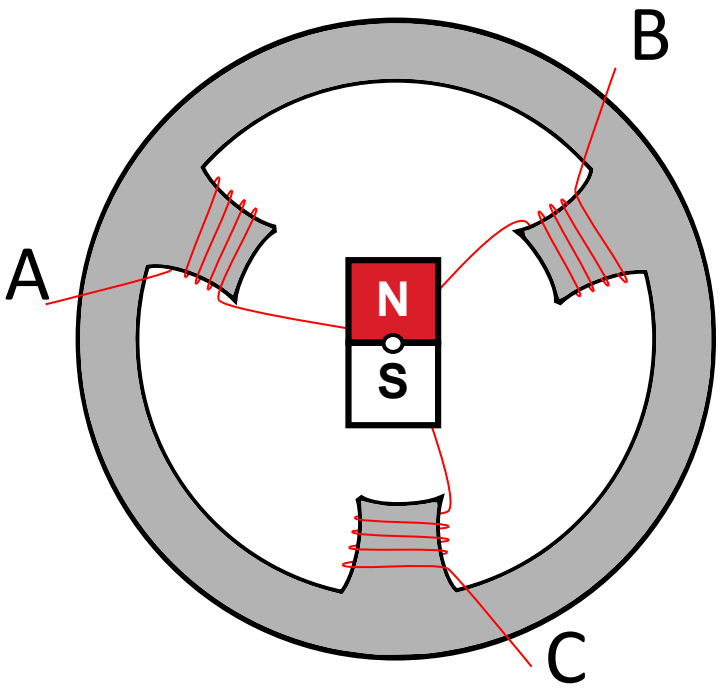
AN0816: EFM32 Brushless DC Motor Control

This application note shows how to drive a brushless dc (BLDC) motor with an EFM32 device. It includes firmware for a BLDC controller and a GUI tool to monitor and control the motor. Both Hall-sensor and sensorless control are supported.

The firmware is written to run on an EFM32GG-DK3750 or EFM32HG-SLSTK3400A board connected to the C8051F850 Motor Control Reference Design Board described in detail in *AN794: C8051F850 BLDC Reference Design Kit*. The firmware has also been ported to run on Series 2 devices, which is available on GitHub.

KEY POINTS

- An EFM32 can be used to drive brushless dc motors.
- This application note includes:
  - This PDF document
  - Source files (zip)
    - Example C code
    - Multiple IDE projects



1. Motor Operation

A brushless dc (BLDC) motor consists of a permanent magnet rotor and stator windings that are driven by an electric circuit. The following figure shows a simple view of a BLDC motor. In the figure, the central permanent magnet rotor is connected to a shaft and can move in response to the magnetic fields set up by the stator windings. The rotor can also be on the outside with the stator windings pointing radially outwards. The permanent magnet in the following figure has two poles (1 pole pair), but a typical BLDC can have rotors with several pole pairs.

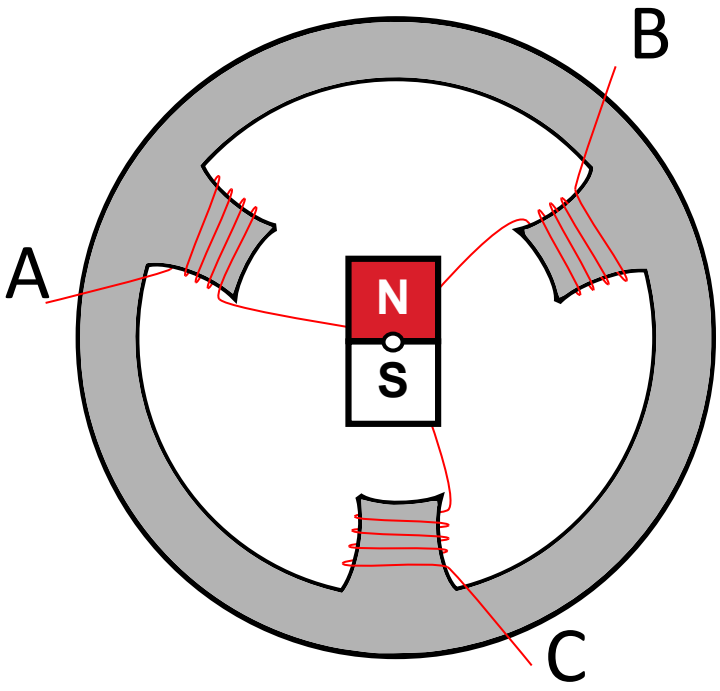


Figure 1.1. Brushless DC Motor

A BLDC motor has three terminals, normally labelled A, B and C. The terminals are connected to the stator windings in either a delta or Y configuration, see the following figure. In both configurations the motor is driven the same way. The rotor is moved by driving two of the terminals, which will cause the rotor to align with the magnetic field set up by the stator coils. When the controller continuously changes which terminals are driven the motor will rotate along with the field. A BLDC is similar to a stepper motor, but the difference is that while the stepper motor is designed for small accurate steps and high holding torque, a BLDC is designed to run faster and provide high torque at higher speeds.

The process of switching which terminals are driven is called commutation. This is similar to a brushed motor, where commutation is done mechanically by the brushes connecting to different coils as the motor rotates.

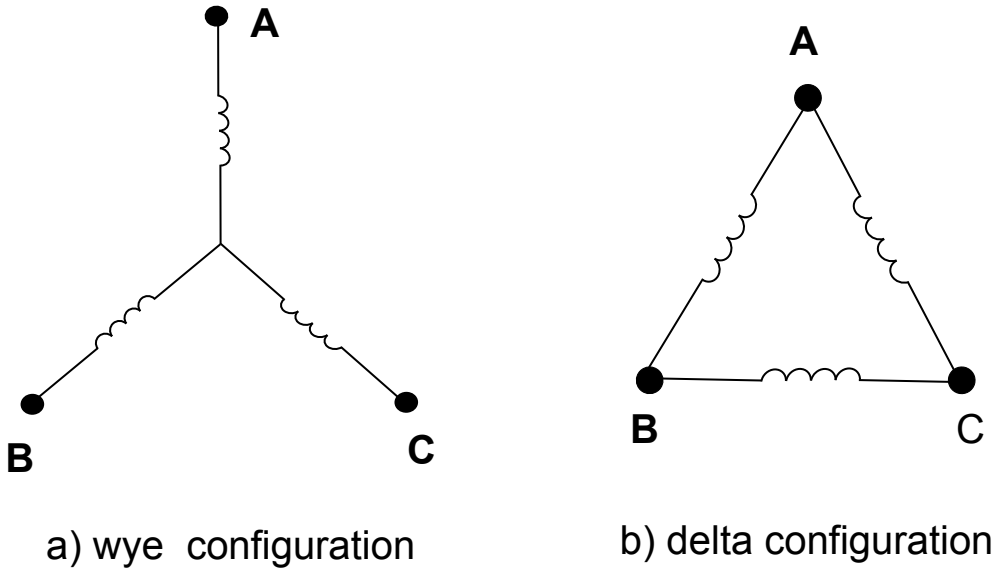


Figure 1.2. Winding Configurations

The motor terminals are driven by a three-stage inverter circuit shown in the following figure and each stage is connected to one of the motor terminals. Each stage of the inverter can drive the terminal voltage to the high or low side of the motor supply. When the low-side transistor of one stage is conducting, the terminal is driven low. When the high-side transistor is conducting the terminal is driven high.

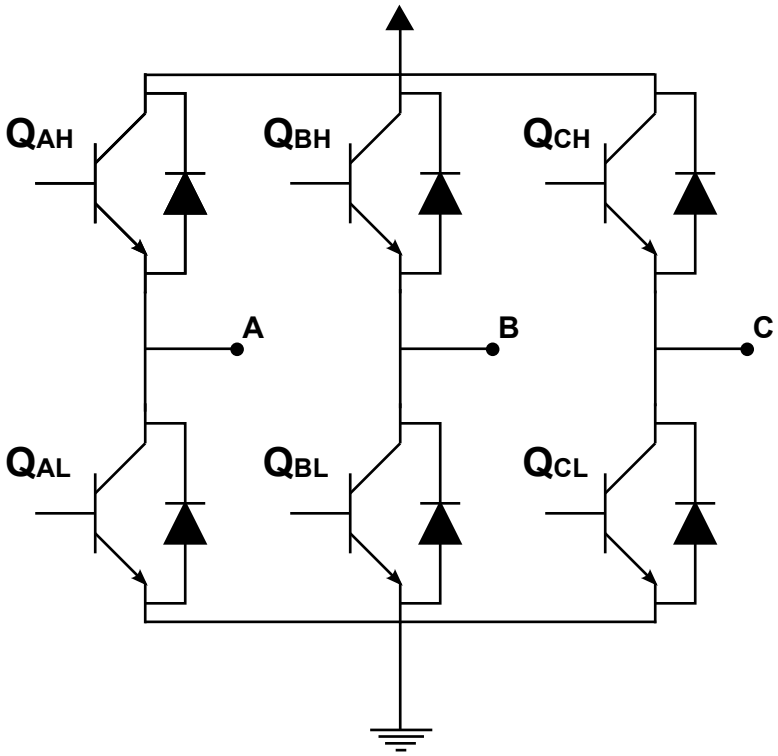


Figure 1.3. Inverter Circuit

The inverter circuit repeatedly goes through 6 different commutation steps, see the following table, which forms one electrical revolution. Note that one mechanical revolution requires multiple electrical revolutions if the permanent magnet rotor has more than one pole pair. The number of electrical revolutions per mechanical revolution is equal to the number of pole pairs of the motor.

Mechanical and Electrical Revolutions

$$\omega_{el} = n_p \omega_{mechanical}$$

Table 1.1. Commutation Sequence

Phase	AH	BH	CH	AL	BL	CL	Open
1 (0° - 60°)	on	off	off	off	on	off	C
2 (60° - 120°)	on	off	off	off	off	on	B
3 (120° - 180°)	off	on	off	off	off	on	A
4 (180° - 240°)	off	on	off	on	off	off	C
5 (240° - 300°)	off	off	on	on	off	off	B
6 (300° - 360°)	off	off	on	off	on	off	A

2. Determining Rotor Position

In order to drive the motor, the controller needs information about the current position of the rotor. More specifically, the controller needs to know at which point to switch to the next phase. Commutating at the correct time ensures that the maximum amount of energy is transferred to the motor. Commutating incorrectly reduces the efficiency of the motor and will apply a breaking effect and can even make the motor spin in the wrong direction. The optimal commutation point is when the rotor is positioned such that it is exactly half-way between any of the permanent magnet poles crossing a stator coil.

There are two popular methods for determining the correct commutation instant. The first is the use of Hall effect sensors to get the rotor position. Using Hall effect sensors is the simplest and most accurate method, but it comes with an extra cost since three Hall sensors are needed per motor.

For many applications, the Hall effect sensors are not strictly needed and it is quite common to get rid of these sensors and instead rely on measuring the back-EMF voltage in the open (undriven) terminal to determine the commutation instant. This method is often called sensorless commutation.

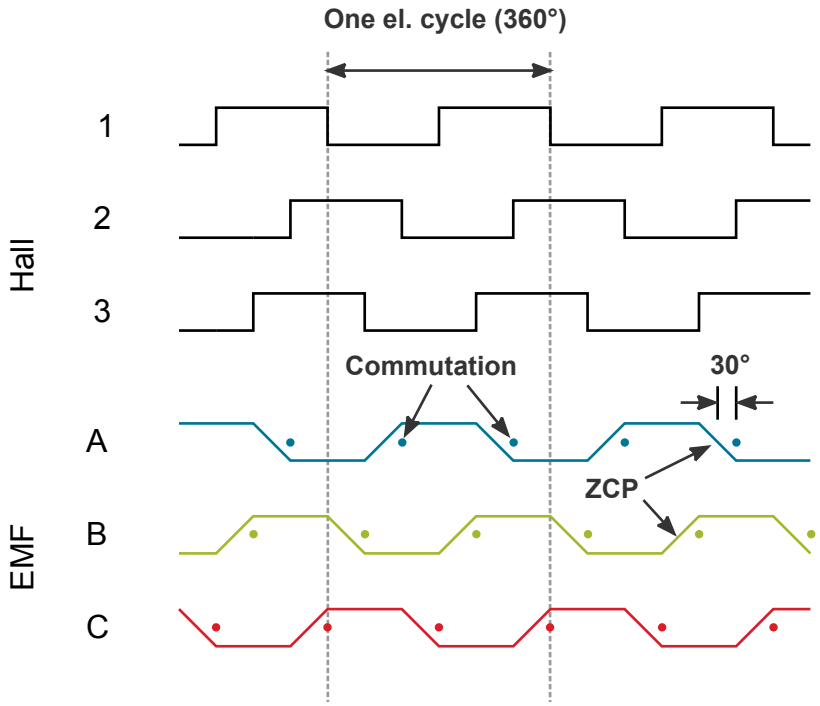


Figure 2.1. Hall Sensor Output and Back-EMF Voltage

2.1 Hall Effect Sensors

Hall effect sensors can measure the presence of a magnetic field. When using Hall effect sensors with a BLDC, three sensors are placed to cover 180 electrical degrees each, but with a 120 degrees phase shift between them. Reading the state of the three sensors are enough to decode the current phase (1-6) of the rotor for any orientation.

The commutation algorithm becomes very simple when using hall sensors. The optimal commutation points are exactly where the hall sensors change state. A GPIO interrupt is therefore enough to get the correct timing for each commutation. If the spinning direction is reversed, the Hall waveform in Figure 2.1 Hall Sensor Output and Back-EMF Voltage on page 5 is shifted 180° with respect to the terminal voltages.

2.2 Sensorless Commutation / Back-EMF Measurements

In any phase, two of the terminals are driven, while the third is open. As the permanent magnet passes by the open winding a voltage is induced, commonly referred to as the back-EMF. This voltage will either be rising or falling as the rotor moves, depending on whether a south or north pole is passing by. The Zero Crossing Point (ZCP) is when the magnet is directly adjacent to the stator winding. At this point the voltage at the terminal is the same as the motor neutral voltage, V_N.

Zero Crossing Point

$$V_A - V_N = 0$$

As can be seen from Figure 2.1 Hall Sensor Output and Back-EMF Voltage on page 5 the commutation instant is 30° after the ZCP. The algorithm to commutate with back-EMF measurements must first detect the ZCP, wait 30° and then switch to the next phase. The 30° offset can be determined by keeping track of the current speed and using a timer to trigger the commutation at a speed-adjusted time after each ZCP. The equation below gives the condition for a commutation event where t is the current time, t_{ZCP} is the last zero crossing point and T_{el} is the period of one electrical revolution.

Commutation Point with ZCP

$$t - t_{ZCP} = \frac{30°}{360°} T_{el} = \frac{T_{el}}{12}$$

One problem with this method is that the neutral point is not directly accessible since it is inside the motor. This can be solved by creating a "virtual neutral point" combined by the voltage at all three terminals, see the figure below.

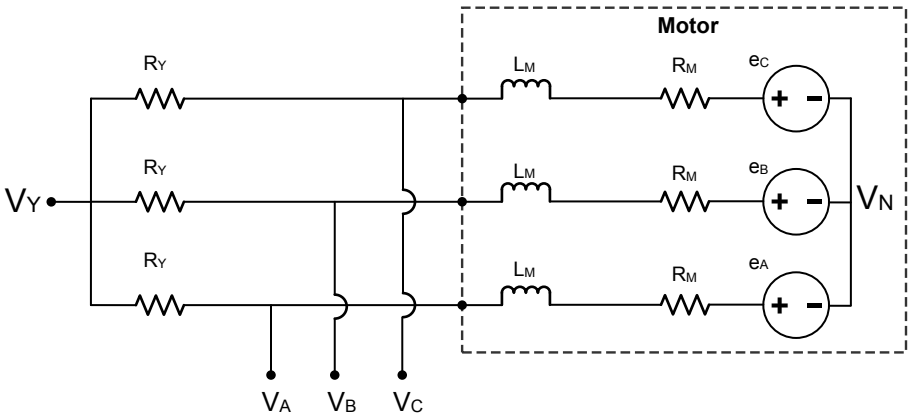


Figure 2.2. Virtual Neutral Circuit

ZCP using Virtual Neutral Point

$$V_A - V_Y = 0$$

2.3 Startup

When the motor is starting up from a rest state, there is no measurable back-EMF. Therefore the motor must be driven like a stepper motor until it reaches sufficient speed. Each phase is activated in turn and the commutation speed is slowly increased until the speed is high enough to switch to back-EMF based commutation. Since the position of the rotor is unknown when starting the motor, it is possible that the motor will move backwards before it starts moving in the forward direction.

The increase in speed must not be too fast or the motor will not be able to keep up. In this application note the time between commutation events is decreased so that the angular speed is increasing linearly. See *AN794: C8051F850 BLDC Reference Design Kit* for more information. Each commutation time T_k is calculated by:

Startup Commutation Intervals

$$T_{k+1} = T_k \frac{\sum_{i=0}^{k-1} T_i}{\sum_{i=0}^{k} T_i}$$

If hall sensors are used, they give the position of the rotor even when it is at rest. Starting a motor with hall sensors is much simpler. The controller can simply read the rotor position and keep moving to the next phase.

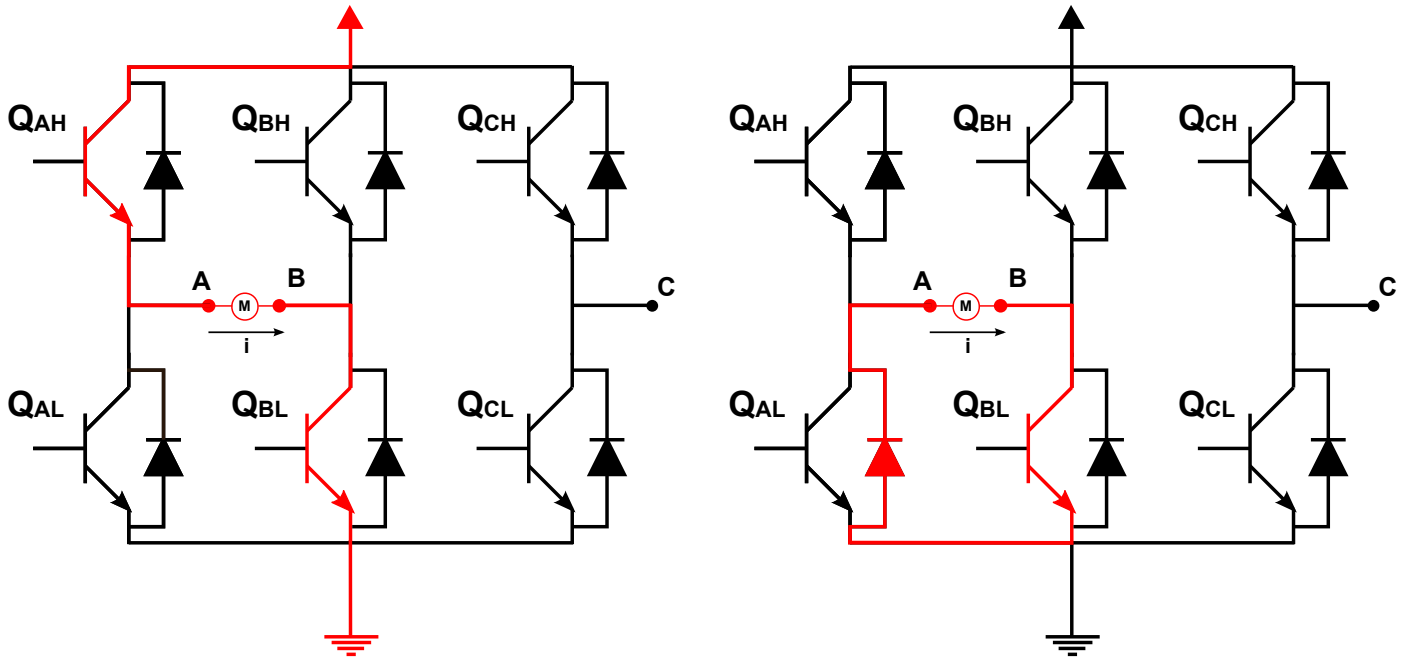
3. Speed Control

The speed of a BLDC motor is proportional to the voltage (for a given load). Instead of modulating the voltage directly a Pulse Width Modulated (PWM) waveform is often used. The PWM can be applied to the top, bottom or both transistors. In this application note only top-side PWM is used. The top transistors are fed a PWM waveform while the bottom transistors are turned on/off. A complementary PWM waveform is applied to the bottom transistor of the PWM fed stage to improve the performance.

The controller needs a way to get the current speed of the motor and a control loop to adjust the setpoint accordingly. In this application note, the speed is determined by measuring the time between commutation events and a PID controller is implemented.

3.1 Complementary PWM with Dead-Time Insertion

During the PWM on period, current flows through the upper transistor of one inverter stage, through the motor windings and finally through the lower transistor of another inverter stage, see the following figure, part (a). At the start of the PWM off period, the inductance in the motor windings will cause the current to keep flowing. Most of this current will go through the body diode of the first invertor stage, see the following figure, part (b). This recirculation current will decay and eventually reach zero if the PWM period is long enough.



a) PWM ON phase. Current flowing through high side of one stage, the motor and bottom side transistor of second stage.

b) PWM OFF phase. Recirculation current through body diode of first stage, the motor and bottom side transistor of second stage.

Figure 3.1. Current Flow in PWM ON/OFF Phases

The recirculation current will cause heat buildup when flowing through the transistor body diode. It also reduces the performance of the PID controller, especially when braking since the controller has less control over the current through the motor. The recirculation current also reduces the accuracy of the ZCP since it creates an offset voltage at the motor terminal.

In order to reduce the impact of the recirculation current, the bottom transistor of the first inverter stage can be turned on during the PWM off phase. I.e. the bottom transistors are being fed a complementary PWM signal to the top transistors. The recirculation current goes through the channel of the bottom transistor instead of the body diode. Heat dissipation is reduced and the recirculation current decays to zero quickly.

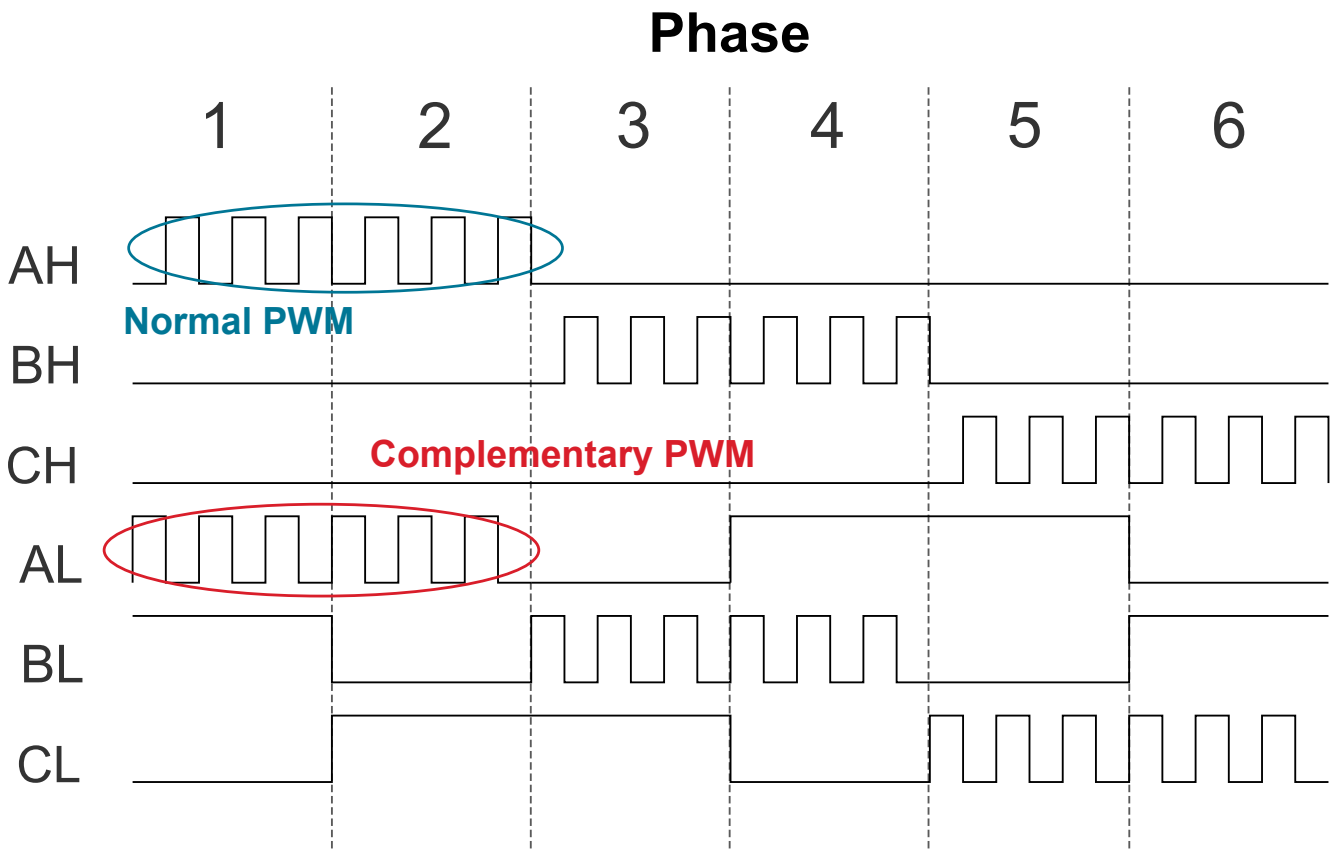


Figure 3.2. Complementary High-Side PWM Waveforms

With a complementary PWM scheme, it is necessary with some dead time between the two PWM signals. The reason is that the transistors do not turn off instantly, but do have some response time before they switch on/off. If no dead time is inserted there could be a short time every PWM cycle where both transistors in the same stage are ON and the circuit would effectively short the power supply to ground.

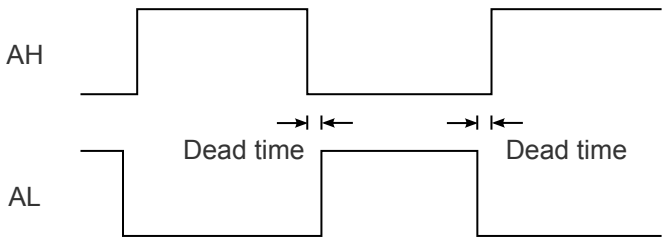


Figure 3.3. Complementary PWM with Dead Time

3.2 PID Controller

In order to control the speed of the motor, a control loop is needed. In this application note a PID controller is implemented. A PID controller tries to correct the error in the measured speed against a setpoint by adjusting the input (PWM) to the motor.

A PID controller has 3 parameters:

- K_p – proportional gain
- K_i – integral gain
- K_d – derivative gain

The correction is calculated by:

PID Equation

$$u(t) = K_p e(t) + K_i \int_0^{\tau} e(t)d\tau + K_d \frac{d}{dt} e(t)$$

where e(t) is the error at time t.

4. Application Overview

Included in this application note is firmware for running a BLDC controller on an EFM32 MCU. The code is intended for running on a EFM32GG-DK3750 Development Kit or EFM32HG-SLSTK3400A Starter Kit connected to a C8051F850 Motor Control Reference Design Board, discussed in detail in *AN794: C8051F850 BLDC Reference Design Kit*.

Additional firmware for running a BLDC controller on Series 2 devices is available on the Silicon Labs Platform Applications Github.

4.1 BLCD Reference Design

The Silicon Labs C8051F850 Motor Control Reference Design Board is a ready-to-use hardware platform for developing BLDC motor control applications. It comes with a C8051F850 MCU plugin board and firmware for the C8051F850. See *AN794: C8051F850 BLDC Reference Design Kit* for a full description of the board and schematics. The reference design consists of three parts: a C8051F850 MCU plugin board, the powertrain board and a BLDC motor.

Figure 4.1. C8051F850 Motor Control Reference Design Board

The EFM32 MCU is connected directly to the powertrain board. The C8501F850 MCU Plugin Board should be disconnected and a standard 0.1" pitch header soldered on to the 1x22 header footprint. The following table shows how to connect the MCU to the powertrain board.

Table 4.1. Motor Connection

Function	Powertrain	EFM32GG DK3750/MCU	EFM32HG SLSTK3400A/MCU
Q_AH Enable	PWM0A	PA0	PF0
Q_BH Enable	PWM1A	PA1	PF1
Q_CH Enable	PWM2A	PA2	PF2
Q_AL Enable	PWM0B	PA3	PF3
Q_BL Enable	PWM1B	PA4	PF4
Q_CL	PWM2B	PA5	PF5
V_A Terminal Voltage	VMA	PC1	PC0
V_B Terminal Voltage	VMB	PC6	PC1
V_C Terminal Voltage	VMC	PC7	PC2
V_Y Virtual Neutral	VMY	PC0	PC4
Current sense resistor +	IM_0P	PD6	PD6
Current sense resistor -	IM_0N	PD7	PD7
Hall Sensor 1[1]	N/A	PB0	N/A
Hall Sensor 2[1]	N/A	PB1	N/A

Function	Powertrain	EFM32GG DK3750/MCU	EFM32HG SLSTK3400A/MCU
Hall Sensor 3[1]	N/A	PB2	N/A
Hall Sensor VDD[1]	N/A	PB3	N/A
Gate Drive Enable	GDx_EN	GND	GND

Note:
1. Optional. Only if using hall sensors.

4.2 BLDC Controller Firmware

The included firmware enables simple control of a BLDC motor. Both sensorless and hall sensor driven motors are possible.

The motor is controlle with the buttons PB0/PB1 on the EFM32HG-SLSTK3400A.

- PB0 – Stop the motor or change direction:
  - If the motor is stopped, this button changes direction.
  - If the motor is running, this button stops the motor.
- PB1 – Start the motor or adjust speed:
  - If the motor is stopped, this button starts the motor.
  - If the motor is running, a long press of this button will decrease the speed of the motor, and a short press of this button will increase the speed of the motor.

The motor is controlled with the buttons PB1/PB2 on the EFM32GG-DK3750.

- PB1 – Stop the motor or change direction:
  - If the motor is stopped, this button changes direction.
  - If the motor is running, this button stops the motor.
- PB2 – Start the motor or adjust speed:
  - If the motor is stopped, this button starts the motor.
  - If the motor is running, a long press of this button will decrease the speed of the motor, and a short press of this button will increase the speed of the motor.

4.2.1 Configuration

Configuration is done at compile time in `config.h`. If another motor is used, the `MOTOR_POLE_PAIRS` must be updated to reflect the number of pole pairs. Here it is possible to change between sensorless or hall driven commutation or enable/disable the use of complementary PWM. A few other timing parameters can also be changed here. See the comments in `config.h` for details.

4.2.2 EFM32 Resources

The controller uses three TIMERs. TIMER0 is used to generate PWM with DTI for the inverter control. The overflow interrupt on TIMER0 is also used to invoke the PID routine. TIMER1 is used to measure the time between commutations and thereby calculating the motor speed. TIMER2 is used to synchronize the ACMP and ADC measurement with the PWM waveform (if they are used).

The ACMP is used in sensorless mode to measure the ZCP. The ACMP is triggered at the same point in the PWM waveform by the CC1 interrupt on TIMER2.

The ADC is (optionally) used to measure the motor current by measuring the voltage across a current sense resistor in series with the inverter circuit. The ADC measurement is triggered half-way in the PWM 'ON' by TIMER2 through a PRS channel. If an overcurrent condition is met (the measured current is higher than the configured maximum) the motor will stop. The motor current can also be logged to the PC tool for monitoring.

UART1 of EFM32GG Giant Gecko or UART1 of EFM32HG Happy Gecko is used for logging and control with the PC tool. UART1 is connected to the on-board RS-232 connector on the EFM32GG-DK3750. USART1 can be connected to the expansion header of the EFM32HG-SLSTK3400.

4.3 PC Tool

A PC tool to log the response and control the motor is included. The `efm32_bldc.exe` is found in the application note directory. It can plot the speed, PWM and current draw while the motor is running. There are also controls to to start/stop the motor and set the PID coefficients and setpoint.

The PC tool communicates with the MCU over UART, so a USB-to-UART adapter is needed. For EFM32GG Giant Gecko, connect either a USB-RS232 cable to the RS232 port on the EFM32GG-DK3750 or use a USB-to-TTL-UART and connect this to PB9/PB10 (MCU TX/RX). For EFM32HG Happy Gecko, use a USB-to-TTL-UART and connect this to PE10/PE11 (MCU TX/RX). The baud rate must match the configuration in `config.h`.

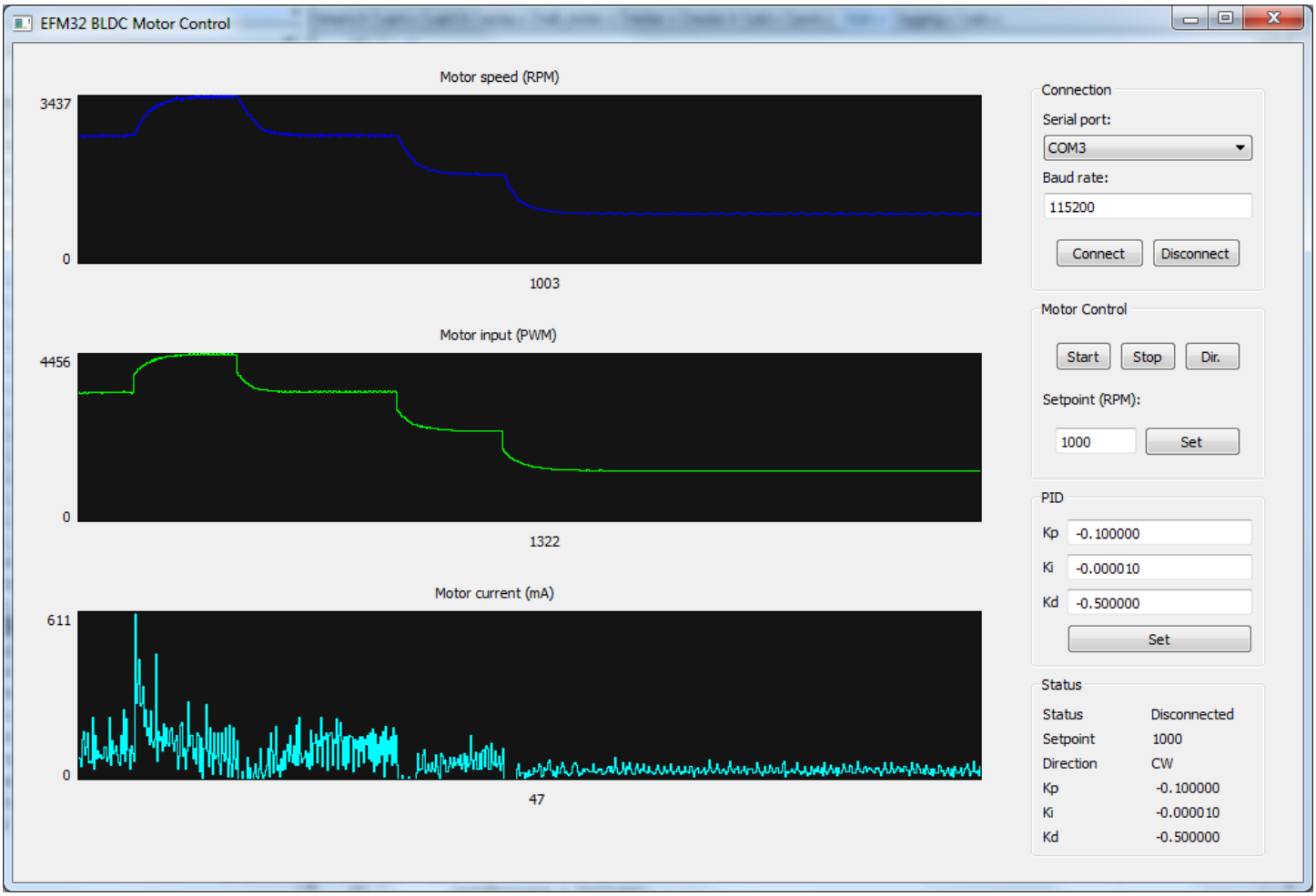


Figure 4.2. BLDC Control/Logging PC Tool

5. References

See the following documents for more information:

1. *AN794: C8051F850 BLDC Reference Design Kit* — This document contains detailed descriptions of the motor control board and its operation, including schematics.

6. Revision History

Rev. 1.2

2025-01-10

Added link for Series 2 GitHub example.

Rev. 1.1

2017-03-08

Updated formatting.

Added support for EFM32HG.

Rev. 1.0

2014-05-23

Initial revision.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com