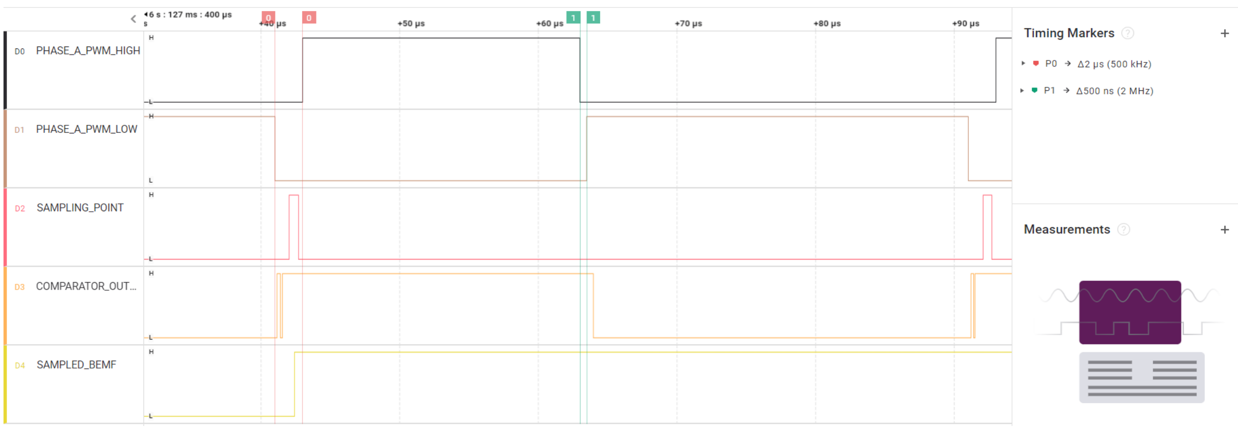
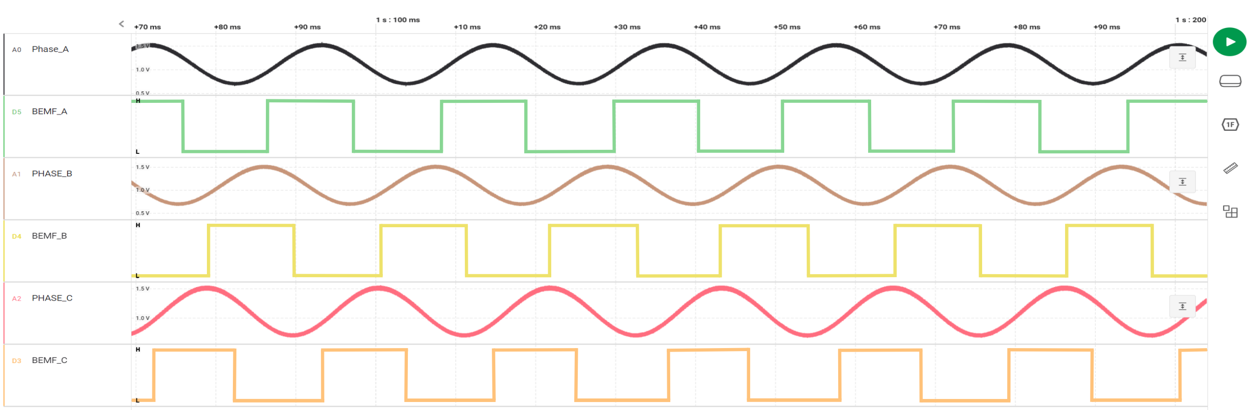



**Figure 6-3.** All Three Motor Phases with their Sampled BEMF Signals.

In this capture the phase signals are filtered with a hardware low-pass filter to observe the sinusoidal drive.




Here, one can find an MCC-generated code example for AVR16EB32 with the same functionality as the one described in this section:

Click to view code examples on MPLAB DISCOVER

# 7. Conclusions

This technical brief provides a method to measure the BEMF from the phases of a BLDC or PMSM motor driven in Sinusoidal Sensorless mode. Enlarging one of the dead times added for transistor switching creates a big enough acquisition window to measure clean BEMF signals. This method can be implemented using 8-bit MCUs that don't have all the capabilities of 16-bit and 32-bit cores.

When using this method, information about the rotor's position can be obtained every 60° when the BEMF's Zero-Cross signal is detected, and the drive is updated to keep synchronization with the motor. The highest efficiency is achieved when the BEMF is synchronized with the driving voltage because the voltage drop across the coils is minimum.

Compared to the method presented in this document, the FOC algorithm is still better because the rotor position information is obtained much more frequently (every PWM cycle). This new method, using dead time to measure BEMF, can be applied for low-end applications, where the control doesn't have to be very complicated, and the dynamic load variation is not too high.

Applications like fans and pumps can use this measurement method for BEMF detection and have a Sinusoidal Sensorless Drive. From a noise and efficiency aspect, the control obtained with this newly proposed method is still superior to the classical Trapezoidal Drive.

# 8. References


- "AVR16EB14/20/28/32 Preliminary Data Sheet" (DS40002522). Microchip Technology Inc., 2023
- "AVR16EB32 Curiosity Nano Pinout" Microchip Technology Inc. 2023
- Getting Started with Timer/Counter Type E (TCE) and WEX Microchip Technology Inc. 2023
- Multi-Phase Power Board (MPPB) Microchip Technology Inc. 2024
- AVR-EB Curiosity Nano Adaptor to MPPB Microchip Technology Inc. 2024

# 9. Revision History

| Document Revision | Date | Comments |
|---|---|---|
| A | 02/2024 | Initial document release |

# Microchip Information

## 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 products:

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

## Legal Notice

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

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

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

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

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, 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, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, 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, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, EyeOpen, GridTime, IdealBridge, IGaT, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, MarginLink, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mSiC, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, Power MOS IV, Power MOS 7, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, Turing, 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.



ISBN: 978-1-6683-3867-4

## Quality Management System

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

# Worldwide Sales and Service

## AMERICAS

**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

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Austin, TX**
Tel: 512-257-3370

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

**Raleigh, NC**
Tel: 919-844-7510

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110
Tel: 408-436-4270

**Canada - Toronto**
Tel: 905-695-1980
Fax: 905-695-2078

## ASIA/PACIFIC

**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

## ASIA/PACIFIC

**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

## EUROPE

**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