C8051F850 BLDC Reference Design Kit

1. Introduction

This design kit provides a complete system-level solution for sensorless, brushless dc (BLDC) motors. This application note includes complete schematics, PCB layouts and firmware.

This kit supports 3-phase BLDC motors that meet the following specifications:

- Trapezoidal back-EMF
- Max operating voltage of between 10 V to 24 V dc
- 24 kHz PWM frequency
- Maximum average current of 10 A
- Maximum speed not exceeding 200,000 rpm for a 2-pole BLDC motor
- Overcurrent detection capability stops the motor when average current exceeds 10 A.
- Motor Stall detect capability stops the motor when a motor stall is detected or extreme loads encountered.
- Tachometer Frequency Generator (FG) output signal

The kit aims to demonstrate the capabilities of the C8051F850 for operating BLDC motors. The unique features offered by this MCU for BLDC motor operation are:

- PWM synchronized blanking of comparator for BEMF Zero-Crossing Detection
- Automatic PWM duty cycle reduction to limit motor current during startup
- Hyperdrive mode to increase the maximum speed of some motors

1.1. Kit Contents

The kit consists of the following:

- One MCU Board: MCRD-MCU-C8051F850 with the motor control firmware preprogrammed into the MCU
- One Powertrain Board: MCRD-PWR-NLV-F85X
- One BLDC Motor: Turnigy 450 Series 3800 kV Brushless Outrunner Helicopter Motor
- One Motor Mount Board
- One 8-bit MCU Kit CD
- One 12 V, 5 Amp Universal Input Power Adapter

1.2. Operational Specifications

The kit uses the Turnigy 450 Series 3800 kV Brushless Outrunner Helicopter Motor.
The motor operating range and specifications are given in Table 1.

Table 1. Motor Specifications

Parameter	Min	Typ	Max	Unit
Number of poles	—	6	—	
Operating voltage	7.4	—	14.8	V
Maximum current	—	—	35	A
Maximum power	—	—	365	W
No-load full-speed average current @ 12 V	—	3.66	—	A
No-load peak startup current @ 12 V	—	—	11	A
Motor constant	—	3800	—	RPM/V
Maximum speed @ 12 V	—	—	45600	RPM
Weight	—	80	—	g

The operating ranges of the hardware and firmware in the kit are shown in Table 2.

Table 2. Hardware and Firmware Operating Range

Parameter	Min	Typ	Max	Unit
Power supply	10	—	24	V
Motor driver PWM frequency	—	24	—	kHz
Continuous average output current	—	—	10	A
Speed range (2-pole motor)	2250	—	200000	RPM
Speed range (4-pole motor)	1125	—	100000	RPM
Speed range (6-pole motor)	750	—	66667	RPM

In the default operation mode using the ac/dc adapter, the operating parameters of the motor are shown in Table 3.

Table 3. Reference Design Kit Operating Specifications

Parameter	Value	Unit
Power supply	12	V
No-load full-speed average current @ 12 V	3.66	A
No-load peak startup current @ 12 V	11	A
Maximum speed @ 12 V	45600	RPM
Motor driver PWM frequency	24	kHz
Maximum Power	43.92	W

2. Theory of Operation

This section describes the theory of operation of 3-phase BLDC motors so that users of this design kit can understand the design choices they may face in their application.

2.1. System Model

Firstly, we will describe the system model of 3-phase BLDC motor including its drive system. This will help us understand the behavior of the system and design the appropriate drive systems and filters for our application.

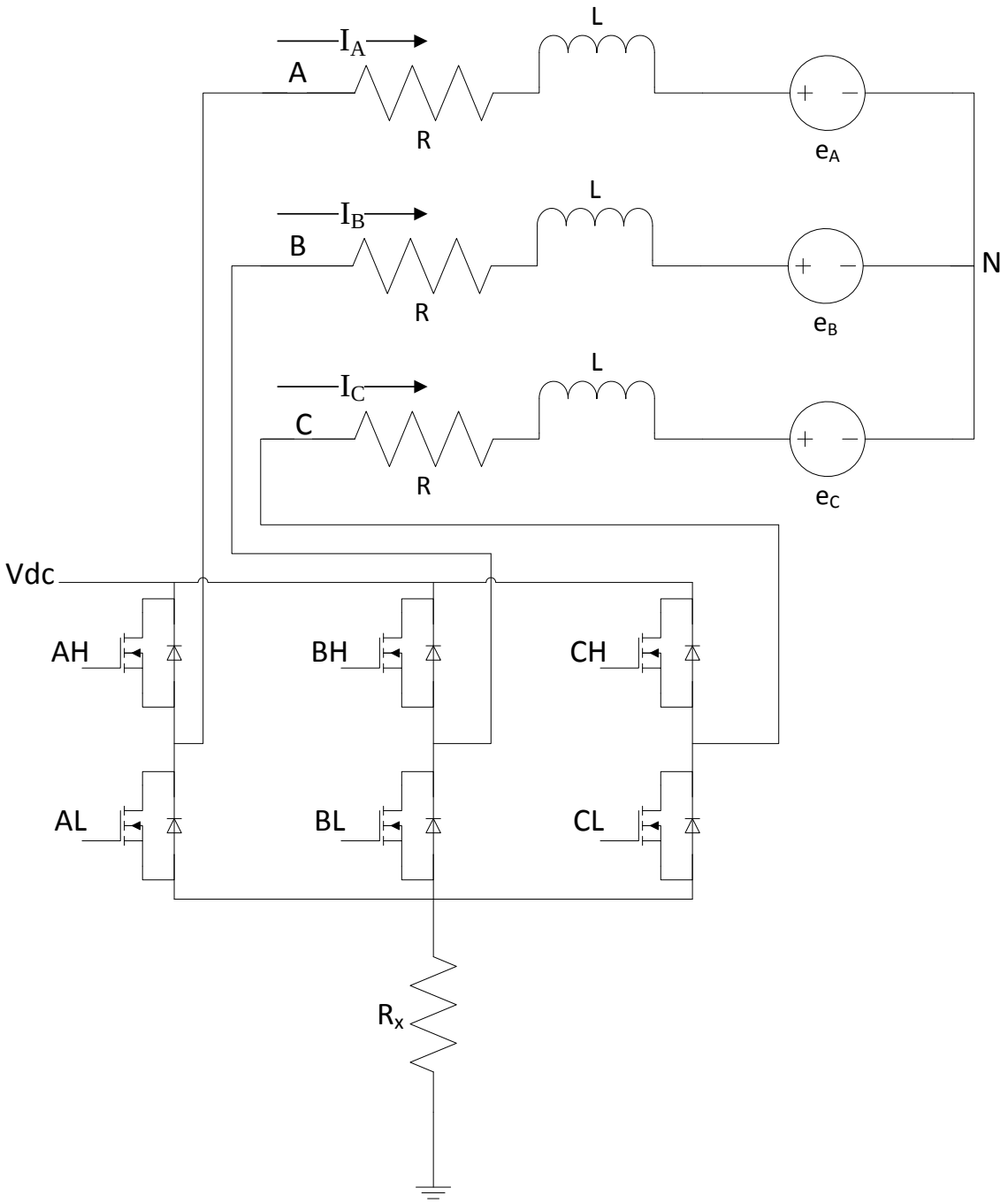


Figure 1. System Model of 3-Phase BLDC Motor Drive

A BLDC motor has 3 stator windings and is driven by an inverter circuit that consists of 6 switches. Figure 1 shows the equivalent circuit of a Y-connection BLDC motor and the inverter circuit topology. In this model, the stator inductance and resistance of each phase are assumed to be equal. RX is a very small valued resistor used for current measurement and can be assumed to be zero to simplify the analysis of the different modes of operation.

The electrical equations of the system model can be expressed by the following equation:

$$V_A - V_N = I_A R + L\frac{dI_A}{dt} + e_A \quad (1)$$

$$V_B - V_N = I_B R + L\frac{dI_B}{dt} + e_B \quad (2)$$

$$V_C - V_N = I_C R + L\frac{dI_C}{dt} + e_C \quad (3)$$

$$e_A = K\omega_m F\theta_e \quad (4)$$

$$e_B = K\omega_m F\left(\theta_e - \frac{2\pi}{3}\right) \quad (5)$$

$$e_C = K\omega_m F\left(\theta_e - \frac{4\pi}{3}\right) \quad (6)$$

$$\omega_m = \frac{2}{N_p}\frac{d\theta_e}{dt} \quad (7)$$

Where:

V_A, V_B, V_C denote the voltages of motor terminals A, B, and C, respectively

I_A, I_B, I_C denote the phase currents entering terminals A, B, and C, respectively

e_A, e_B, e_C denote the phase back-EMF (BEMF) associated with terminals A, B, and C, respectively

R is the stator phase resistance

L is the stator phase inductance

V_N is the neutral voltage of the Y connection

K is the motor constant

N_P is the number of poles in the motor

θ_e is the electrical angle of the motor

ω_m is the angular speed of the motor

$F(\theta_e)$ represents the BEMF reference wafeform as a function of rotor electrical angle

The electromagnetic torque is described by the following equation:

$$T_e = KI_A F(\theta_e) + KI_B F\left(\theta_e - \frac{2\pi}{3}\right) + KI_C F\left(\theta_e - \frac{4\pi}{3}\right) \quad (8)$$

BLDC and PMSM (permanent magnet synchronous motor) motors are differentiated by the BEMF reference waveform F(θe). BLDC motors are identified by their trapezoidal BEMF reference waveforms on each phase as shown in Figure 2.

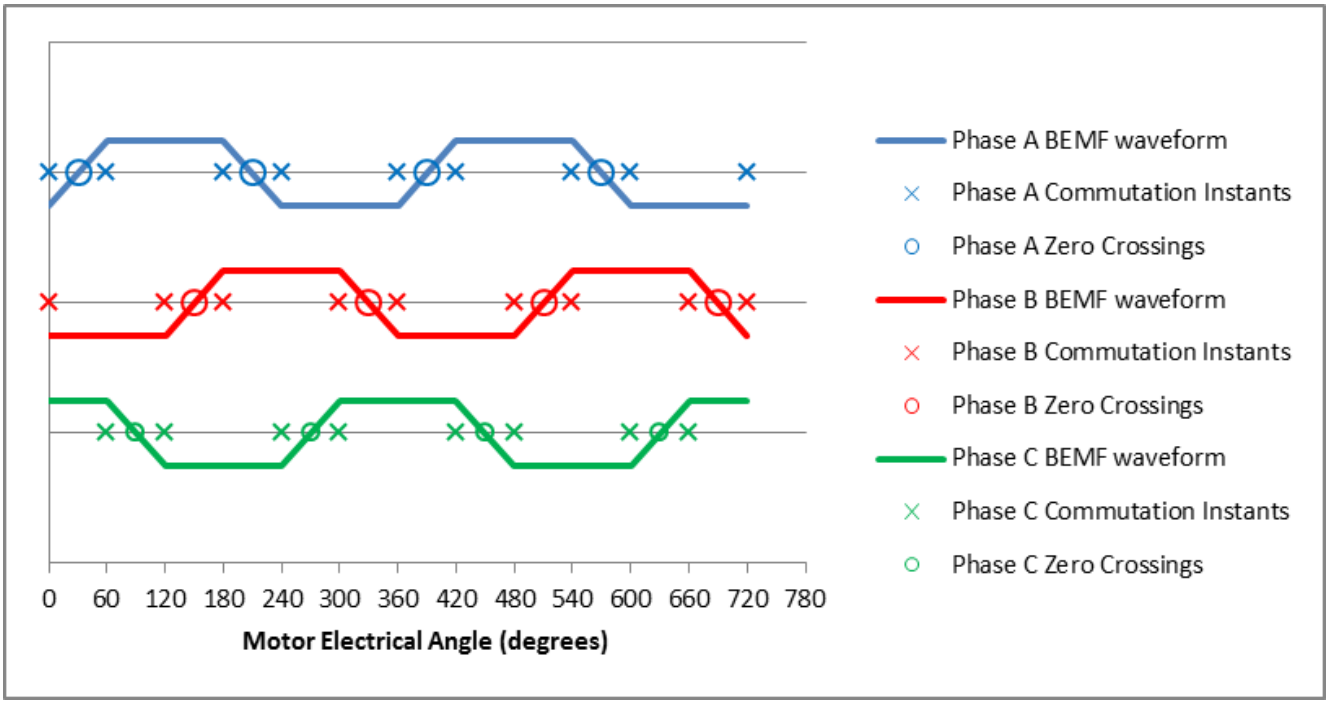


Figure 2. BEMF Reference Waveforms of all Phases

2.2. Driving 3-Phase BLDC Motors

There are many methods of driving a 3-phase BLDC motor. For the simple 8-bit MCU implementation, block commutation is used to drive the motor. This method of driving requires the inverter circuit to commutate the current every 60 degrees of the motor phase electrical angle according to the rotor position given by Hall sensors or a sensorless method. This is performed efficiently by adjusting the commutation sequence to synchronize with the BEMF reference waveforms as listed in Table 4.

Table 4. Commutation Sequence for Block Commutation

Commutation Phase	Motor Phase Electrical Angle (Degrees)	Gate Drive Outputs						Open Phase
		High-Side			Low-Side			
		AH	BH	CH	AL	BL	CL	
0	0–60	Off	Off	On	Off	On	Off	A
1	60–120	On	Off	Off	Off	On	Off	C
2	120–180	On	Off	Off	Off	Off	On	B
3	180–240	Off	On	Off	Off	Off	On	A
4	240–300	Off	On	Off	On	Off	Off	C
5	300–360	Off	Off	On	On	Off	Off	B

The conducting interval of each phase is 120°, and only two phases are conducted at any time. The motor speed can be controlled by applying PWM to either the high side or low side MOSFETs during each commutation phase. There are many PWM schemes that can be applied to control the current. The following sections described the different schemes.

2.2.1. High-Side PWM

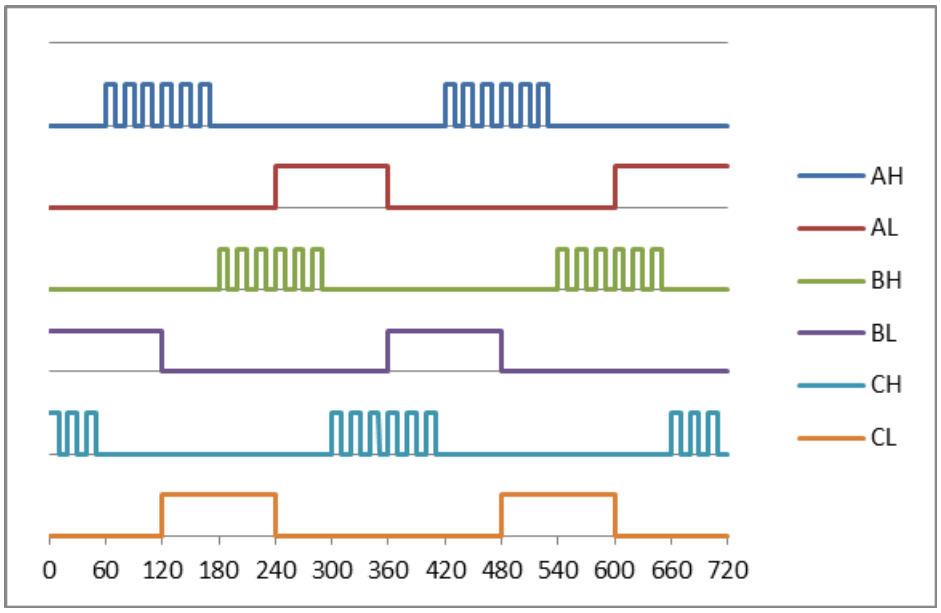


Figure 3. High-Side PWM Scheme

High-side PWM can be used if the high-side MOSFETs are N-channel enhancement type MOSFETs. It is not advisable to apply this scheme to P-channel MOSFETs because they are typically slower than N-channel MOSFETs. The advantage of high-side PWM scheme is that it is simple to implement since it requires only 1 PWM signal to be active at any time. Another advantage is that the bootstrap capacitor of a typical high-side driver of an N-channel MOSFET is guaranteed to get enough charge when its low-side MOSFET is enabled. The disadvantage of this scheme is heat built-up by re-circulation current through the body diodes of the low-side MOSFETs during the PWM off cycle can cause problems if these MOSFETs cannot dissipate the heat. The sequence of current flow through the inverter circuit is illustrated in Figures 4, 5, and 6.

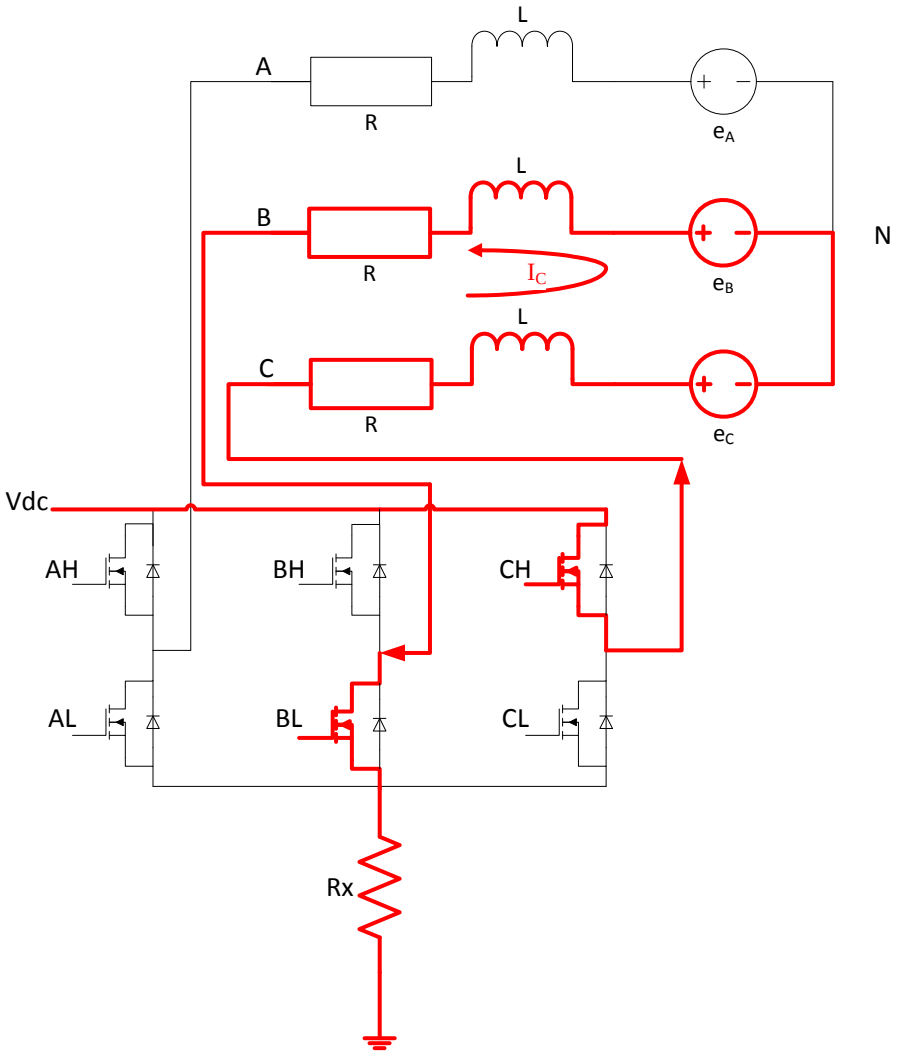


Figure 4. Current Flow: High-Side PWM, PWM On Cycle

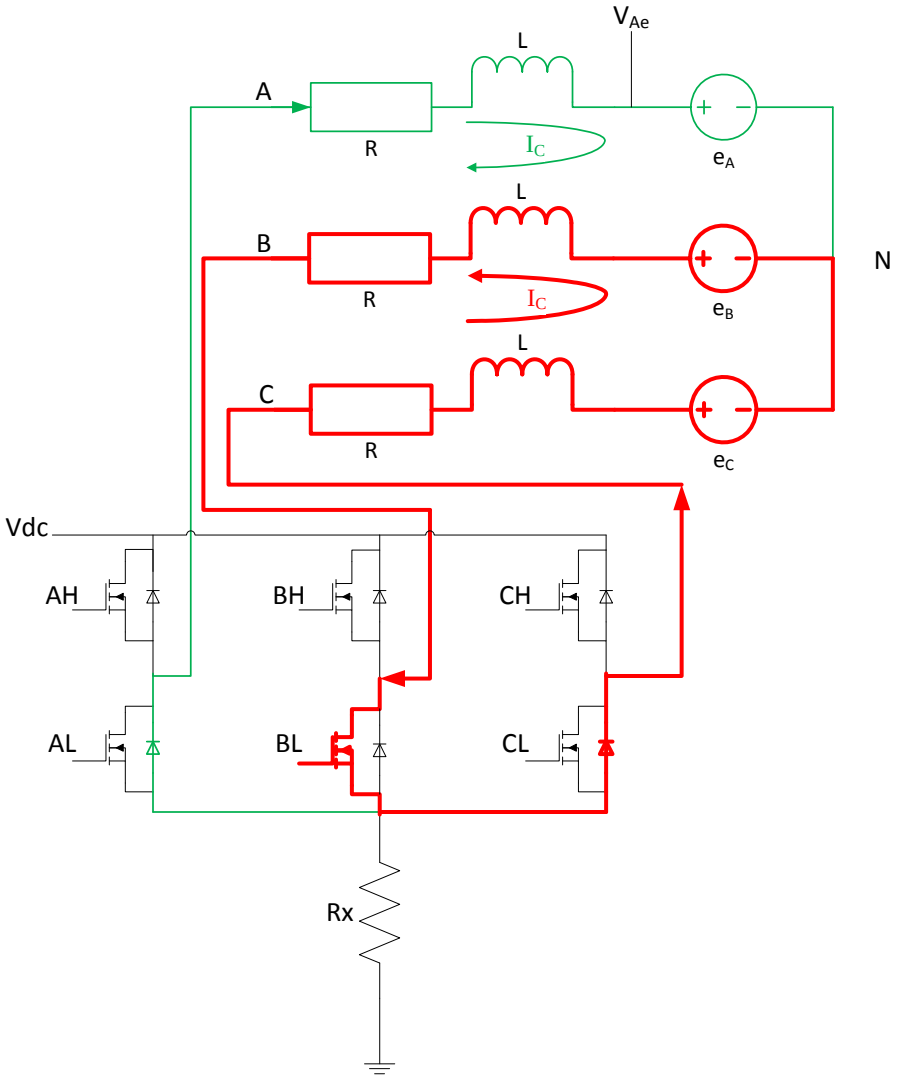


Figure 5. Current Flow: High-Side PWM, Start (Part 1) of PWM Off Cycle

At the start of the PWM off cycle, current can flow in the open terminal. This effect will be explained in the next section. When the low level current in the open terminal decays to zero, only the recirculation current remains flowing as shown in Figure 6.

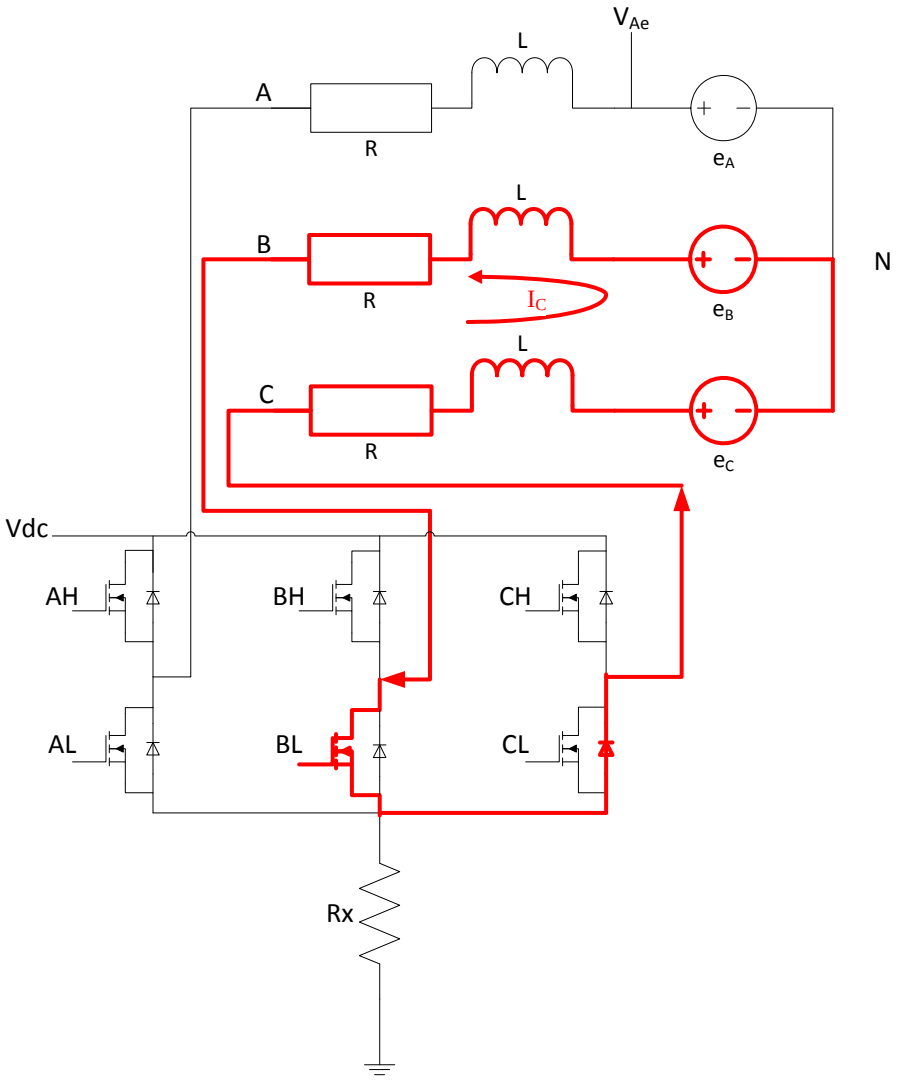


Figure 6. Current Flow: High-Side PWM, (Part 2) of PWM Off Cycle

If the PWM off cycle is sufficiently long, the remaining recirculation current will decay to zero. The oscilloscope capture in Figure 7 shows the voltage of an open terminal for all the phases of the PWM off cycle:

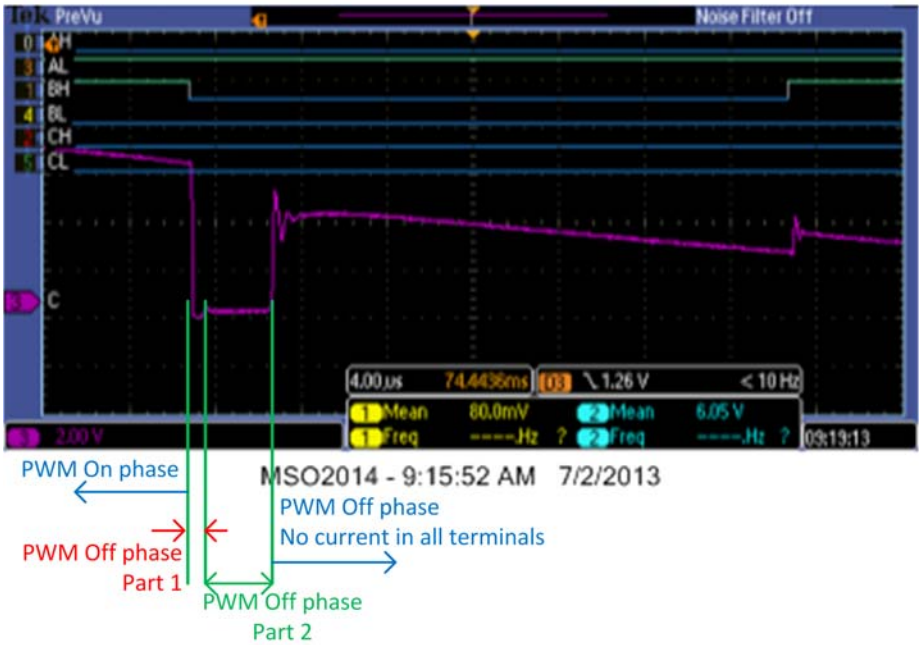


Figure 7. Oscilloscope Capture of Open Terminal

Note the slight dip in voltage during “PWM Off phase Part 2”. This corresponds to the period when a small current flows in the open terminal.

2.2.2. Low-Side PWM

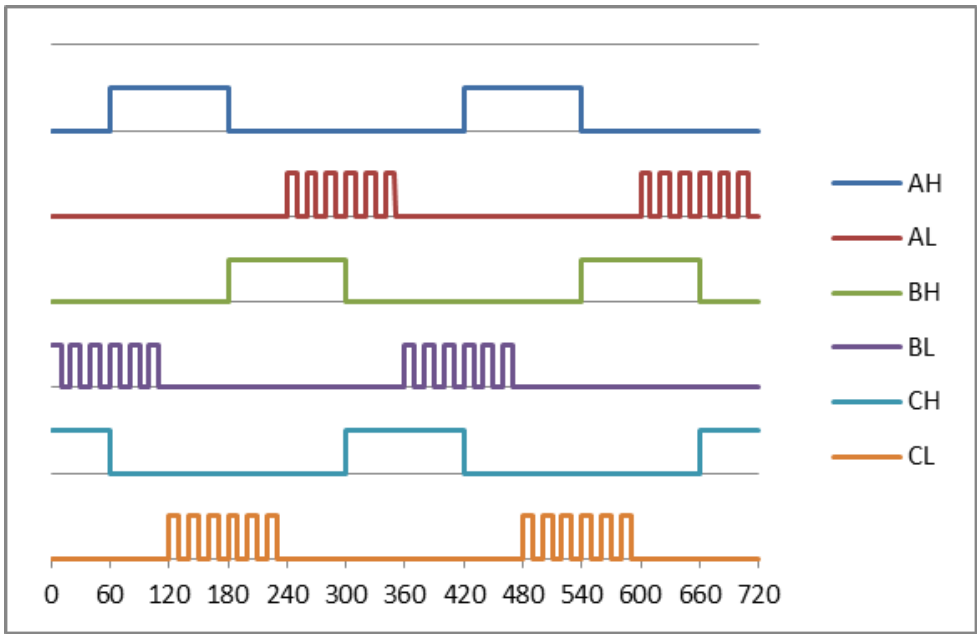


Figure 8. Low-Side PWM Scheme

Low-side PWM is typically used in low-cost designs where the high-side MOSFETs are P-channel type MOSFETs with simple discrete drivers where high-side PWM would not be suitable. As in the high-side PWM scheme, low-side PWM is simple to implement as it requires only one PWM signal to be active at any time. Like the high-side PWM, the disadvantage is the potential heat problems due to the recirculation current.

2.2.3. Complementary High-Side PWM

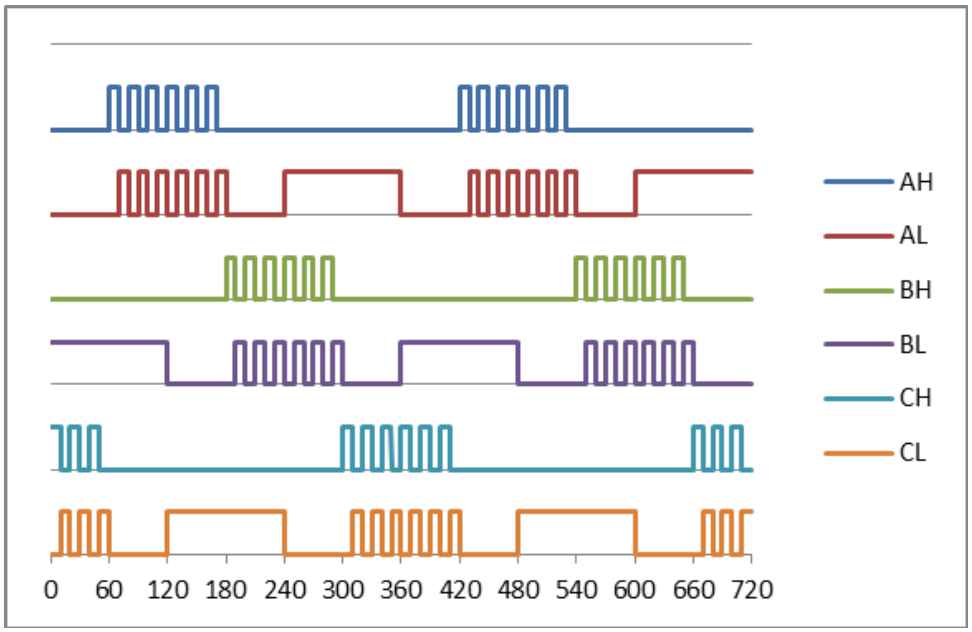


Figure 9. Complementary High-Side PWM

Complementary high-side PWM should only be applied when the high-side MOSFETs are N-channel types. The scheme resolves the heat problems mentioned earlier because the re-circulation current now passes through the low-side MOSFET channel instead of the body diode. The disadvantage is that it is more complicated to implement as it requires two complementary PWM signals with dead time between the enable periods of the two signals. Another disadvantage is that this method is not suitable for applications where the load is light; this causes the recirculation current during the PWM off cycle to decay to zero quickly and flow in the opposite direction, effectively applying a braking effect on the motor, which lowers the efficiency of the motor.

2.2.4. Complementary Low-Side PWM

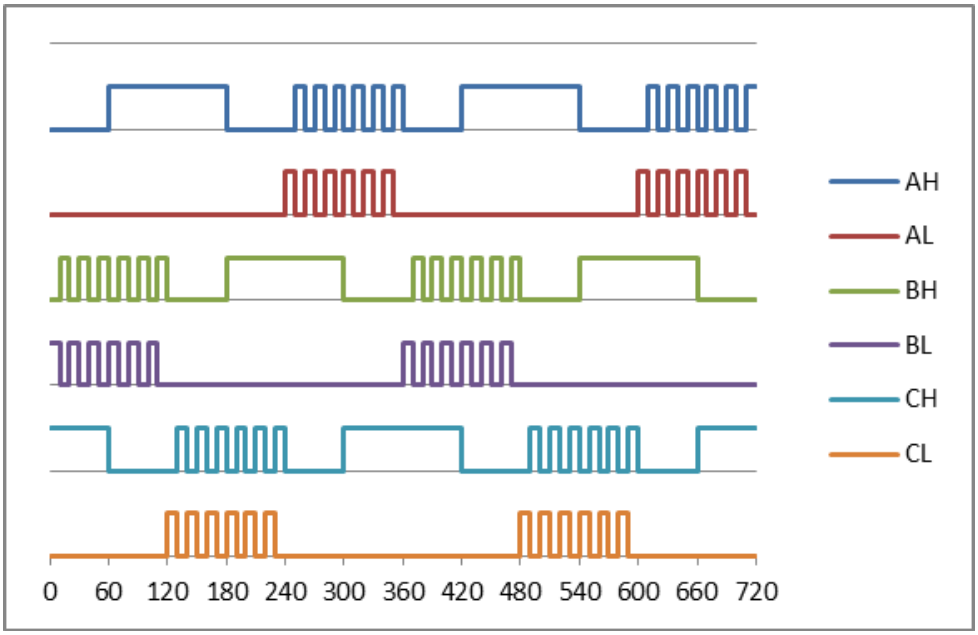


Figure 10. Complementary Low-Side PWM

Complementary low-side PWM should only be applied when the high-side MOSFETs are N-channel type. It has similar advantages and disadvantages as complementary high-side PWM.

2.2.5. Mixed Mode PWM

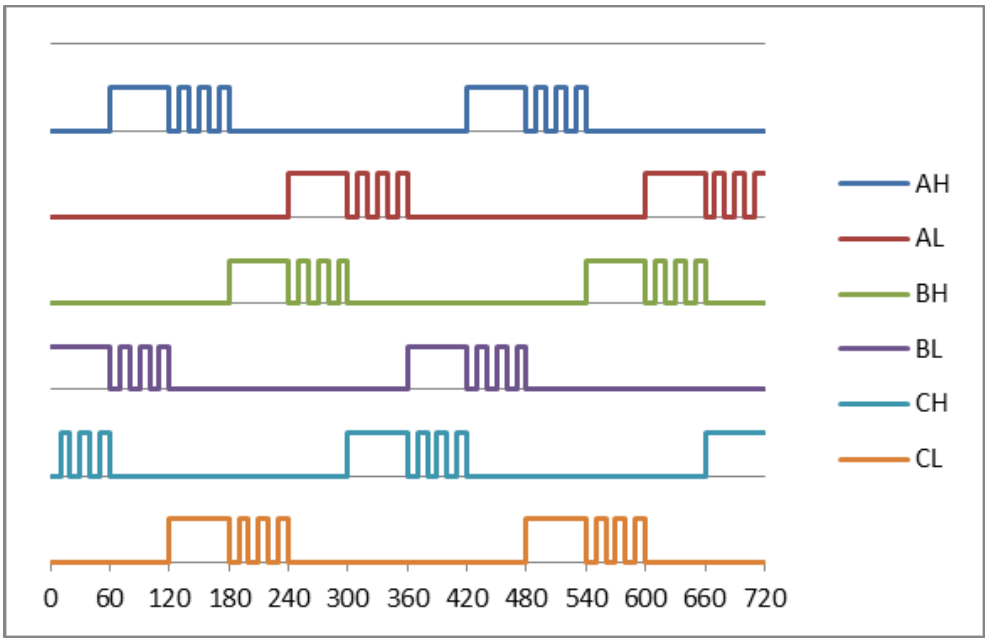


Figure 11. Mixed Mode PWM

Mixed mode PWM is a combination of high-side and low-side PWM. Mixed mode PWM is simple to implement because it requires only 1 active PWM signal at any time. But it can normally be applied only if all the power MOSFETs are N-channel type. The advantage of mixed mode PWM over either high-side or low-side only PWM is that the re-circulation current load is shared between the body diodes of all the 6 MOSFETs. In high-side or low-side PWM schemes, the re-circulation current load is shared between the body diodes of 3 MOSFETs. Hence, mixed mode PWM offers improved long term reliability compared to high-side only or low-side only PWM schemes.

2.3. Determining Commutation Instant

The key problem to solve in sensorless operation of 3-phase BLDC motor is determining the time of commutation. There are various sensorless drive methods based on the zero crossing point (ZCP) detection of the open BEMF. The relationship between the ZCP of the open phase and the commutation instant is shown in Figure 2. It can be observed that the required commutation instant lags 300 behind the ZCP. However, the phase BEMF cannot be extracted directly because the neutral point N is not accessible in most BLDC motors.

Hence, the following circuit using 3 resistors (Rp) is typically used to construct a virtual neutral point:

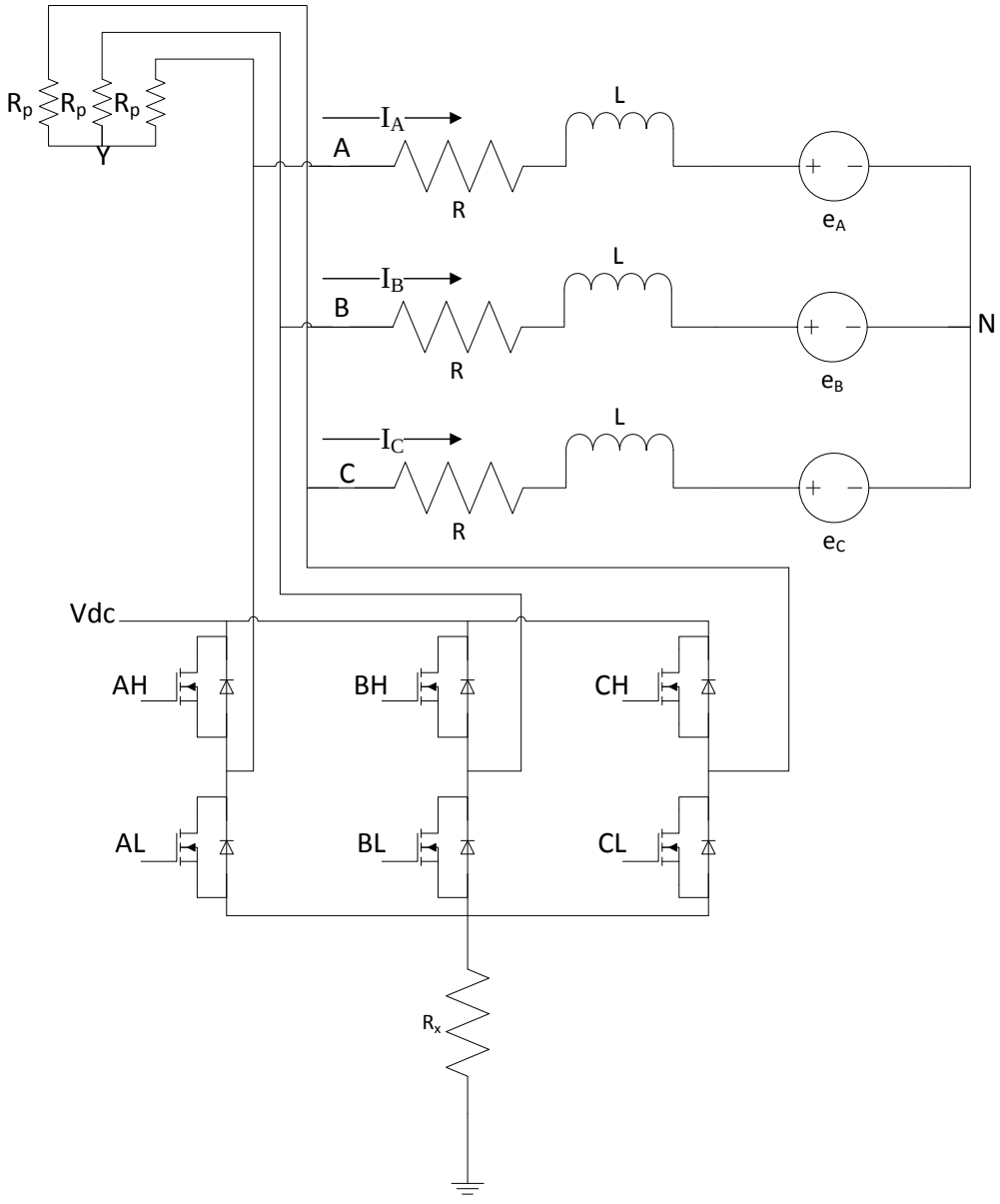


Figure 12. BLDC Motor Drive Circuit with Virtual Neutral Circuit

The voltage at the virtual neutral point Y can be shown to be:

$$V_Y = \frac{1}{3}(V_A + V_B + V_C) \qquad (9)$$

During the commutation phase 1 (0° to 60°), phases C and B are conducting; the phase current and voltage relationships can be easily expressed as follows:

$$I_C = -I_B \quad (10)$$
$$I_A = 0 \quad (11)$$
$$e_C = -e_B \quad (12)$$
$$V_A = e_A + V_N \quad (13)$$

Substituting (10) to (12) into (2) and (3), we can derive the following expression for the neutral voltage:

$$V_N = \frac{1}{2}(V_B + V_C) \quad (14)$$

Substituting (13) and (14) into (9), we derive the virtual neutral voltage:

$$V_Y = \frac{1}{3}e_A + V_N \quad (15)$$

From (14) and (15), it can be observed that a comparator can be used to compare voltages at A and Y to detect the ZCP of the BEMF of A:

$$V_Y - V_A = -\frac{2}{3}e_A \quad (16)$$

It is important to note that the above equations hold irrespective of whether PWM drive is applied to the high side or low side MOSFET. Hence, ZCP occurs when $V_Y - V_A$ crosses zero, and a scheme based on comparing the voltages between Y and A will yield the zero crossing.

2.3.1. High-Side PWM (Part 1 of PWM Off Cycle)

In the earlier analysis, the assumption was made that the open phase A was not conducting during commutation phase 1. But this is not true during the initial phase of the PWM off cycle. During the PWM off phase (assuming high-side PWM), majority of the current flow is as shown below:

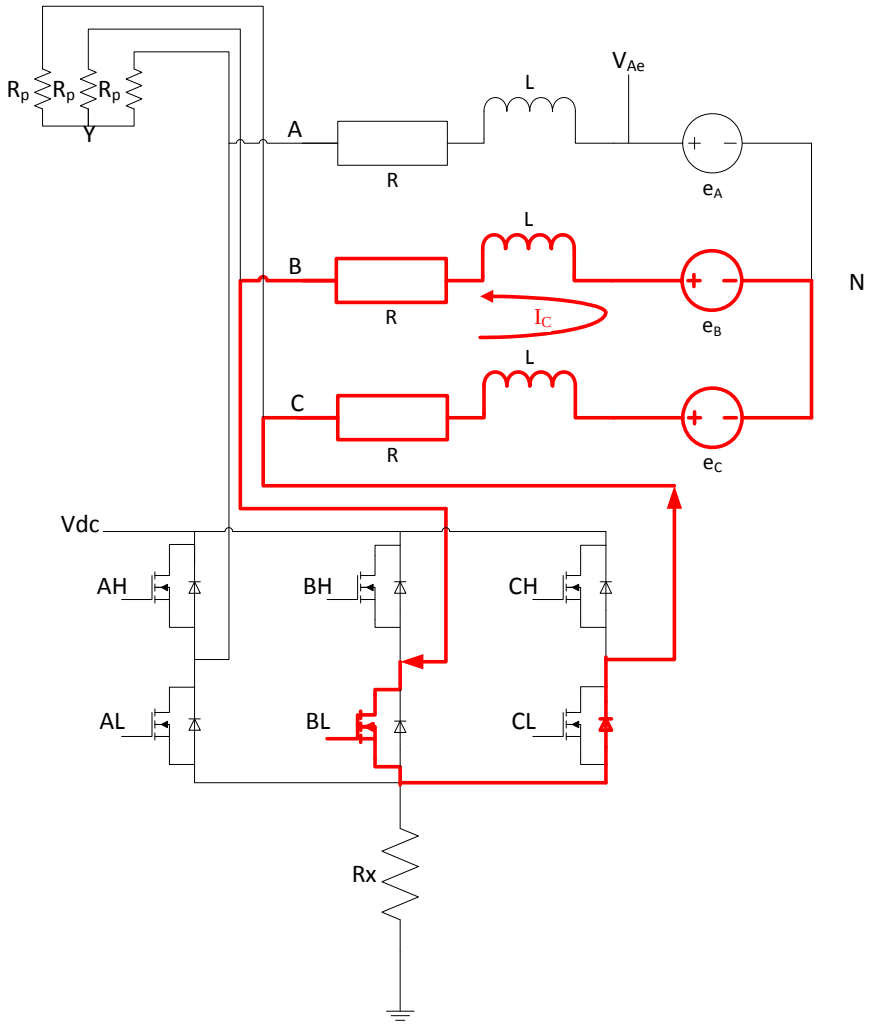


Figure 13. Commutation Phase 1: High-Side PWM Off Majority Current Flow

If we assume the forward bias voltage of the body diode of the CL MOSFET is V_{FCL}, then during the start of the off phase:

$$V_C = -V_{FCL} \quad (17)$$

$$V_B = 0 \quad (18)$$

Substituting (17) and (18) into (14), the neutral point voltage is:

$$V_N = \frac{-V_{FCL}}{2} \quad (19)$$

The voltage after the BEMF element in the system model (refer to Figure 15), V_{Ae}, is:

$$V_{Ae} = V_N + e_A = e_A - \frac{V_{FCL}}{2} \quad (20)$$

If the minimum forward diode voltage required for current to flow is VFmin, then current will flow in the open phase if the following condition is met:

$$V_{Ae} < -V_{Fmin}$$

$$e_A < \frac{V_{FCL}}{2} - V_{Fmin} \quad (21)$$

The current flow is illustrated in Figure 14 and also earlier in Figure 5.

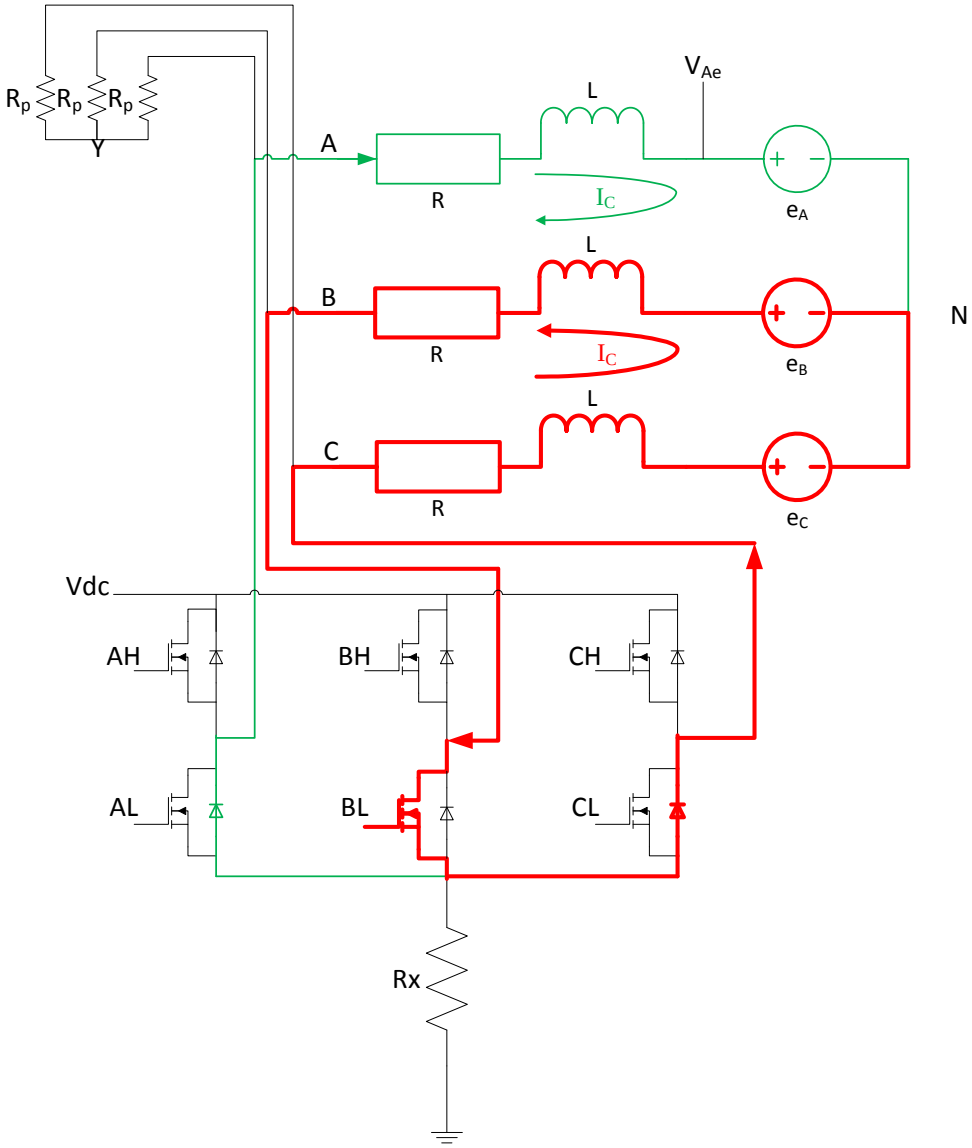


Figure 14. High-Side PWM: PWM Off with Current Flow in Open Terminal

During the commutation Phase 1, when the BEMF of phase A is rising and ZCP of A has not yet occurred, the BEMF of phase A (e_A) is negative (see 0° to 30° in Figure 2); so, it is possible for the condition of inequality (21) to be met. When this occurs, the actual terminal voltage of A when the body diode of AL conducts is:

$$V_A = -V_{FAL} \leq -V_{Fmin} \qquad (22)$$

And the virtual neutral voltage can be derived from (9), (17), (18), and (22):

$$V_Y = -\frac{1}{3}(V_{FAL} + V_{FCL}) \qquad (23)$$

When PWM is off and current flows in the open terminal, the comparison relationship is:

$$V_Y - V_A = -\frac{1}{3}(-2 \times V_{FAL} + V_{FCL}) \qquad (24)$$

This small-value current flowing in the open terminal may or may not decay to zero before the PWM returns to the PWM on cycle. Hence, when current flows in the open terminal, it is impossible to measure the zero-crossing point. And this problem can be further exacerbated by parasitic capacitance across the drain-source terminals of the power MOSFETs that can cause ringing at the motor terminals.

2.3.2. High-Side PWM (Part 2 of PWM Off Cycle)

When the current in the open terminal decays to zero, only the re-circulation current flows through the circuit along the path illustrated in Figure 13. The voltages at terminals A and Y are simply:

$$V_A = V_{Ae} = e_A - \frac{V_{FCL}}{2} \qquad (25)$$

$$V_Y = \frac{1}{3}\left(e_A - 3 \times \frac{V_{FCL}}{2}\right) \qquad (26)$$

The comparison relationship is still the same as when PWM is on:

$$V_Y - V_A = -\frac{2}{3}e_A \qquad (16)$$

2.3.3. Low-Side PWM (Part 1 of PWM Off Cycle)

At the start of the PWM off cycle of the low-side PWM scheme, current flows through the path shown in Figure 15:

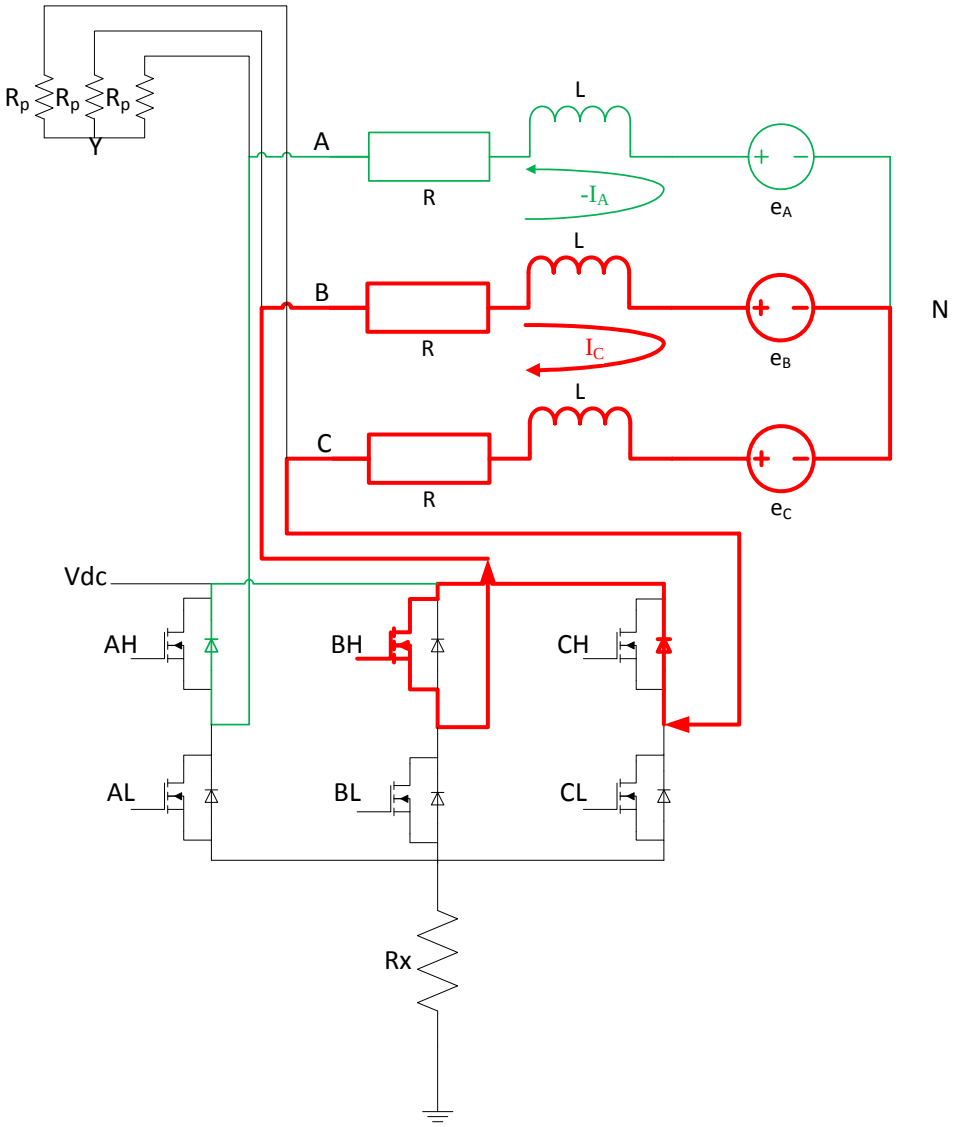


Figure 15. Low-Side PWM: PWM Off with Current Flow in Open Terminal

Similar analysis can be applied to obtain the voltages for low-side PWM during the off phase:

$$V_B = V_{DC} \quad (27)$$

$$V_C = V_{DC} + V_{FCH} \quad (28)$$

$$V_N = V_{DC} + \frac{V_{FCH}}{2} \quad (29)$$

$$V_{Ae} = V_N + e_A = e_A + V_{DC} + \frac{V_{FCH}}{2} \quad (30)$$

And the inequality condition to satisfy to achieve current flow is:

$$V_{Ae} > V_{DC} + V_{Fmin}$$

$$e_A > V_{Fmin} - \frac{V_{FCH}}{2} \quad (31)$$

And the terminal voltages at A and Y are:

$$V_A = V_{DC} + V_{FAH} \quad (32)$$

$$V_Y = V_{DC} + \frac{1}{3}(V_{FAH} + V_{FCH}) \quad (33)$$

The corresponding comparison relationship is:

$$V_Y - V_A = \frac{1}{3}(-2 \times V_{FAH} + V_{FCH}) \quad (34)$$

2.3.4. Low-Side PWM (Part 2 of PWM Off Cycle)

When the open terminal current decays to zero, the voltages at terminals A and Y are simply:

$$V_A = V_{Ae} = e_A + V_{DC} + \frac{V_{FCH}}{2} \quad (35)$$

$$V_Y = V_{DC} + \frac{1}{3}\left(e_A + 3 \times \frac{V_{FCH}}{2}\right) \quad (36)$$

The comparison relationship is still the same as when PWM is on:

$$V_Y - V_A = -\frac{2}{3}e_A \quad (16)$$

2.3.5. Summary of Terminal Voltage Comparisons for ZCP Detection

Table 5 presents a summary of the terminal voltage comparison with the virtual neutral point voltage:

Table 5. Terminal Voltage Comparison Equations

Active PWM State	VY-VA Equation
PWM on	$V_Y - V_A = -\frac{2}{3}e_A$ (16)
High-side/Low-side PWM off, zero current flows in open terminal A	$V_Y - V_A = -\frac{2}{3}e_A$ (16)
High-side PWM off, non-zero current flows in open terminal A	$V_Y - V_A = -\frac{1}{3}(-2 \times V_{FAL} + V_{FCL})$ (24) conditional upon: $e_A < \frac{V_{FCL}}{2} - V_{Fmin}$ (21)
Low-side PWM off, non-zero current flows in open terminal A	$V_Y - V_A = \frac{1}{3}(-2 \times V_{FAH} + V_{FCH})$ (34) conditional upon: $e_A > V_{Fmin} - \frac{V_{FCH}}{2}$ (31)

These equations will aid in the selection of the hardware design, PWM scheme, and firmware algorithm to be used in the design kit.

3. System Implementation

This section describes the system implementation of the design kit. Schematics for the design kit can be found in "8. Schematics" on page 57. The kit consists of two boards: an MCU board and a powertrain board. The block diagram of the reference design is shown in Figure 16.

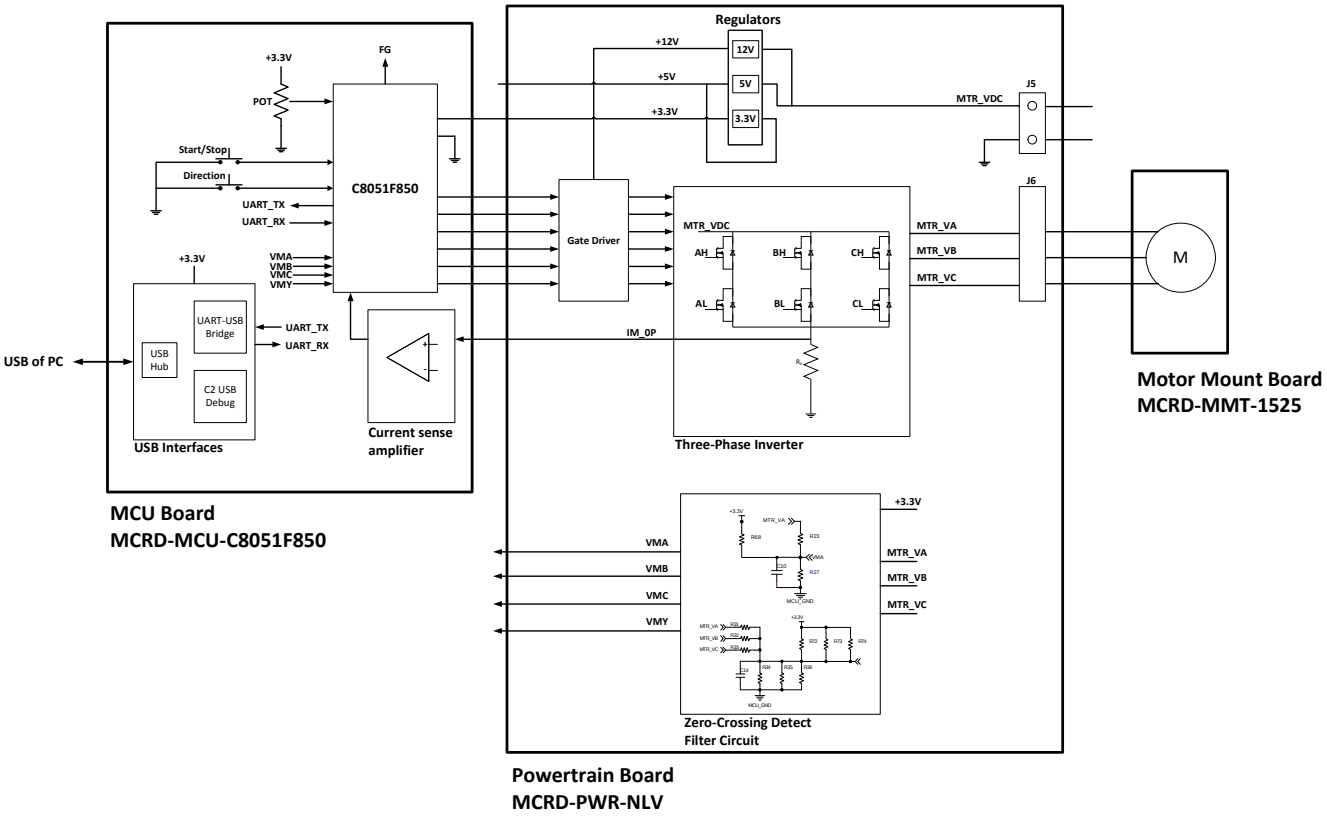


Figure 16. Reference Design Block Diagram

3.1. Powertrain Board

The powertrain board is designed to meet the following key motor specifications:

- Motor supply voltage range of 10 to 24 V
- Maximum average current of 10 amps

It consists of the following components that are relevant to this application:

- 6 IRFH7446 Power MOSFETS for the inverter circuit
- 3 Silicon Labs Si8230 isolated dual drivers
- An LDO to generate the 3.3 V required by the MCU board
- 50 mΩ current sensing resistor rated for 10 W
- Motor terminal blocks to allow user to attach their own motor
- Resistor divider to generate attenuated motor voltage supply (VMDC) - allows MCU to determine if motor supply voltage is high enough for safe operation
- Resistor dividers to generate attenuated motor phase voltages with a small positive offset voltage (VMA, VMB, VMC)
- Resistor network to generate attenuated sum of motor phase voltages with a small positive offset voltage (VMY)

3.1.1. Attenuated Motor Voltage Circuit

There is a circuit to provide an attenuated motor voltage signal to the MCU as shown below:

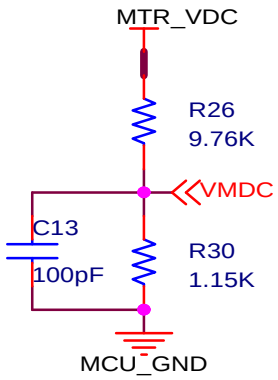


Figure 17. Attenuated Motor Voltage Circuit

This circuit allows the MCU to measure the motor voltage and determine whether the voltage is high enough to operate the motor.

3.1.2. Back-EMF Filter Circuit

The BEMF filter circuit is intended to attenuate the motor terminal voltage to a level that can be used by the MCU. The circuit for one of the terminals is shown below:

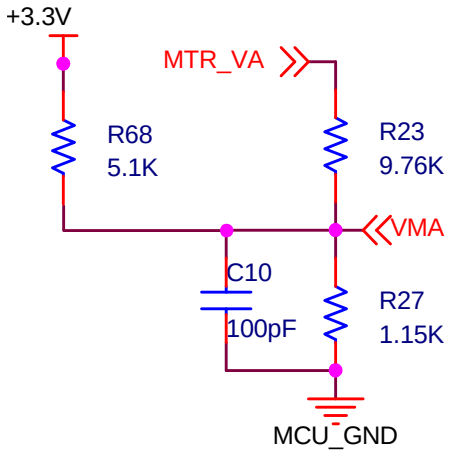


Figure 18. Back-EMF Filter Circuit for Motor Terminal Voltage

A slight positive offset voltage via the pull-up R68 because the voltage at MTR_VA may be negative during the PWM off cycle (refer to equations 22 and 25) of high-side PWM operation. It can be shown that the voltage at VMA is:

$$V_{VMA} = \frac{\frac{3.3}{R68}}{\frac{1}{R23}+\frac{1}{R27}+\frac{1}{R68}} + \frac{\frac{V_{MTR\_VA}}{R23}}{\frac{1}{R23}+\frac{1}{R27}+\frac{1}{R68}} \quad (37)$$

This sets the offset voltage at 0.55 V and the gain at 0.0877. C10 is a 100 pF capacitor that filters sharp ringing voltages at the motor terminal.

3.1.3. Virtual Neutral Filter Circuit

The virtual neutral voltage is also adjusted similarly with a resistor network:

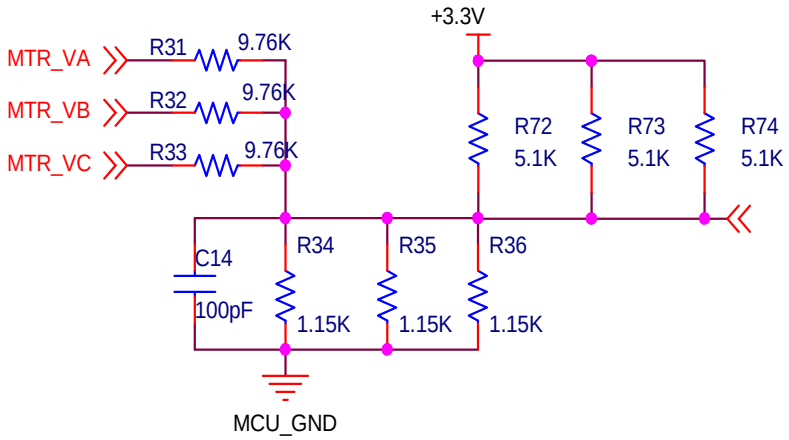


Figure 19. Virtual Neutral Filter Circuit

It can be similarly demonstrated that the voltage at VMY is:

$$V_{VMY} = \frac{\frac{+3.3}{R72}}{\frac{1}{R31}+\frac{1}{R34}+\frac{1}{R72}} + \frac{\frac{(V_{MTR\_VA}+V_{MTR\_VB}+V_{MTR\_VC})}{(3\times R31)}}{\frac{1}{R31}+\frac{1}{R34}+\frac{1}{R72}} \qquad (38)$$

Recall equation 9:

$$V_Y = \frac{1}{3}(V_A + V_B + V_C) \qquad (9)$$

This shows that the voltage at VMY is the virtual neutral voltage with the same scale and offset applied in the same manner as VMA. Thus, the VMx signals from the resistor network circuits can be used for back-EMF zero crossing detection by the MCU's comparator without saturating. C14 is a 100 pF capacitor that filters sharp ringing voltages at the motor terminals.

3.2. MCU Board

The MCU board consists of the following:

- C8051F850-A-GU QSOP-24 package
- 2 push-buttons
- 3 controllable LEDs
- 1 rotary variable resistance potentiometer
- Op-amp to amplify and bias the current sense voltage
- USB Hub to support:
  - C2 USB debug interface
  - CP2103 USB-UART bridge operating at 115200 baud
  - CP2112 USB-I2C bridge
- Configurable jumpers to select either Hall-sensored or sensorless mode of operation
- Test points for connecting to gate drive of user powertrain board and motor.

The crossbar is configured such that all digital pins have pull-ups disabled. This is an unusual configuration of the digital pins, which will be explained in "3.3. Back-EMF Zero Crossing Point Detection Technique" on page 25.

3.2.1. Gate Drive Pin Connections

Pins P1.2 to P1.7 are connected to the gate drive pins of the powertrain board. These pins are configured as push-pull outputs by the firmware. The pin connections, pin names and associated motor phase are specified in Table 6.

Table 6. MCU Motor Gate Drive Pin Connection

MCU Pin	Connection Name	Controlled Motor Phase
P1.2	PWM0B_GD0_EN	A
P1.3	PWM0A	A
P1.4	PWM1B_GD1_EN	B
P1.5	PWM1A	B
P1.6	PWM2B_GD2_EN	C
P1.7	PWM2A	C

This sequence facilitates the commutation using the crossbar pin skip register P1SKIP. For example in the commutation sequence shown in Figure 20:

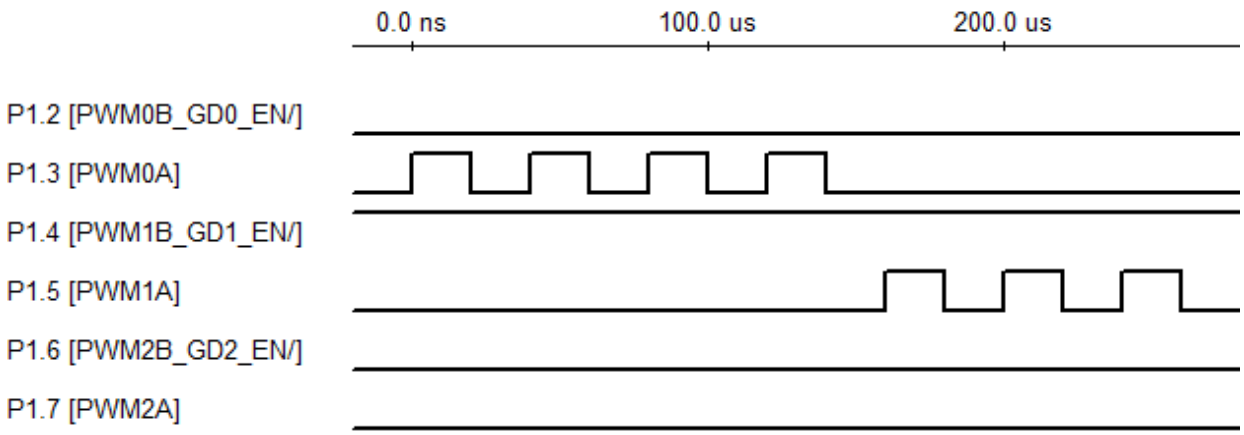


Figure 20. Commutation from PWM0A to PWM1A

Prior to the commutation event, P1SKIP contains the value 0xF7. The commutation event can be easily executed by a single C language instruction:

```
P1SKIP = 0xDF; // P1.5 can be assigned by crossbar
```

Depending on the firmware configuration, users can observe PWM signals on either the high-side or low-side gate drive pins.

3.2.2. Motor Sensor Connections

Pins P0.0, P0.1 and P0.2 are connected to jumpers J110, J111, J112 respectively. The jumpers allow the user to select either Hall-sensored mode operation or sensorless mode operation. For sensorless mode operation, the MCU expects these pins to be connected to attenuated motor phase voltages that are offset with a small positive voltage (see "3.1.2. Back-EMF Filter Circuit" on page 21). In sensorless mode operation, P0.3 must be connected to the attenuated virtual neutral that is offset with a small positive voltage. Refer to the attenuation and offset circuit in "3.1.3. Virtual Neutral Filter Circuit" on page 22. As explained earlier sections, the offset voltage is required because the voltage at the motor terminal may be negative during the PWM off cycle when PWM is applied to a high-side FET.

3.2.3. Current Sensing Circuit

Pin P0.6 is connected to the current sensing voltage of the motor drive circuit. Jumper 113 allows the user to select 1 of 2 options to measure this voltage:

- Direct connection to low-side current sensing resistor. This is a low-cost option when the user is only concerned with over-current protection and hence is only required to detect high current on this pin.
- Output of a op-amp circuit (refer to U201 in schematics). This is the default jumper selection. This option is for applications that need to measure small current for motor start-up.

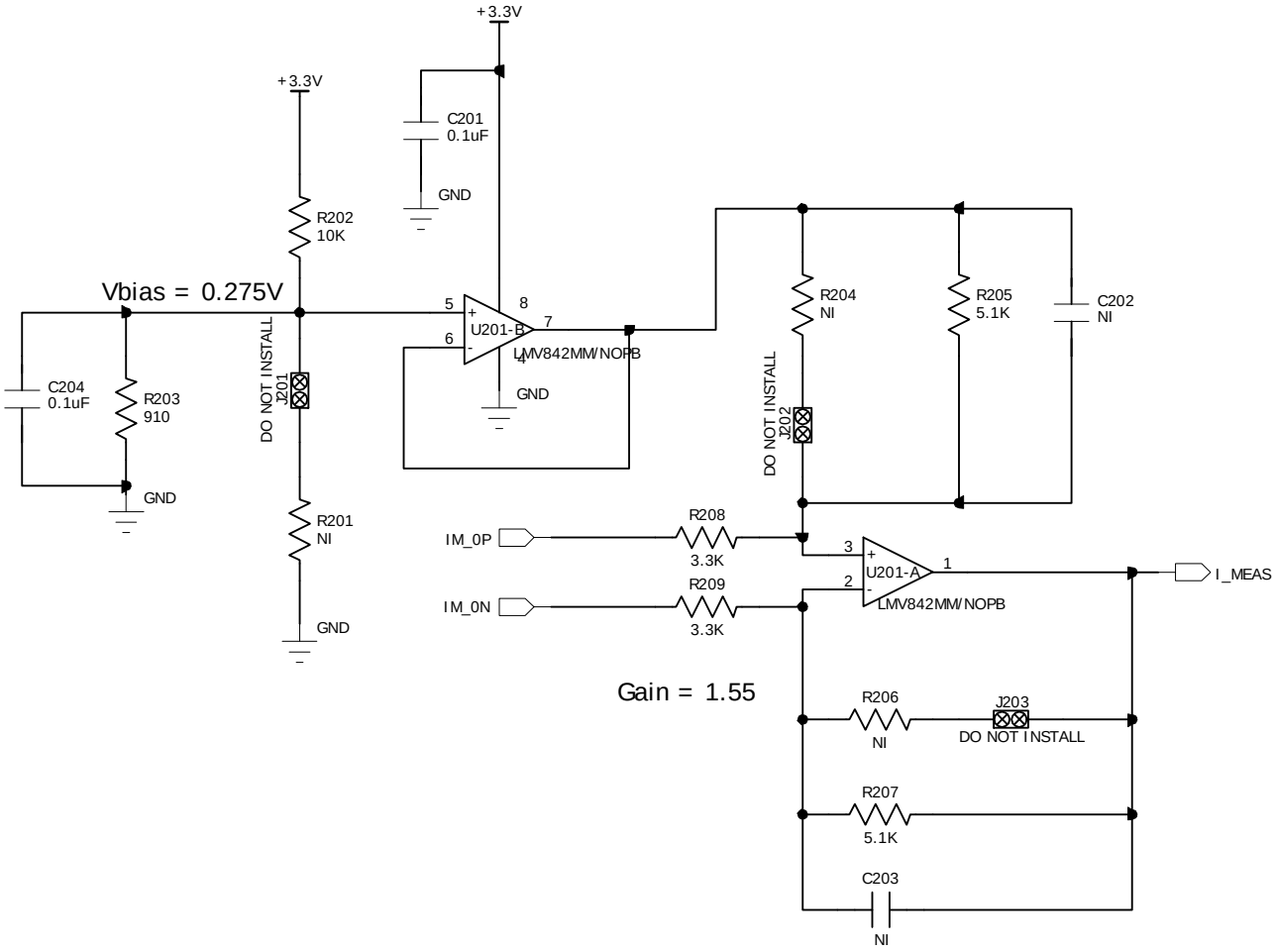


Figure 21. Current Sensing Opamp Gain Circuit

The current sense circuit adds a small 0.275V offset and gain of 1.545 for the current sense voltage.

3.3. Back-EMF Zero Crossing Point Detection Technique

Detecting the Back-EMF zero crossing point can be challenging when there is an active PWM signal that interferes with the BEMF signal. Some designs implement a low-pass filter for the terminal signals and the virtual neutral. However, a low-pass filter is not suitable for motors with high commutation frequencies because of the phase shift caused by the filter.

This design kit implements a technique that takes advantage of some unique features of the C8051F850. Referring to the terminal comparison equations in Table 5 on page 19, it can be observed that the open terminal does not yield any zero crossing information when current flows through that terminal. So, a tracking signal is used to disable a comparator input so that the comparator is effectively not operational when current is flowing in the open terminal.

When BEMF is rising in the open terminal, the firmware configures the peripherals to operate as shown in Figure 22.

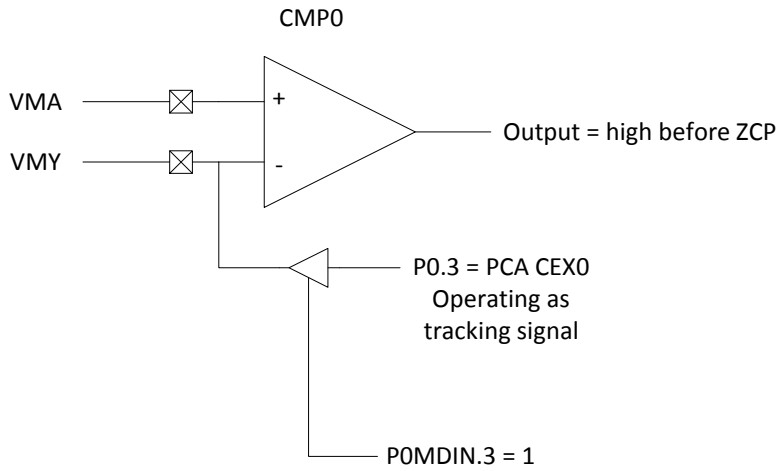


Figure 22. Peripheral Configuration for ZCP Detection on Rising BEMF Signal

When the BEMF is falling in the open terminal, the firmware re-configures the peripherals to operate as shown in Figure 23.

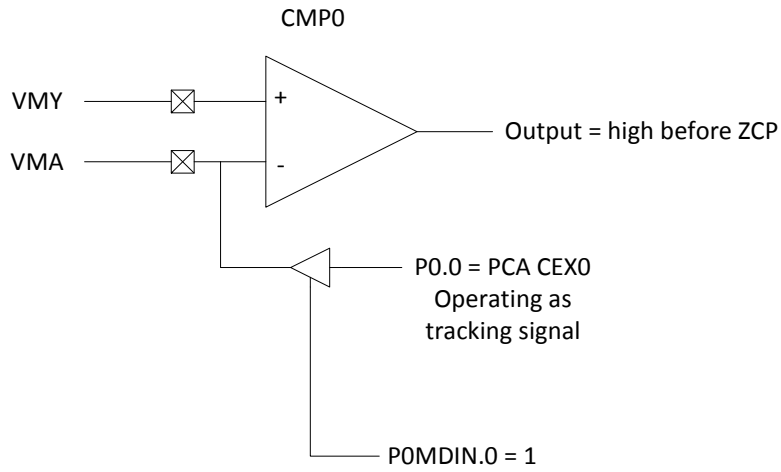


Figure 23. Peripheral Configuration for ZCP Detection of Falling BEMF Signal

Recall from Table 5 the terminal voltage comparison equations; they show that ZCP cannot be detected reliably when current is flowing in the open terminal because the voltages are dominated by the forward bias voltage of the body diode of the power MOSFETs. Thus, a tracking signal is used to enable the comparator for use at appropriate times during the PWM cycle.

The tracking signal technique requires the following conditions to be met:

1. Ensuring the filtered terminal and virtual neutral voltages are positive - this is accomplished by adding a small positive offset voltage in the filter circuits as shown in Figure 18 on page 21 and Figure 19 on page 22.
2. MCU disables the pull-ups via the crossbar.
3. The pin that is connected to negative input of comparator is configured as digital input (P0MDIN.x = 1).
4. CEX0 is setup as a PWM tracking signal connected to the negative input of the comparator as shown in Figures 22 and 23.

When the motor PWM duty cycle is low (inactive period is much longer than the active period), CEX0 is setup to synchronize with the motor PWM signal to observe the BEMF only at the tail end of the inactive part of the PWM cycle as shown in Figure 24:

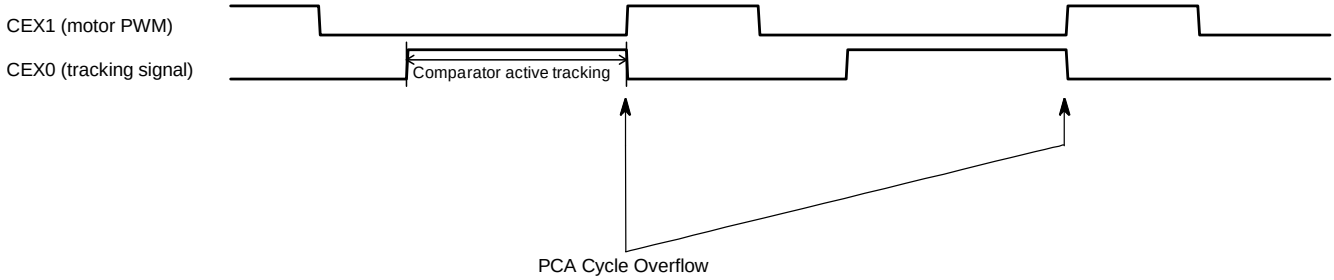


Figure 24. Active High Motor PWM (Tracking Synchronization for Low Duty Cycle)

This corresponds to the last part of the motor PWM off cycle when no current flows in the open terminal, as shown in Figure 7 on page 10.

When the motor PWM duty cycle is high, CEX0 is setup to observe the BEMF at the tail end of the inactive part of the PWM cycle and the entire active part of the PWM cycle as shown in Figure 25.

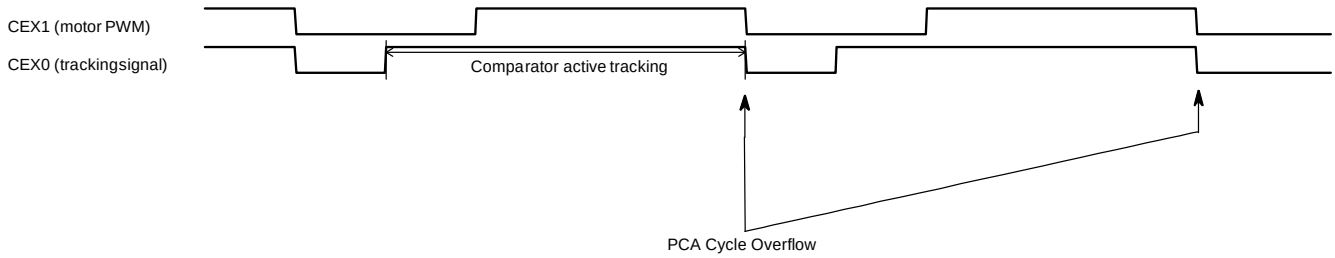


Figure 25. Active High Motor PWM (Tracking Synchronization for High Duty Cycle)

The advantage of this technique is that blanking out the undesirable parts of the BEMF signal does not incur any CPU overhead. If the tracking signal is not used, the MCU must be interrupted at least twice every PWM cycle to enable and disable the comparator to observe the BEMF. When this processing overhead is removed, a higher frequency motor PWM signal can be applied. This leads to lower current ripple and reduced torque ripple during the operation of the BLDC motor.

3.4. PWM and Comparator Configuration Sequences

Based on the different PWM techniques and Back-EMF ZCP detection techniques, we can derive the sequences of peripheral assignments for different PWM modes. Tables 7, 8, and 9 specify the peripheral assignment sequences for the three different PWM modes in a specific direction.

Table 7. High-Side PWM Peripheral Configuration Sequence

State	Gate Drive Outputs						BEMF			
	High-Side			Low-Side			Open Phase	Slope	CP0+	CP0- and CEX0
	AH	BH	CH	AL	BL	CL				
0	PWM	L	L	L	H	L	C	\	VMC	VMY
1	L	L	PWM	L	H	L	A	/	VMY	VMA
2	L	L	PWM	H	L	L	B	\	VMB	VMY
3	L	PWM	L	H	L	L	C	/	VMY	VMC
4	L	PWM	L	L	L	H	A	\	VMA	VMY
5	PWM	L	L	L	L	H	B	/	VMY	VMB

Table 8. Low-Side PWM Peripheral Configuration Sequence

State	Gate Drive Outputs						BEMF			
	High-Side			Low-Side			Open Phase	Slope	CP0+	CP0- and CEX0
	AH	BH	CH	AL	BL	CL				
0	H	L	L	L	PWM	L	C	\	VMC	VMY
1	L	L	H	L	PWM	L	A	/	VMY	VMA
2	L	L	H	PWM	L	L	B	\	VMB	VMY
3	L	H	L	PWM	L	L	C	/	VMY	VMC
4	L	H	L	L	L	PWM	A	\	VMA	VMY
5	H	L	L	L	L	PWM	B	/	VMY	VMB

Table 9. Mixed Mode PWM Peripheral Configuration Sequence

State	Gate Drive Outputs						BEMF			
	High-side			Low-side			Open phase	Slope	CP0+	CP0- and CEX0
	AH	BH	CH	AL	BL	CL				
0	H	L	L	L	PWM	L	C	\	VMC	VMY
1	L	L	PWM	L	H	L	A	/	VMY	VMA
2	L	L	H	PWM	L	L	B	\	VMB	VMY
3	L	PWM	L	H	L	L	C	/	VMY	VMC
4	L	H	L	L	L	PWM	A	\	VMA	VMY
5	PWM	L	L	L	L	H	B	/	VMY	VMB

For rotation in the opposing direction, the peripheral configuration sequences can be adjusted by exchanging the BEMF rows with the same “Open Phase” terminal.

3.5. BLDC Motor Startup Technique

In the typical BLDC motor sensorless starting phase, the motor is driven like a stepper motor. The motor is initially commutated very slowly then velocity is increased while the PWM duty cycle is increased to boost the applied motor voltage in an attempt to keep the current constant.

However, it is not easy to predetermine PWM duty cycle for constant current level because the motor load may change or the motor supply voltage fluctuates. The design kit uses the comparator clear feature to trim the motor PWM duty cycle automatically to ensure that the current does not exceed a predetermined level regardless of the motor load or motor supply voltage. During motor startup, the comparator and PWM are configured as shown in Figure 26.

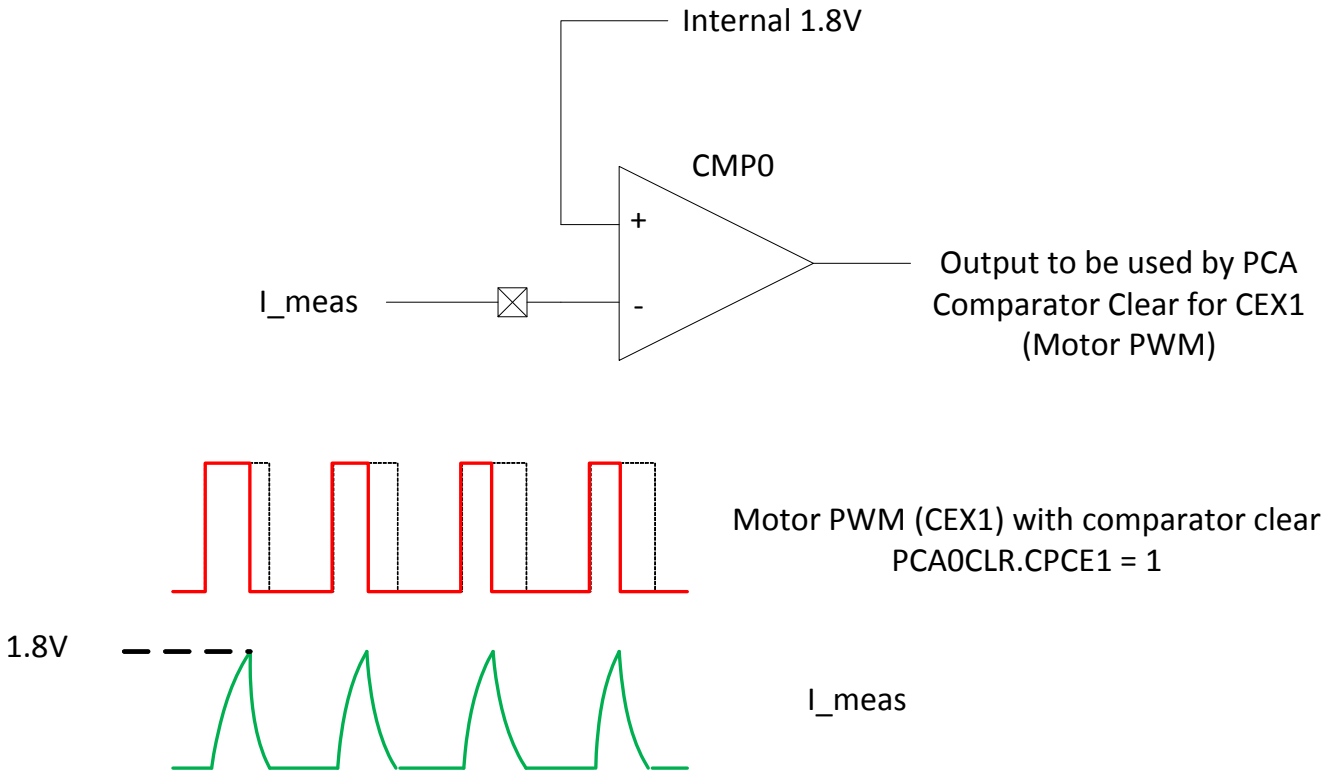


Figure 26. Comparator and PCA Configuration for Motor Startup

Using the comparator clear mechanism, the PWM signal is automatically shut off for that cycle when the current sensing voltage exceeds 1.8 V. The firmware programs the MCU to generate a 50% duty cycle PWM signal for motor startup and lets the comparator clear functionality trim the duty cycle to limit the peak current. The current trip level can be adjusted by changing the resistors R208 and R209 shown in Figure 21 on page 24. The gain of the op-amp = R207/R209. R208 and R209 should always have the same value; R205 and R207 should always have the same value. In the reference design kit, the limit is set such that the current sensing voltage will trigger the comparator at twice the maximum current supported by the motor. The trigger limit is set at 20 A in the design because the average current will not exceed 10 A if the motor PWM is running at the maximum 50% duty cycle. Using this method, the firmware does not need to store a table of duty cycles to use during startup.

During startup, the firmware commutates at a rate such that the angular speed is increased at a constant rate. As commutation always drives the rotor a fixed angle, the angular speed is directly proportional to the reciprocal of the time interval since the last commutation. At the same time, we also want the speed to be increased at a linear rate:

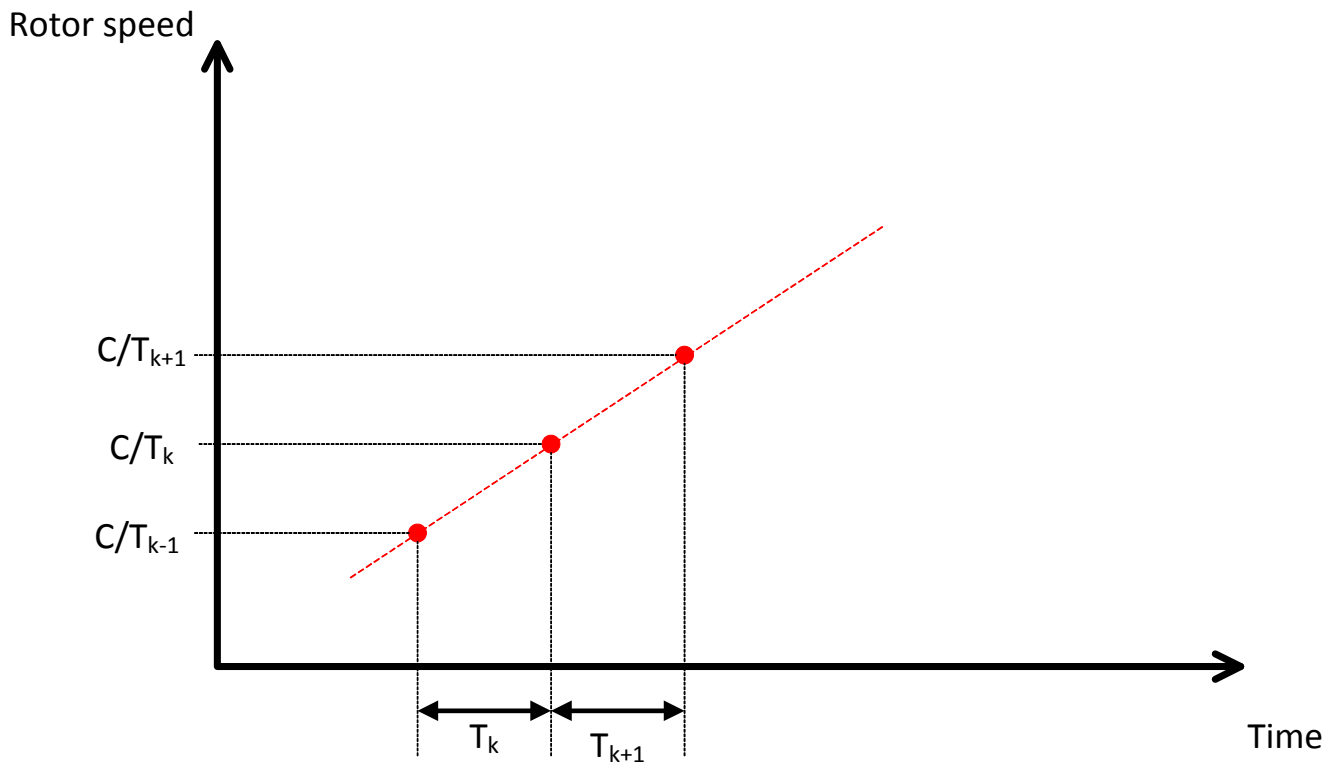


Figure 27. Motor Startup (Linearly Increasing Angular Speed)

Each T_i represents the time interval between successive commutation instants. It can be demonstrated that the time interval can be represented by the following recurrence relationship:

$$T_{k+1} = \left(\frac{1}{2}\sum_{i=0}^{k} T_i\right) \times \left[\sqrt{1 + \frac{4T_k}{\sum_{i=0}^{k} T_i}} - 1\right] \qquad (39)$$

The first 3 terms can be worked out off line and stored within the firmware using only 6 bytes of code memory. Using binomial expansion, the subsequent terms can be computed in firmware using the following approximation:

$$T_{k+1} \approx \frac{T_k \sum_{i=0}^{k-1} T_i}{\sum_{i=0}^{k} T_i} \qquad (40)$$

This method significantly reduces the amount of code space to store the startup interval table. The firmware will operate in this startup mode until the motor speed reaches 5% of the maximum motor speed.

3.6. Hyperdrive Mode

In block commutation driving method, maximum speed is achieved when the motor PWM duty cycle is at 100%. Hyperdrive mode is a technique to further increase this maximum speed. Recall the electrical torque equation shown earlier:

$$T_e = KI_A F(\theta_e) + KI_B F\left(\theta_e - \frac{2\pi}{3}\right) + KI_C F\left(\theta_e - \frac{4\pi}{3}\right) \qquad (8)$$

In the typical block commutation, there is zero current through one motor terminal at any one time because the phase is open for ZCP detection. If the third terminal can be energized, there will be increased electrical torque generated to further increase the speed of the motor, but the third terminal is required for ZCP detection. However, the open phase is free to be energized after ZCP has been detected. This technique is the most beneficial for motor designs in which the motor current saturates well before the next commutation event.

4. Firmware

The firmware is located in *<Install Directory>\F850_BLDC_RD\Firmware*, where *<Install Directory>* is *C:\Silabs\MCU* if the default option is chosen during installation. The firmware consists of two IDE projects. One project is for sensorless designs and is called f85x_bldc.wsp; the other project is for Hall-sensor designs and is called f85x_bldc_hall.wsp.

The firmware can be re-compiled with the following steps:

1. Launch the Silicon Laboratories IDE from the Start Menu.
2. Select the “Project | Open Project…” menu item.
3. Browse to the source code location listed above, and open the file f85x_bldc.wsp (or f85x_bldc_hall.wsp for Hall-sensored design).
4. Select the “Project | Rebuild Project” menu item, or click on the “Rebuild All” button. The IDE will compile and link all the files in the project.

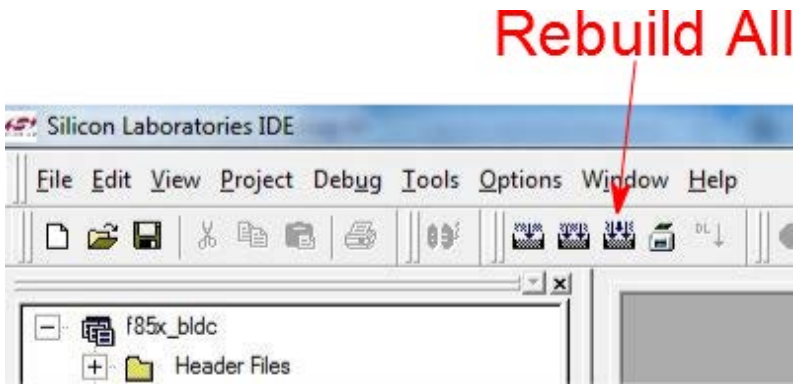


Figure 28. Rebuild BLDC Source Code

5. Select the “Options | Connection Options…” menu item to bring up the Connection Options dialog.

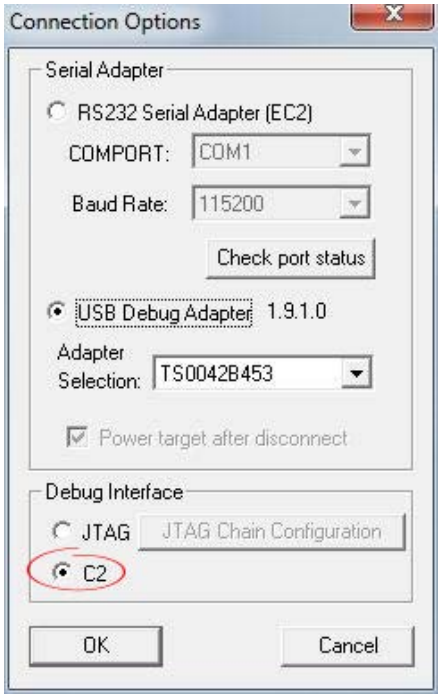


Figure 29. Connection Option Dialog

6. Select the USB Debug Adapter of the BLDC kit. Make sure that “C2” is selected as the Debug Interface. Then, click OK.
7. Select the “Debug | Connect” menu item, or click on the “Connect” toolbar button, or press the Alt+C keys. This will connect the IDE to the MCU on the BLDC kit.

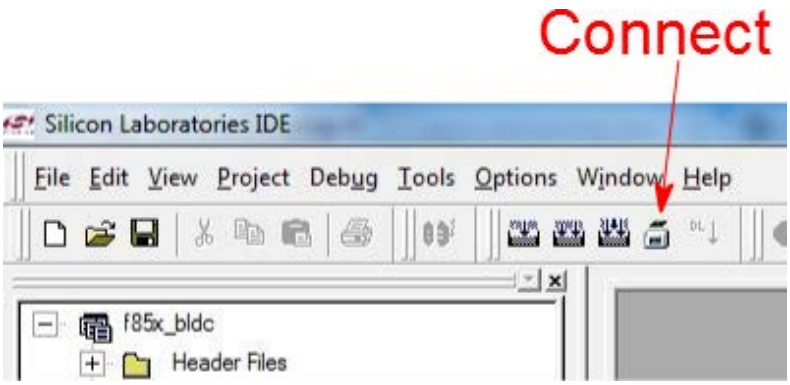


Figure 30. Connect to MCU on BLDC Kit

8. Select the “Download code” toolbar button.

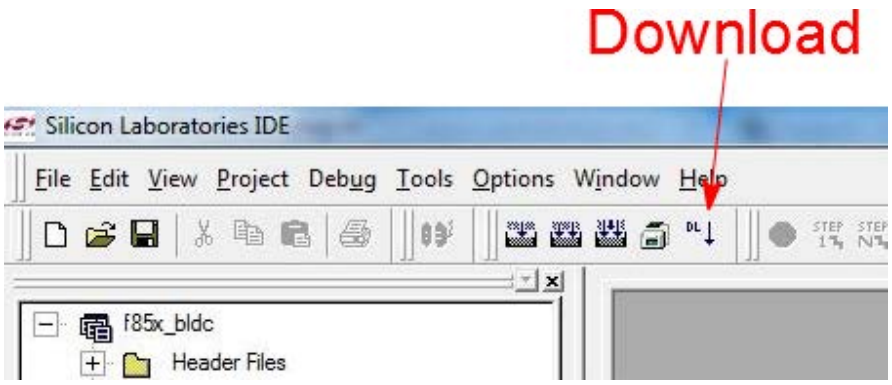


Figure 31. Download Code into MCU

9. When completed, click on the “Disconnect” toolbar button to disconnect from the MCU.

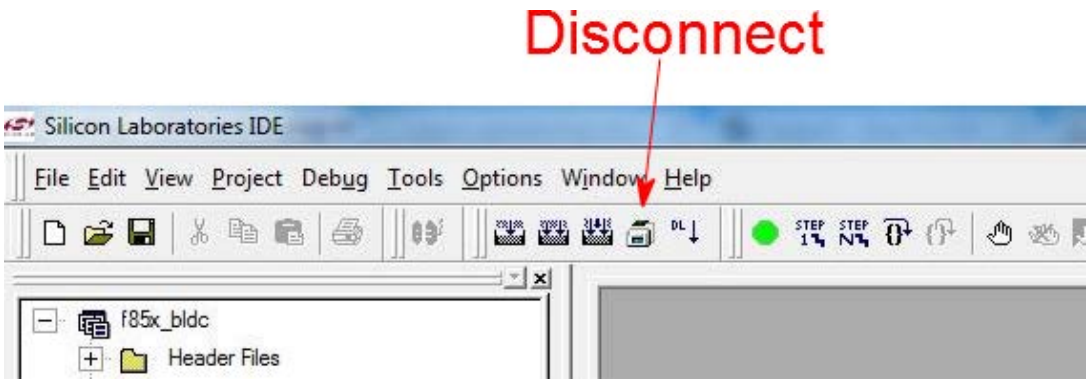


Figure 32. Disconnect IDE from the MCU

The motor control application firmware framework is designed and organized such that user application and motor control services are separated. Motor control services are exposed through a set of variables and functions. Figure 33 shows a high-level overview of this organization.

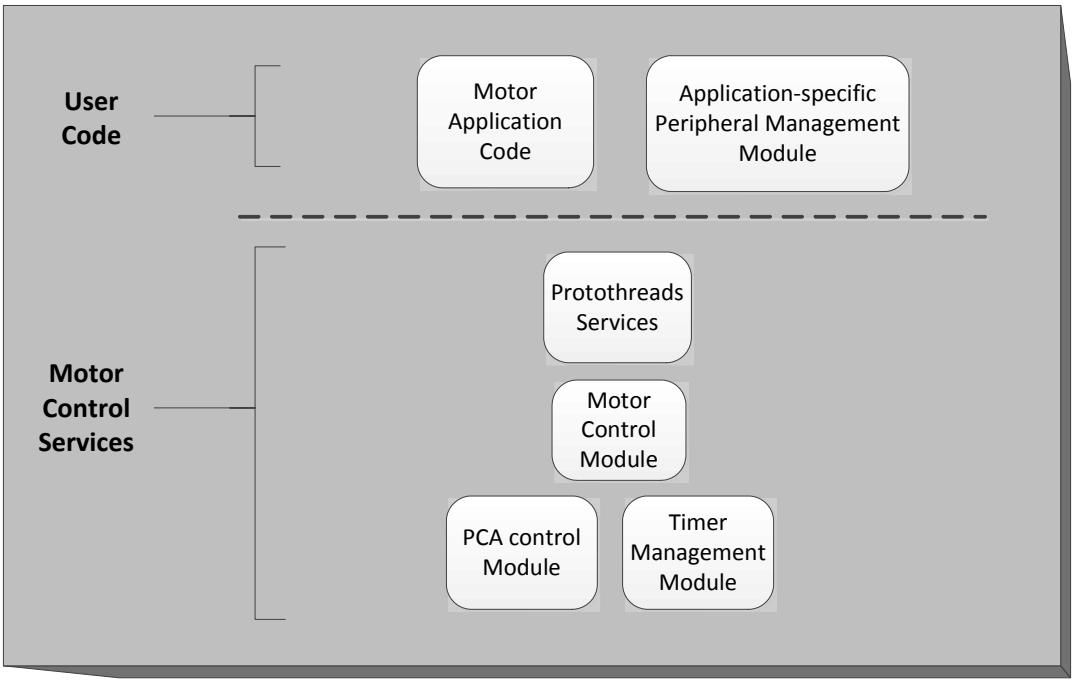


Figure 33. Motor Control Application Firmware Framework

This framework simplifies the organization and implementation of user-specific application code. Users can write application code to access the following services:

- Standard C library
- Standard C8051F85x peripheral registers such as CLKSEL, P0, P1.
- Protothreads Services
  - Protothreads are extremely lightweight, stackless threads developed by Adam Dunkels. More information can be found at this website: http://dunkels.com/adam/pt
  - The Protothreads API are prefixed by PT_.
- Motor control services. These are services provided to the application for the purpose driving the BLDC motor. There are 3 groups of API services:
  - Functions. These are prefixed by SL_.
  - Read-only variables. These are prefixed by SLR_.
  - Read-write variables. These are prefixed by SLW_.

The firmware provided in the design kit is organized as follows:

Table 10. Categorization of BLDC Reference Design Firmware

Category	Description	Files
User Application	This is the user application code. Users should re-implement the code here for their own applications. The default firmware is complex because it communicates with the Silicon Labs Spinner over the UART interface and presents a register-based configuration interface.	btn.c main.c MCP_core.c MCP_Registers.c mtrapp.c UART_Driver.c bldcdk.h btn.h MCP_Core.h MCP_Registers.h mtrapp.h UART_Driver.h
Standard C8051F85x Services	Standard header files to access C8051F85x SFRs in a compiler-independent manner.	C8051F850_defs.h compiler_defs.h
Protothreads Services	An extremely lightweight, stackless, cooperative threading service.	lc.h lc-addrlabels.h lc-switch.h pt.h pt-sem.h
Motor Control Services	These files implement the motor control services. Functions and variables that are accessible by user application are prefixed by SL_, SLR_, SLW_. BLDC_RD_Build_Params.h and BLDC_RD_System.h can be modified to test out other operating modes – such has high-side PWM or low-side PWM modes. These 2 files are configuration files and they provide the configuration interface between the motor control module and the user application. The ADC module can be easily modified to read other user-defined voltages.	adc.c BLDC_RD_Build_Params.h BLDC_RD_System.h comp.c motor.c mtrpid.c adc.h pca.c timers.c comp.h motor.h mtrpid.h pca.h timers.h

4.1. Motor Control API

The motor control API functions consists of the following functions.

Table 11. Motor Control API Functions

Function Name	Description
SL_MTR_init(void)	Initializes the motor control variables and peripherals (PCA, timers, comparator). User application should call this function during MCU initialization. It initializes the motor state machine to the MOTOR_STOPPED state.
SL_MTR_motor(void)	Executes the motor control state machine according to the motor state. User application should call this function regularly within its state machine or background loop.
SL_MTR_start_motor(void)	This function starts the BLDC motor spinning if the motor is in the MOTOR_STOPPPED state. This function will first validate if the motor is still spinning. If the motor was still spinning in the desired direction, this function will start spinning the motor immediately. Otherwise, this function will align the motor in a known position before it starts spinning the motor. After starting the motor, the motor state machine is transitioned to the MOTOR_RUNNING state.
SL_MTR_stop_motor(void)	This function stops energizing the coils of the BLDC motor if the motor is in the MOTOR_RUNNING state. The motor state machine will then be transitioned to the MOTOR_STOPPPED state.
SL_MTR_time(void)	Gets the high 16-bit of the 32-bit running time of the system. Each time unit is 2.7 ms.
SL_MTR_GET_32BIT_TIME(x)	A macro to read the 32-bit running time of the system and store in x. x is a UU32 type variable.
SL_MTR_change_pid_gain(pg, ig	Initializes the PI proportional gain and PI integral gain parameters. The proportional gain is represented in units of 256/(SPEED_UNIT RPM). Refer to "4.5.9. Rotation Speed Resolution" on page 49 for definition of SPEED_UNIT. The integral gain has the same units based on a time resolution unit of 1.
SL_MTR_change_num_poles(poles)	Change the number of poles in the motor. This is normally not used in an actual application, but this is useful in a demo kit where users may wish to test a new motor without recompiling the firmware.

The motor control API also exports six read-only variables for the user application. These variables are prefixed by SLR_.

Table 12. Motor Control API Read-Only Variables

Variable Name	Description
SLR_motor_state	Represents the state of the motor state machine. Can be one of 2 states: MOTOR_STOPPPED or MOTOR_RUNNING.
SLR_motor_current_rpm	Represents the current motor rotation speed in units of (SPEED_UNIT x RPM). SPEED_UNIT is a user-configurable compile-time parameter to constrain the maximum rotation speed to a 16-bit value.
SLR_pwm_duty	Represents the duty cycle of the motor PWM signal. This value is a linear 16-bit quantity where 0xffff represents 100% duty cycle.
SLR_motor_stalled	This is a 1-bit variable to indicate that the motor has stalled due to a system error event.
SLR_motor_current	Average motor current measured by the sense resistor. This is expressed in units of 0.01 A.
SLR_motor_voltage	Average motor supply voltage in units of 0.1 V. This is available only if FEATURE_MEAS_VMDC is enabled.

The motor control API exports writable variables for the user application to control the operation of the motor control state machine. These variables are prefixed by SLW_.

Table 13. Motor Control API Writable Variables

Variable Name	Description
SLW_target_rpm	This unsigned 16-bit variable is used only when the BLDC_RD_RPM_OR_PWM macro is set to RPM_PARAMETER. This variable is used by the user application to indicate the target rotation speed that the motor control state machine should achieve. It is specified in units of (SPEED_UNIT * RPM).
SLW_target_pwm_duty	This unsigned 16-bit variable is used only when the BLDC_RD_RPM_OR_PWM macro is set to PWM_PARAMETER. This variable is used by the user application to indicate the target motor PWM duty cycle that the motor control state machine should achieve. This value is a linear quantity where 0xffff represents 100% duty cycle
SLW_acceleration_step_size	This unsigned 16-bit variable is used only when the BLDC_RD_RPM_OR_PWM macro is set to PWM_PARAMETER. This variable is used by the user application to indicate the rate at which the motor PWM duty cycle is incremented towards the target PWM duty cycle specified in the SLW_target_pwm_duty variable.
SLW_deceleration_step_size	This unsigned 16-bit variable is used only when the BLDC_RD_RPM_OR_PWM macro is set to PWM_PARAMETER. This variable is used by the user application to indicate the rate at which the motor PWM duty cycle is decremented towards the target PWM duty cycle specified in the SLW_target_pwm_duty variable.
SLW_oc_debounce	This is the overcurrent debounce count. This is normally not updated by the application. But this is useful in a demo kit where users may wish to test a new motor without re-compiling the firmware.

Table 13. Motor Control API Writable Variables (Continued)

Variable Name	Description
SLW_current_limit	This is the maximum current limit in units of 0.01A. This is normally not updated by the application. But this is useful in a demo kit where users may wish to test a new motor without re-compiling the firmware.
SLW_motor_max_rpm	This is the maximum RPM of the motor. This is normally not updated by the application. But this is useful in a demo kit where users may wish to test a new motor without re-compiling the firmware.
SLW_user_direction	This is a 1-bit variable used by the user application to indicate to the motor control state machine the desired direction of rotation.
SLW_rpm_updated	This is a 1-bit variable that the user application can test to check if the SLR_motor_current_rpm variable had been re-computed since the last time that this bit was cleared. The user application should clear this bit when it reads the SLR_motor_current_rpm variable. User application need not use this variable to read SLR_motor_current_rpm if the application does not care whether SLR_motor_current_rpm has been re-computed since the last time the variable was read.
SLW_pwm_updated	This is a 1-bit variable that the user application can test to check if the SLR_pwm_duty variable had been modified since the last time that this bit was cleared. The user application should clear this bit when it reads the SLR_pwm_duty variable. User application need not use this variable to read SLR_pwm_duty if the application does not care whether SLR_pwm_duty has been re-computed since the last time the variable was read.

4.2. Motor Control Demo Application Firmware Flowcharts

The flowcharts in this section provide a guide to the demo firmware application flow. The application developer can use these flowcharts and "4.3. Typical Motor Control Implementation" on page 43 as guides to implement their own application.

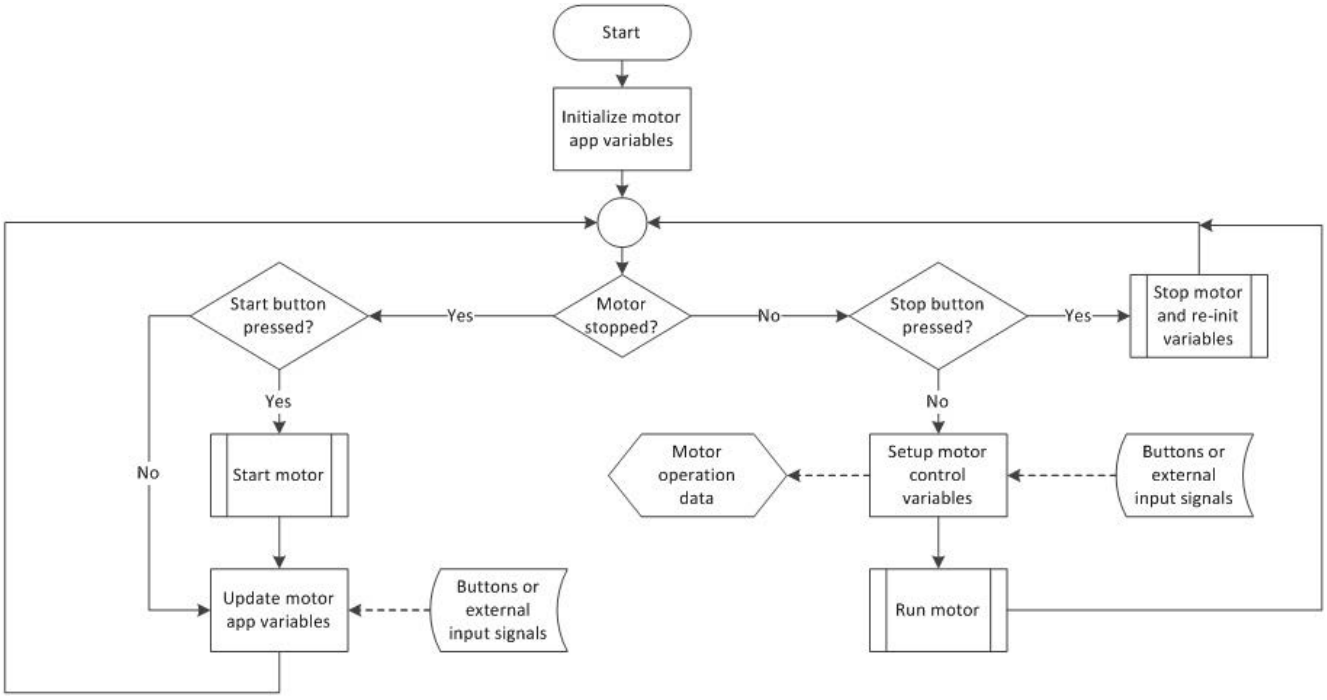


Figure 34. Demo Motor Application Top-Level Flowchart

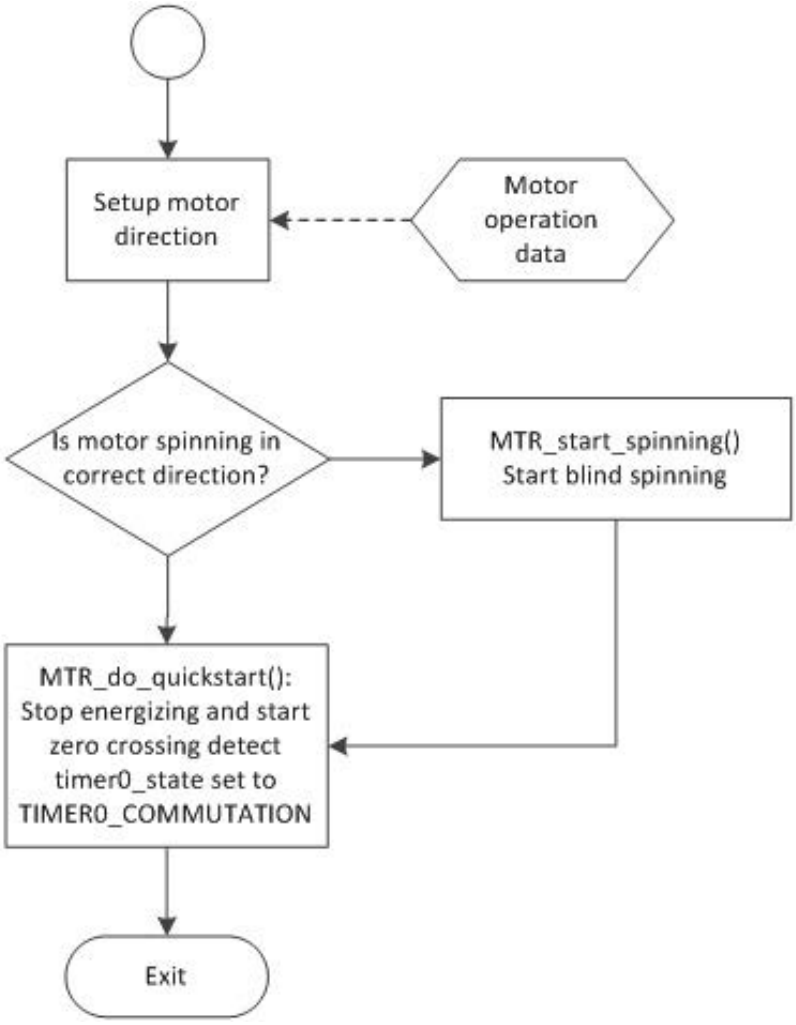


Figure 35. Start Motor Top-Level Flowchart

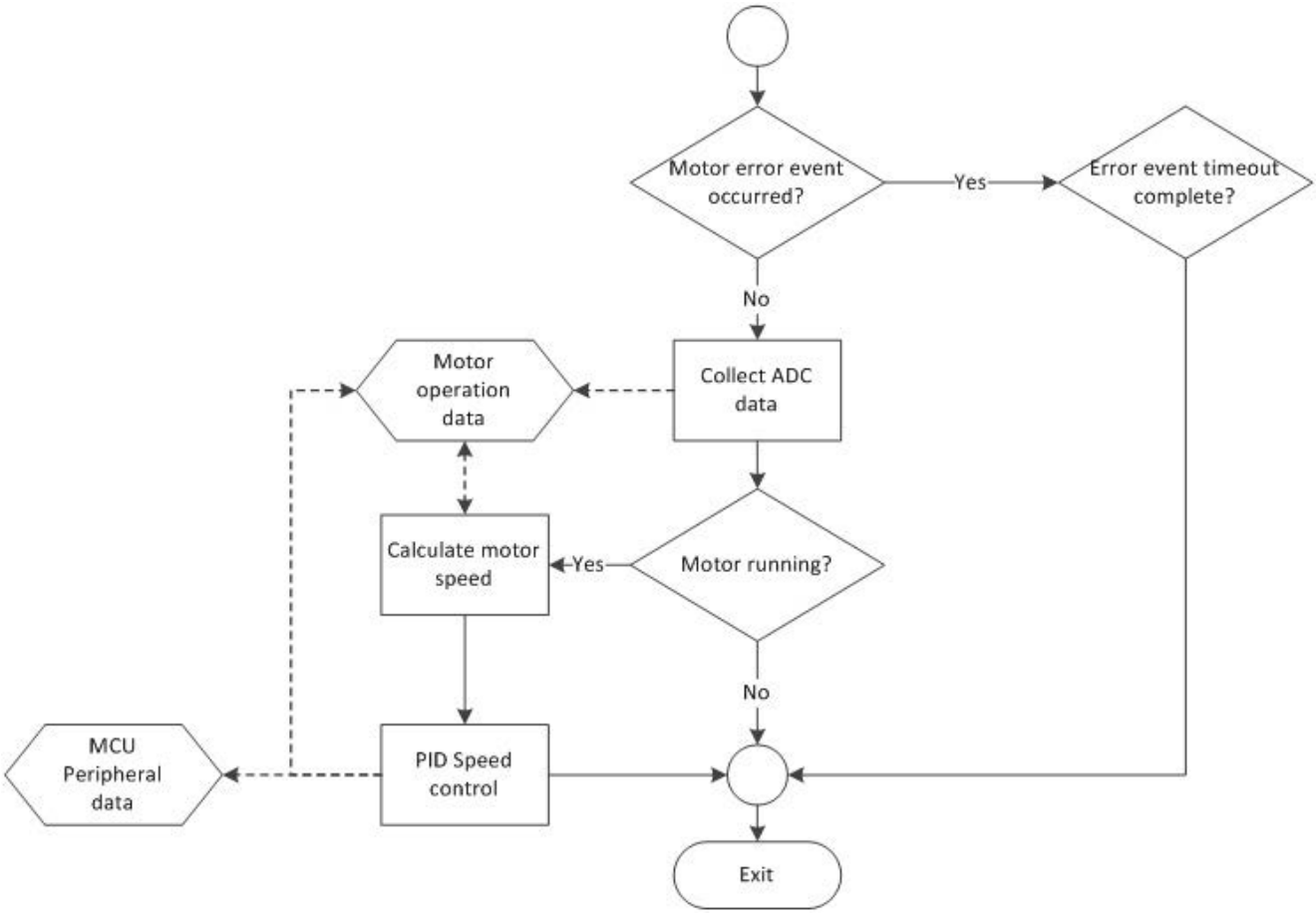


Figure 36. Run Motor Top-Level Flowchart

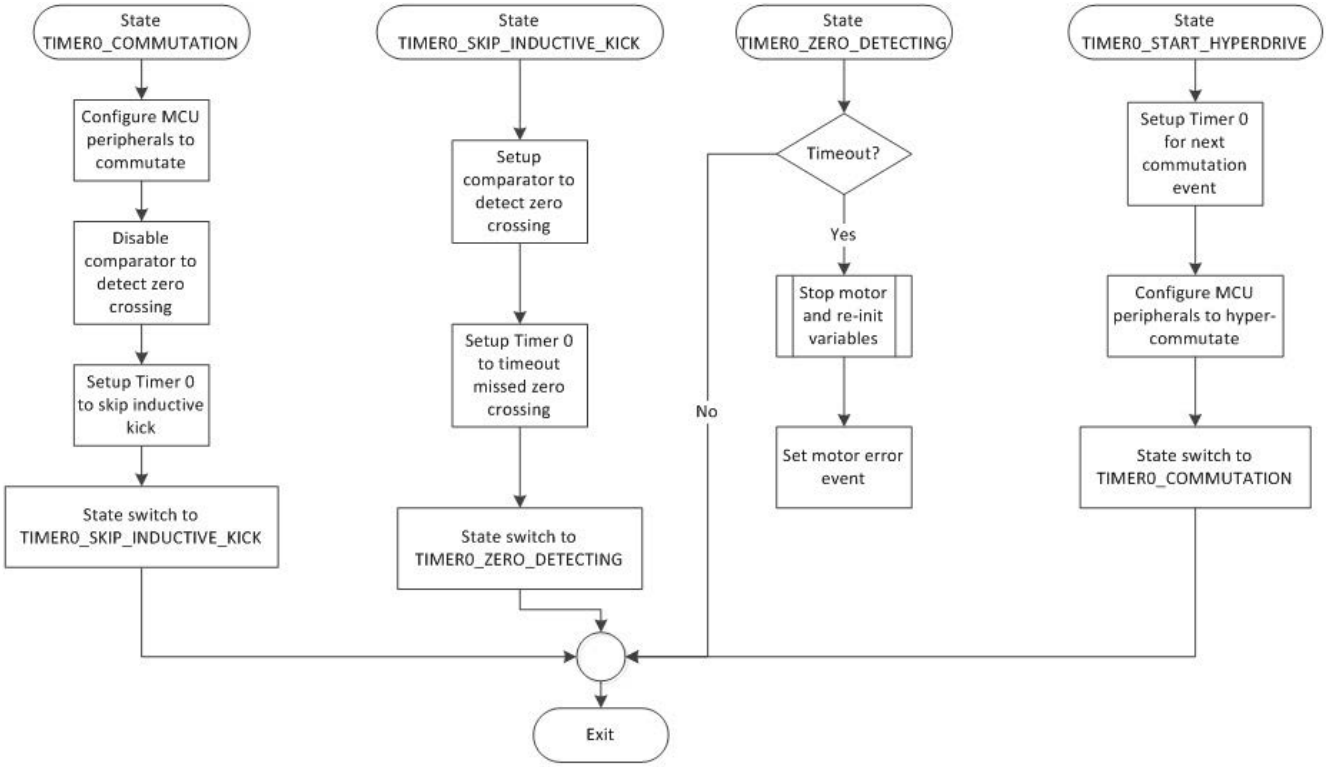


Figure 37. Timer 0 ISR Top-Level Flowchart

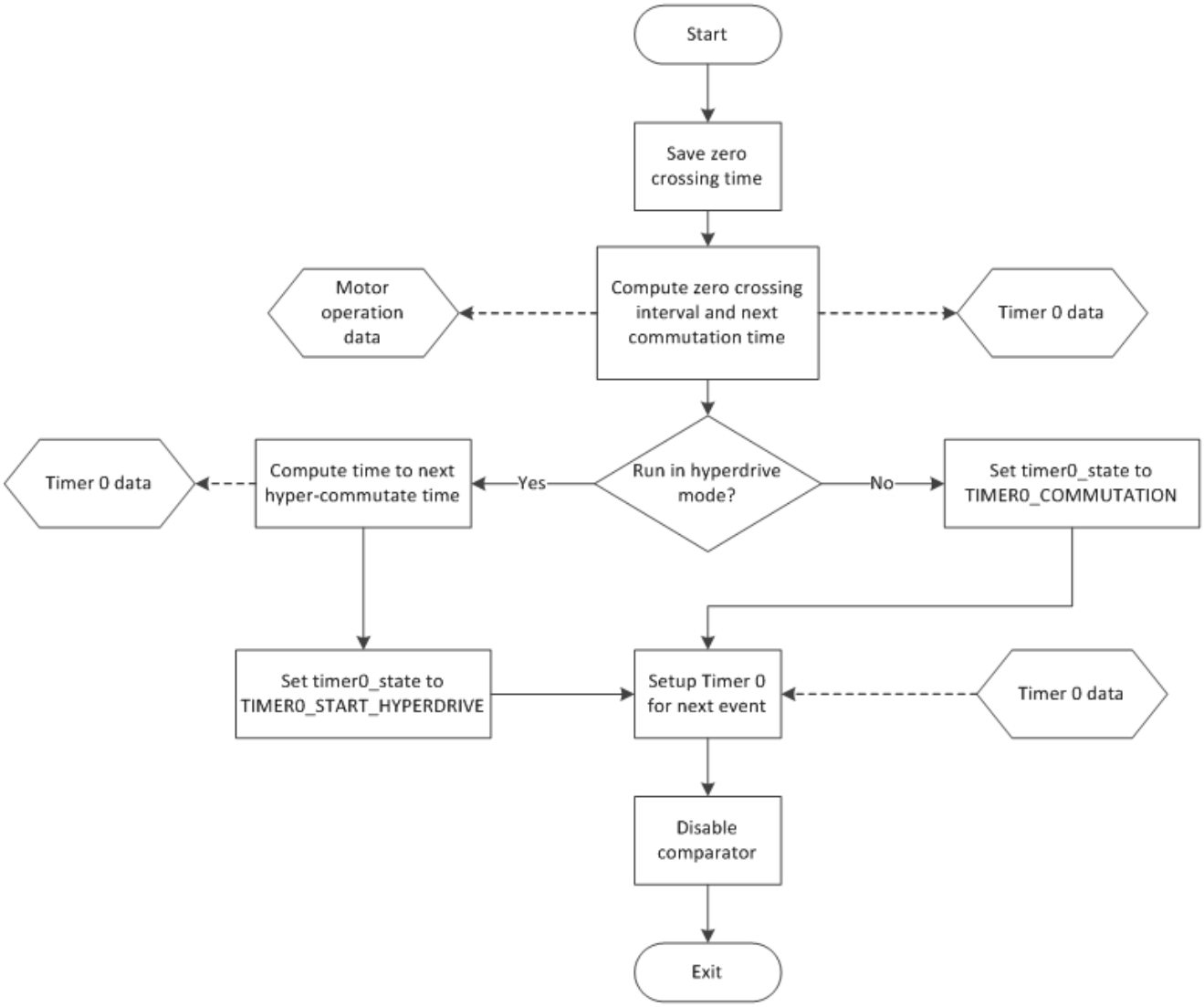


Figure 38. Comparator 0 ISR Top-Level Flowchart

4.3. Typical Motor Control Implementation

The typical motor control application implementation is expected to be based on a standard non-blocking execution loop as shown in the following code sample:

```
void main()
{
      // Application-specific peripheral initialization
      // …

      // Initialize motor state machine and PID gain values
      SL_MTR_init();
      SL_MTR_change_pid_gain(6000, 200);

      while (1)
      {
            SL_MTR_motor();

            if ( MOTOR_STOPPED == SLR_motor_state )
            {
                  /* Application-specific code for running state only such as:
                        - determine target rotation direction
                        - monitor motor voltage for undervoltage protection
                  */
                  // …

                  if ( /* event to trigger start */ )
                  {
                        // Initialize the Gains of PI controller
                        SL_MTR_change_pid_gain(pgain, igain);

                        SL_MTR_start_motor();   // Start the motor

                        // Other application-specific code for start event – if any
                  }
            }
            else if ( MOTOR_RUNNING == SLR_motor_state )
            {
                  /* Application-specific code for running state only such as:

                        - determine target rotation speed
                        - monitor motor current for overcurrent detection
                        - monitor motor voltage for undervoltage protection
                        - check current motor speed
                   */
                   // …

                   if ( /* event to trigger stop */ )
                   {
                         SL_MTR_stop_motor();    // Stop the motor

                         // Other application-specific code for stop event  – if any
                   }
             }

             // Application-specific code …

       }
 }
```

4.4. Application Firmware Configuration

The default application provided in the kit can be configured for different modes of operation. These configurations are defined in two header files, BLDC_RD_Build_Params.h and BLDC_RD_System.h. Users can modify these files in any text editor and build the firmware for desired configuration. Some of the configurations defined in BLDC_RD_Build_Params.h can also be modified by the Silicon Labs Spinner.

Note that some of configurations may require changes to the jumper settings.

4.4.1. Speed Control Type

The firmware can be configured to accept 1 of 2 different types of speed control input: motor speed command or duty cycle of the motor PWM signal. The user can select the speed control type at build-time by defining:

```
#define BLDC_RD_RPM_OR_PWM      XXX_PARAMETER
```

where XXX_PARAMETER can be one of the following:

- RPM_PARAMETER—User selects motor rotation speed as the type of speed control user command.
- PWM_PARAMETER—User selects motor PWM duty cycle as the type of speed control user command

This configuration is defined in BLDC_RD_Build_Params.h.

4.4.2. Speed Control Command Source

The firmware supports different input sources of the Speed Control Command:

- External PWM signal
- External analog signal (this is controlled by the potentiometer on the kit)

The user can select the speed control command source at build-time by defining:

```
#define BLDC_RD_RPM_PWM_SRC      XXX_SPEED_SOURCE
```

where XXX_SPEED_SOURCE can be one of the following:

- PWM_SPEED_SOURCE—External PWM signal sets the target value of Speed Control Parameter.
- POT_SPEED_SOURCE—A potentiometer on BLDC kit sets the target value of Speed Control Parameter.

Note that configuring for PWM_SPEED_SOURCE or POT_SPEED_SOURCE requires a change in the jumper setting at J108.

This configuration is defined in BLDC_RD_Build_Params.h.

4.4.3. Build for Protocol

The firmware implements a protocol to interact with the Silicon Labs Spinner. This protocol can be enabled at build-time by defining:

```
#define BUILD_FOR_PROTOCOL
```

This configuration is defined in BLDC_RD_Build_Params.h.

4.4.4. Overcurrent Detection

The kit firmware can measure current and detect over-current condition. This feature is enabled at build-time by defining:

```
#define FEATURE_OVERCURRENT
```

If this feature is not required by user application then this line can be removed or commented. Note that this feature is automatically disabled if the FEATURE_PID_TUNE_FUNCTION is enabled - this is because of code memory limitations.

When this feature is enabled, the current is measured on a user-assigned pin. This signal is assigned to P0.6 of the MCU in the BLDC kit. Users can configure the pin assignment as below:

```
#define IMEAS_ADCMX      6
```

When this feature is enabled, the appropriate PxMDIN assignment is automatically handled by the PxMDIN_INIT_VAL enumerated type definition in BLDC_RD_System.h header file.

When an op-amp is used to amplify current signal, the gain of the Op-Amp can also be configured as:

```
#define OP_AMP_GAIN      (5.1/3.3)
```

If no op-amp is used, OP_AMP_GAIN should be set to 1.

These configurations are defined in BLDC_RD_System.h.

4.4.5. Motor Stall Detection

The motor stall detection is enabled at build-time by defining:

```
#define FEATURE_RPM_STALL_DETECTION
```

If this feature is not required by user application then this line can be removed or commented. This feature is automatically disabled if BLDC_RD_RPM_OR_PWM is defined as PWM_PARAMETER. Motor stall condition is detected by determining whether any increase in motor current is matched by a corresponding increase in motor speed. Motor stall detection can be tuned by the following parameters:

```
#define STALL_CHECK_COUNT               100
#define COMPENSATION_CONSTANT_FACTOR    (1UL * 65536UL / 100)
#define DELTA_CURRENT_FACTOR_K          (70UL * 65536UL / (100UL * CURRENT_UNIT *
MOTOR_MAX_CURRENT))
```

STALL_CHECK_COUNT defines the number of motor current samples to collect before the algorithm to detect stall is executed. To increase sensitivity to detect motor stall, users can either decrease: COMPENSATION_CONSTANT_FACTOR or increase: DELTA_CURRENT_FACTOR_K.

4.4.6. Motor Voltage Measurement

The kit firmware can measure the motor operating voltage. This feature is enabled at build-time by defining:

```
#define FEATURE_MEAS_VMDC
```

If this feature is not required by user application then this line can be removed or commented. Note that this feature is automatically disabled if the FEATURE_PID_TUNE_FUNCTION is enabled - this is because of code memory limitations.

When this feature is enabled, the voltage is measured on a user-assigned pin. This signal is assigned to P0.7 of the MCU in the BLDC kit. Users can configure the pin assignment as below:

```
#define VMDC_ADCMX      7
```

When this feature is enabled, the appropriate PxMDIN assignment is automatically handled by the PxMDIN_INIT_VAL enumerated type definition in BLDC_RD_System.h header file.

These configurations are defined in BLDC_RD_System.h.

4.4.7. Potentiometer Measurement

The kit firmware can measure a voltage of a potentiometer - this is required if the speed control command source (section 4.3.2) is an analog voltage input. This feature is enabled at build-time by defining:

```
#define FEATURE_MEAS_POT
```

If this feature is not required by user application then this line can be removed or commented. When this feature is enabled, the voltage is measured on a user-assigned pin. This signal is assigned to P1.0 of the MCU in the BLDC kit. Users can configure the pin assignment as below:

```
#define POT_ADCMX      8
```

When this feature is enabled, the appropriate PxMDIN assignment is automatically handled by the PxMDIN_INIT_VAL enumerated type definition in BLDC_RD_System.h header file.

These configurations are defined in BLDC_RD_System.h.

4.4.8. Motor-Specific Configurations

In addition to BLDC_RD_NUM_POLES, following configurations depend on motor selected by user. Users should define additional headroom of 10% to 20% for the MOTOR_MAX_SPEED definition; this allows an opportunity for hyperdrive to kick in if this feature is enabled.

```
#define MOTOR_MAX_CURRENT    10.0
#define MOTOR_MAX_SPEED    (54720)
#define MOTOR_MAX_RPM ((U16)(MOTOR_MAX_SPEED / SPEED_UNIT))
```

where MOTOR_MAX_CURRENT and MOTOR_MAX_SPEED are appropriate values given in the motor specifications and system design.

These configurations are defined in BLDC_RD_System.h.

4.4.9. Buttons

The BLDC kit firmware supports a 2 button user interface: Button0 (SW101) and Button1 (SW102). The default application firmware uses Button0 for direction control (if Direction Command Source is configured to use button) and Button1 to start and stop the motor. These features can be enabled at build-time by defining:

```
#define FEATURE_BTN0
#define FEATURE_BTN1
```

If any button is not required by the user application then the corresponding line can be removed or commented.

When the button feature is enabled, the buttons are assigned to user-defined pins as:

```
#define BTN0_PORT    P1
#define BTN0_BIT     1
#define BTN1_PORT    P2
#define BTN1_BIT     1
```

The following helper macros will be defined to aid the user in writing firmware to use these buttons:

```
CONFIG_BTN0()
IS_BTN0_PRESSED()
CONFIG_BTN1()
IS_BTN1_PRESSED()
```

But if the button feature is not enabled, then the corresponding CONFIG_BTNx() macro will be empty and IS_BTNx_PRESSED() will always return 0.

These configurations are defined in BLDC_RD_System.h.

4.5. Motor Control Module Configuration

When users are ready to implement their motor control design and application, the default application and design as supplied by the kit is not likely to be compatible to their requirements - for example, BLDC fans do not normally have buttons. The kit provides design-specific configuration of the motor control module (motor.c, mtrpid.c) for different modes of operation. These configurations are defined in two header files, BLDC_RD_Build_Params.h and BLDC_RD_System.h. Users can modify these files in any text editor and build the firmware in a suitable configuration.

4.5.1. PWM Scheme

The firmware supports 3 different PWM schemes: high-side, low-side or mixed mode. The different PWM schemes are discussed earlier in section 2.2. The user can select the PWM scheme at build-time by defining:

```
#define BLDC_RD_PWM_METHOD     H_BRIDGE_XXX_PWM
```

where `H_BRIDGE_XXX_PWM` can be one of the following:

`H_BRIDGE_HIGH_SIDE_PWM - High-side only PWM.`

`H_BRIDGE_LOW_SIDE_PWM - Low-side only PWM.`

`H_BRIDGE_MIXED_MODE_PWM - Mixed-mode PWM.`

This configuration is defined in BLDC_RD_Build_Params.h.

4.5.2. Commutation Method

The firmware supports 2 different methods of commutation: zero-crossing timing, or detection by Hall sensors. The user can select the commutation method at build-time by defining:

```
#define BLDC_RD_COMMUT_METHOD     COMMUTATION_BY_XXX
```

where `COMMUTATION_BY_XXX` can be one of the following:

`COMMUTATION_BY_COUNTDOWN` - Commutate using zero-crossing timing method.

`COMMUTATION_BY_HALL` - Commutate using detection by Hall sensors method.

This configuration is defined in BLDC_RD_Build_Params.h.

4.5.3. Number of Poles

Different BLDC motors are constructed with different number of poles. The motor rotation speed is calculated based on the number of poles. The user can select the number of poles at build-time by defining:

```
#define BLDC_RD_NUM_POLES     N
```

where `N` is the number of poles. If motor specification mentions pole-pairs then N should be two times that number.

This configuration is defined in BLDC_RD_Build_Params.h.

4.5.4. Frequency Generator Signal

This configuration enables FG feature - a feature commonly used in fan applications to output a digital signal that toggles ever 3 commutations (i.e. 1 cycle every 1 motor electrical cycle). This feature is enabled at build-time as:

```
#define FEATURE_FG
```

If the FG signal is not required by user application then this line can be removed or commented.

When this feature is enabled, it generates FG signal on a user-assigned pin. This signal is assigned to P2.0 of the MCU in the BLDC kit. Users can configure the pin assignment as below:

```
#define FG_PORT     P2
#define FG_BIT      0
```

When this feature is enabled, the following macros will be defined:

```
CONFIG_FG()
SET_FG()
CLR_FG()
TOGGLE_FG()
```

These macros will be used by the motor control module to toggle the FG pin - so users need not change the motor control source code to use this feature. If this feature is not required (i.e. FEATURE_FG is not defined), then these macros will be empty - hence, no code is generated even though the motor control source code call these macros.

All these configurations are defined in BLDC_RD_System.h.

4.5.5. Motor Startup Current Control Pin

The kit firmware implements a motor startup technique that is based on limiting the current through the motor (see section 3.5). This requires some form of motor current measurement to be available to the MCU so that the comparator CMP0 can be used to implement this feature. This motor current measurement pin must be assigned to a pin on Port 0 because only CMP0 implements the comparator clear functionality. It is possible to assign this pin to the same pin as the over current detection pin (see "4.4.4. Overcurrent Detection" on page 44). This pin is assigned to P0.6 of the MCU in the BLDC kit. Users can configure the pin assignment as below:

```
#define CPT0MX_IMEASURE      6
```

This configuration is defined in BLDC_RD_System.h.

4.5.6. Hyperdrive Mode

The hyperdrive mode can be used to achieve higher speed. This feature is enabled at build-time by defining:

```
#define FEATURE_HYPERDRIVE
```

If this feature is not required by user application then this line can be removed or commented. Note that this feature is automatically disabled if the `FEATURE_PID_TUNE_FUNCTION` is enabled - this is because of code memory limitations.

These configurations are defined in BLDC_RD_System.h.

4.5.7. Motor Gate Drive Peripheral Assignment

One PCA channel is used to generate the motor PWM signal. This PCA channel can be modified by users for their application:

```
#define MOTPWM_CHANNEL     1
```

Note that channel 0 is reserved for blanking signal.

The motor gate drive pins must be assigned to Port 1, but the pins can be assigned by users for their design as follows:

```
#define MOTDRV_AL_PIN    2
#define MOTDRV_AH_PIN    3
#define MOTDRV_BL_PIN    4
#define MOTDRV_BH_PIN    5
#define MOTDRV_CL_PIN    6
#define MOTDRV_CH_PIN    7
```

The active level of the motor gate drive pins can also be defined as high (1) or low (0) to match the hardware design:

```
#define MOTDRV_LOW_ACT     1
#define MOTDRV_HIGH_ACT    1
```

These configurations are defined in BLDC_RD_System.h.

4.5.8. Filtered Back-EMF Pins

The filtered back-EMF signals can be assigned to Port 0 pins by users for their design:

```
#define FILTERED_A_PIN    0
#define FILTERED_B_PIN    1
#define FILTERED_C_PIN    2
#define FILTERED_Y_PIN    3
```

These configurations are defined in BLDC_RD_System.h.

4.5.9. Rotation Speed Resolution

The BLDC kit supports motor rotation speeds of up to 200000 RPM. However, the firmware uses only 16-bit variables for motor speed to optimize code memory and speed. Hence, to support high speeds, the firmware defines the smallest unit of RPM that the system measures as:

```
#define SPEED_UNIT    10
```

This configuration is defined in BLDC_RD_System.h. The Spinner application assumes this value to be 10.

5. Silicon Labs Spinner

The Silicon Labs Spinner provides a GUI interface to configure some of the build-time parameters in the firmware. The changes must be recompiled into a new firmware and downloaded into the MCU. The tool also provides an interface to view run-time parameters and plot them in real-time.

5.1. Build-Time Configuration

Build-time configurations are set in the Create New Project dialog. This dialog is activated through the “File | Create New Project...” menu item. When Generate Project button is clicked (after selecting appropriate configurations), the entire firmware project is created in the project directory with new settings within BLDC_RD_Build_Params.h. Users can use the Silicon Labs IDE to open the workspace f85x_bldc.wsp in the project directory to rebuild a new firmware image.

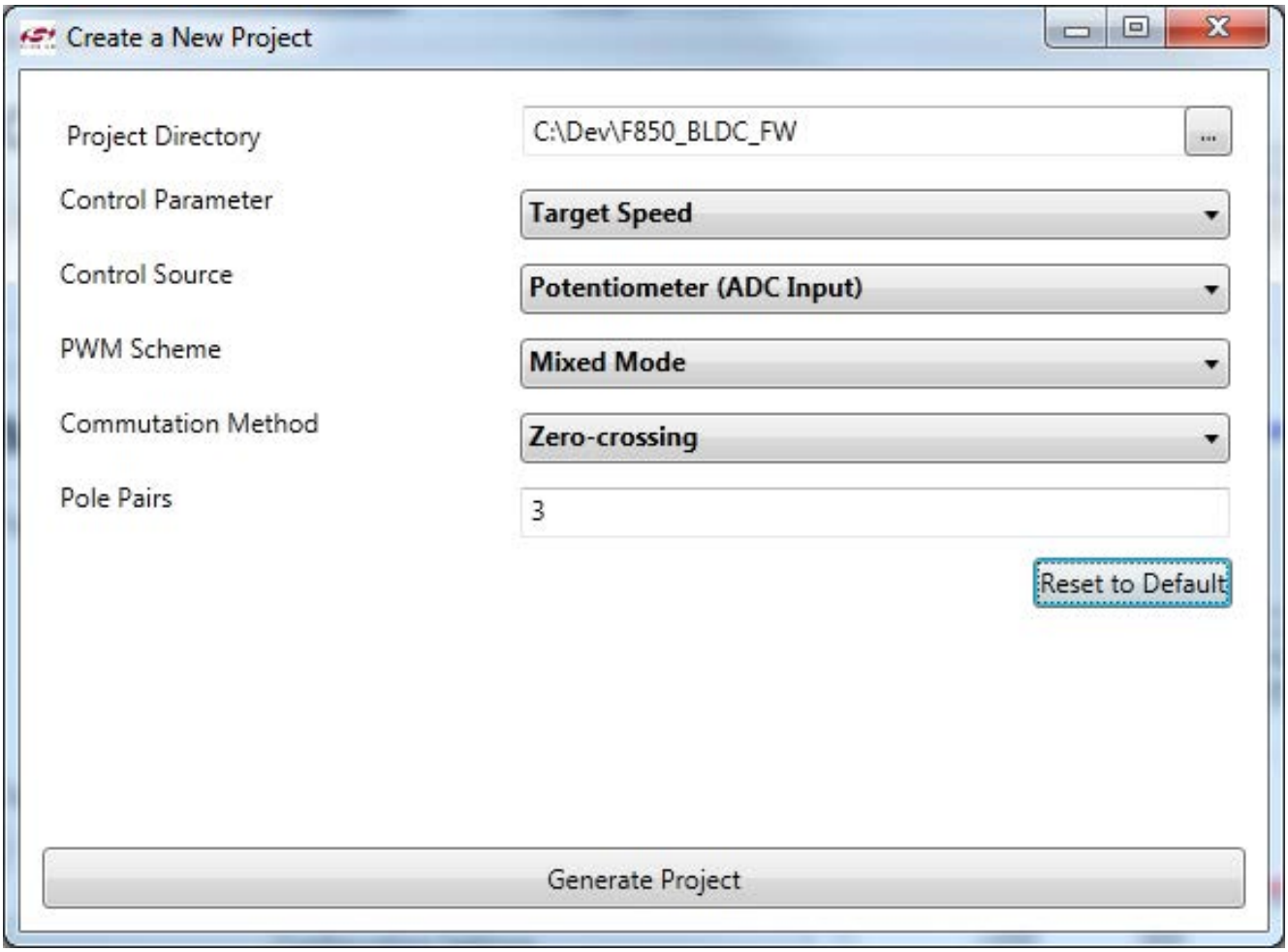


Figure 39. Silicon Labs Spinner Build-Time Configuration

Most of the build-time configuration has been covered in "4.3. Typical Motor Control Implementation" on page 43. The”Project Directory” text box field lets the user specify the directory in which the firmware project should be generated.

5.2. Runtime Monitor and Control

Run-time parameters can be read, plotted, and set to desired values using the Easy Mode view or the Advanced Mode view. The appropriate view can be selected from the View menu.

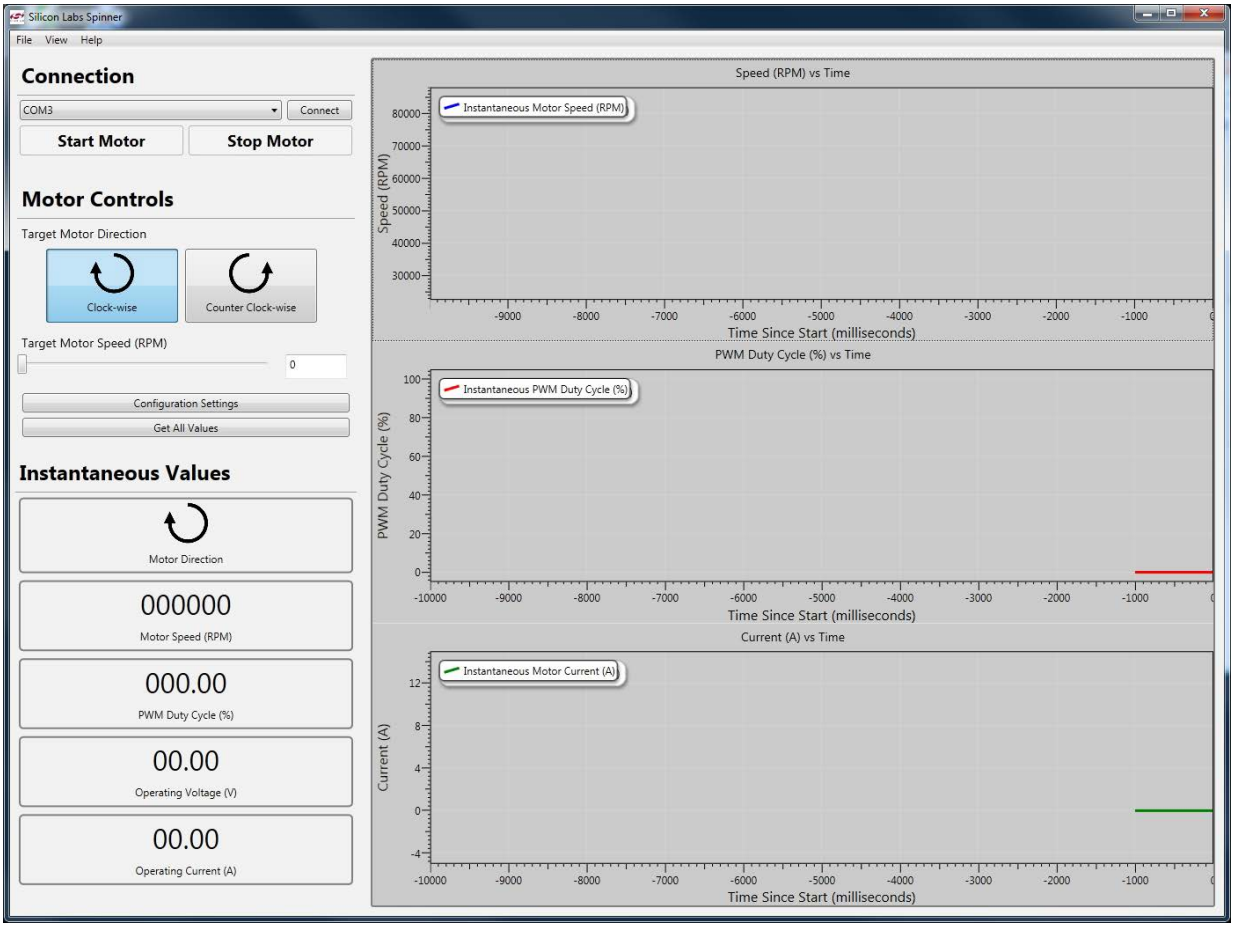


Figure 40. Easy Mode View

After the BLDC kit is powered and connected to a host PC, the user can establish a session with the BLDC kit to operate the motor. This is accomplished by performing the following steps with the application (Easy Mode view):

1. Connect a USB cable between the PC and the MCU board.
2. Apply power to the powertrain board (must be done after Step 1).
3. Use Device Manager to identify the COM Port: “Silicon Labs CP210x USB to UART Bridge”.
4. Select the COM port found in Step 3, and click the “Connect” button.
5. Select the appropriate rotational direction.
6. Select the target motor speed.
7. Click the “Start Motor” button to run the motor.
8. The user can change the target motor speed while the motor is running.
9. The user can click the “Stop Motor” button to stop the motor while the motor is running.
10. The user can also press the “Start/Stop” pushbutton on the board to stop the motor while the motor is running. Refer to Figure 44, “User Interface for Standalone Operation of BLDC Demo Kit,” on page 55 for the location of this pushbutton on the board.

The “Configuration Settings” button allows the user to change the “Maximum Motor Speed” and “Pole Pairs” parameters; these parameters must be changed when testing a different motor model.

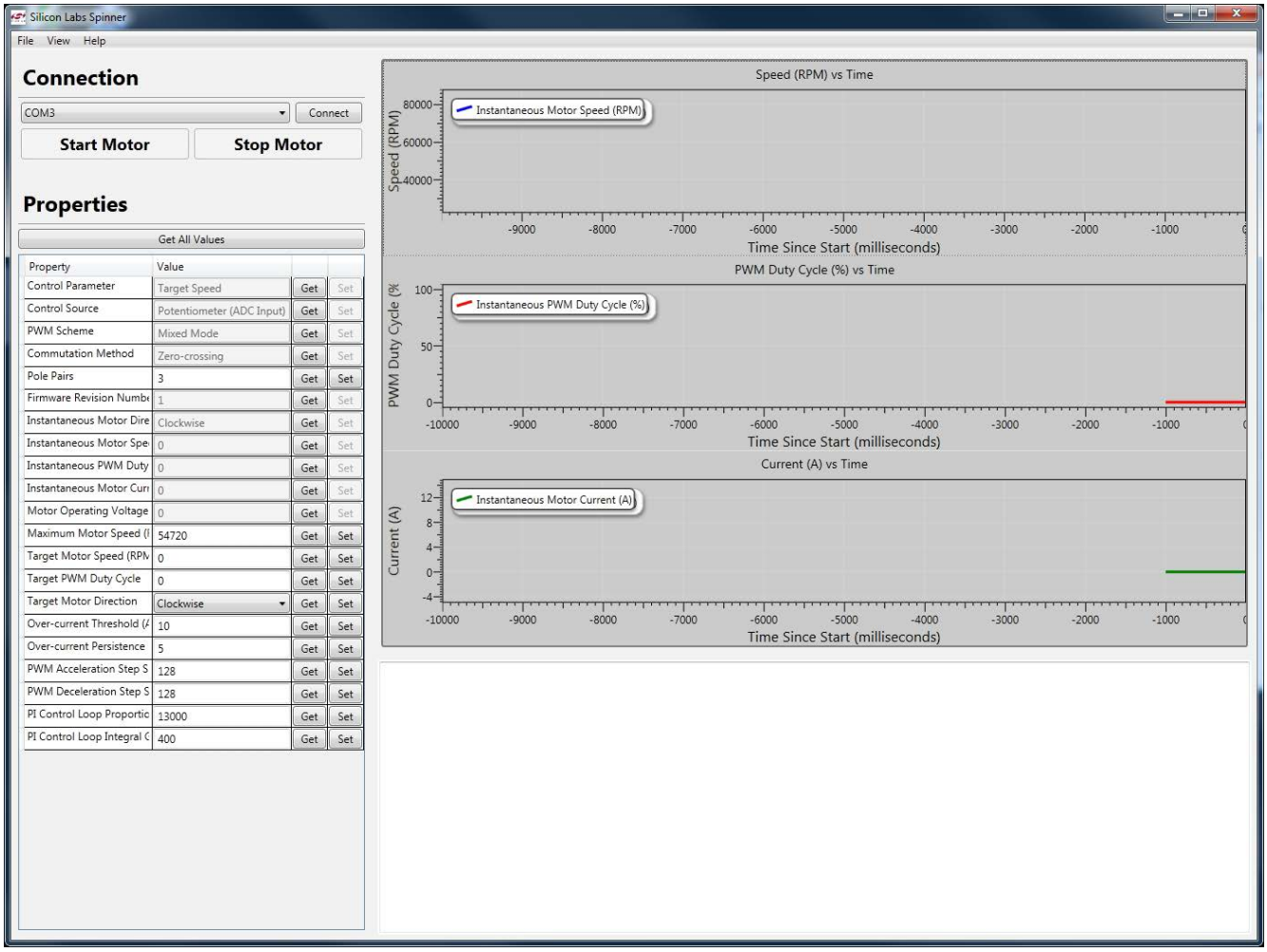


Figure 41. Advanced Mode View

In the Advanced Mode view, there are more parameters available for the user to change. The “Get” buttons allow the user to read parameters from the MCU, and the “Set” buttons allow the user to write parameters to the MCU. These parameters have been discussed in "4.4. Application Firmware Configuration" on page 44 and "4.5. Motor Control Module Configuration" on page 47.

6. Operating Another BLDC Motor

The kit can be used to operate any 3-phase BLDC motor that conforms to the following operation limits:

- Maximum Motor voltage: ≤ 24 Vdc
- Maximum average current: ≤ 10 A
- Maximum speed: 200000 RPM or lower for a 2-pole BLDC motor

If the motor current is expected to exceed 5 A, then a suitable external power supply must be used as a power source for the kit. The power source can be connected through one of the two power connections shown in Figure 42.

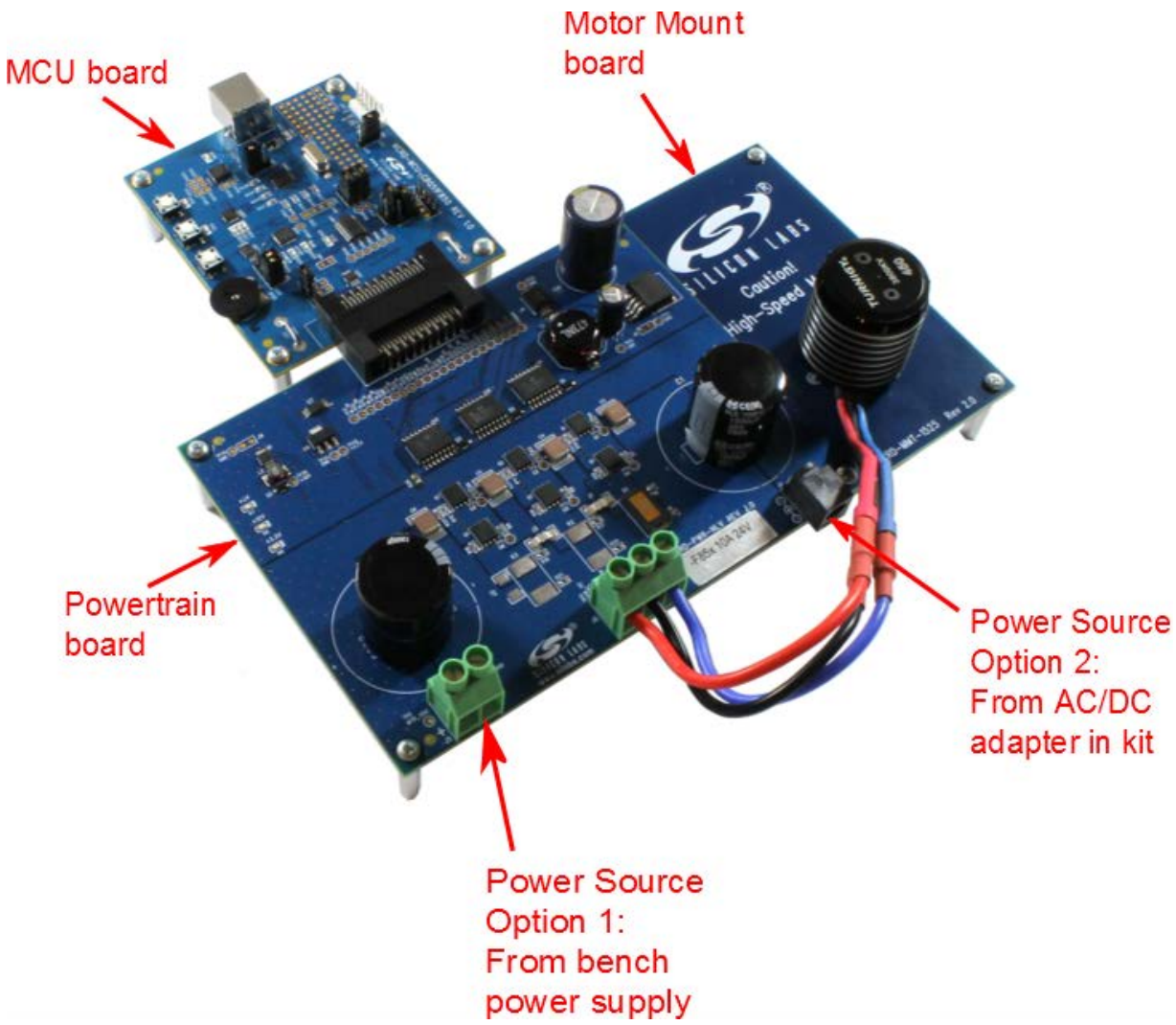


Figure 42. Power Source Options for BLDC Demo Kit

The power supply and the USB connection to PC must be connected in the following sequence:

1. Connect USB to PC; then
2. Apply power to the powertrain board

The user must run the Silicon Labs Spinner to connect to the kit and change 2 key parameters to match the motor specifications:

- Number of poles in the motor
- Maximum operation speed of the motor (i.e. Maximum RPM)

Click on the “Configuration Settings” button in the Easy Mode view to change these two parameters.

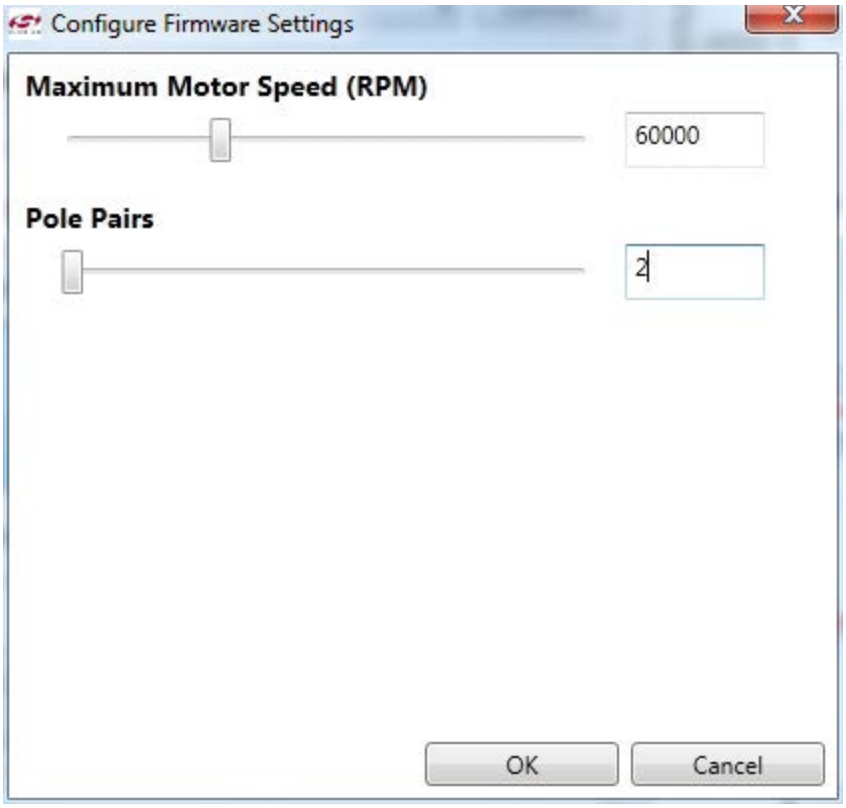


Figure 43. Pending Screenshot

After these two parameters have been changed to match the motor specifications, the user can operate the motor using the Start/Stop button, Direction button and potentiometer as shown in Figure 44.

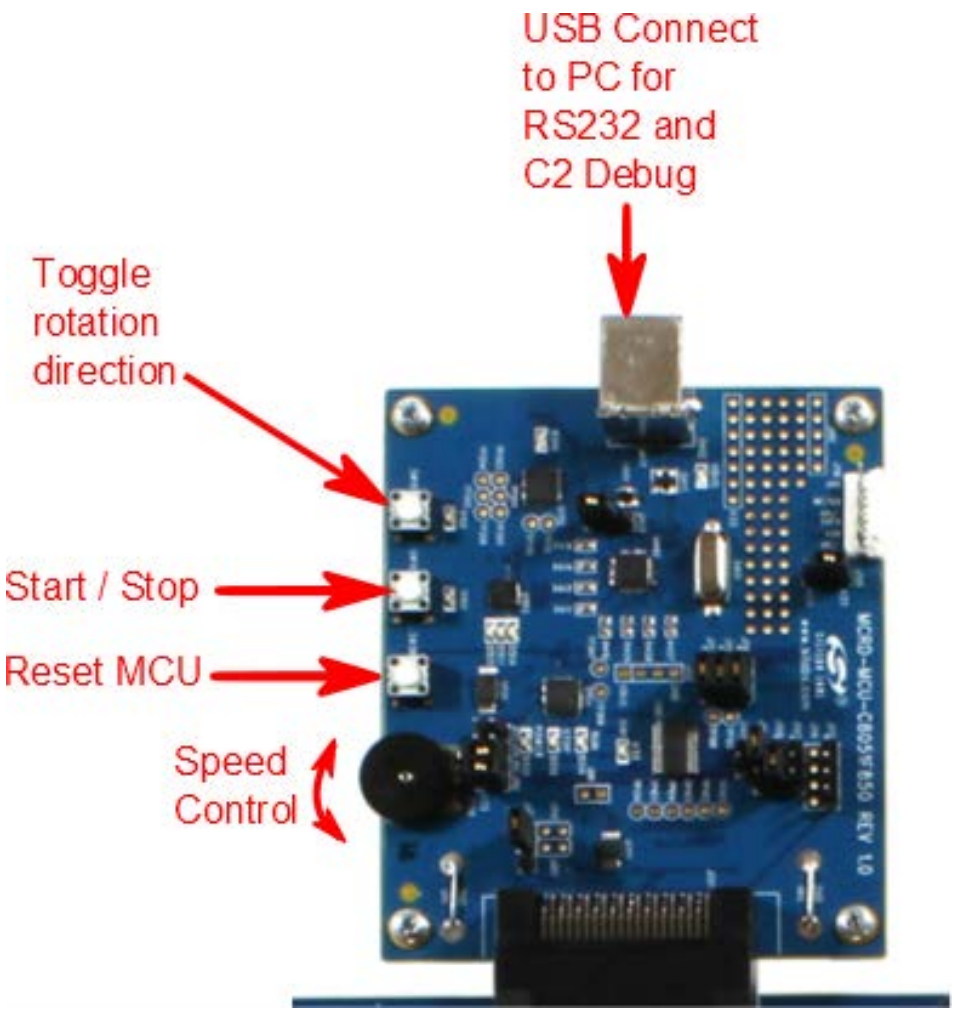


Figure 44. User Interface for Standalone Operation of BLDC Demo Kit

The details of the user interface behavior are described in Table 14.

Table 14. BLDC Motor Kit user Interface for Standalone Operation

User Interface	Description
SW101	Pushbutton. Controls the direction of rotation of the motor. Pressing this button reverses the direction of rotation. This button is read by the firmware only when the motor is not running.
SW102	Pushbutton. Starts/stops the motor.
SW301	Pushbutton. Reset button of the C8051F850
R109	Rotary potentiometer. Controls the speed of the motor. Clockwise direction increases the speed of the motor.
TP308	Header pin. This is the C2D pin of the F850. The motor firmware synchronizes output of this pin with the electrical cycle of the motor so that the user can monitor the speed of rotation of the motor. The rotor speed can be calculated by this formula: Rotor speed (RPM) = (Frequency of TP308 in Hz) x 120 / (Number of poles in motor)

7. Hardware Design Guide

There are a few key design parameters to observe in the hardware design if the user wishes to take advantage of the unique features of the C8051F850 and the firmware. The kit includes a spreadsheet tool to guide the user to select the optimal resistor values to use in the design. The spreadsheet is located in *<Install Directory>\F850-BLDC-RD\DesignTools\bldcdesign.xlsx*, where *<Install Directory>* is *C:\Silabs\MCU* if the default option is chosen during installation.

7.1. Design of Op-amp Gain for Current Sense Circuit

"3.2.3. Current Sensing Circuit" on page 24 discussed the operation of the current sense circuit, while "3.5. BLDC Motor Startup Technique" on page 28 explained how the current sensing is applied in the motor startup technique as used by the firmware. The design of the op-amp gain and offset is such that 2 times the maximum motor current will result in a 1.8 V output at the op-amp. The design goal is to match the op-amp voltage output to the internal C8051F850 comparator voltage reference (1.8 V) so that this voltage can be used as the trip point for the motor startup technique without using an additional pin to provide a voltage reference. The spreadsheet tool included in the kit can be used to guide the user to select the correct resistor values for the op-amp circuit.

7.2. Design of BEMF Zero-Crossing Detection Circuit

Sections "3.1.2. Back-EMF Filter Circuit" on page 21 and "3.1.3. Virtual Neutral Filter Circuit" on page 22 describe the Back-EMF filter circuit and the Virtual neutral filter circuit respectively. The design goals of these two circuits are as follows:

- The filtered voltages do not exceed the MCU operating voltage even when the motor supply voltage is at maximum voltage. The design tool allows the designer to specify some headroom to compensate for voltage spikes at the motor terminals.
- The offset of the 2 circuits must be the same (see equations 37 and 38 for the offset voltage formula)
- The gain of the virtual neutral filter circuit must be 1/3 of the gain of the BEMF filter circuit

The spreadsheet tool includes an option to reduce the number of resistors in the virtual neutral resistor network from 9 resistors to 5 resistors. This option is meant for designs that have limited PCB space for layout of all components. This trades off a slight inaccuracy in the zero-crossing detection in return for reduced component count and PCB footprint.

7.3. Low-Cost, Low-Current Design

The powertrain of the reference design kit was based on a design that had to operate at 80 V motor voltage. The design of this kit was then scaled down to meet the requirements of the design kit (24 V, 10 A). However, it is possible to reduce costs further in some applications.

Figure 53 on page 65 shows a schematic of a low-cost version of a design for low-current motors. The key changes in this version are:

- Replace high-side N-channel power MOSFETs with P-channel MOSFETs. This simplifies the design and lowers the cost of the high-side gate drivers.
- Replace the Si8230 isolated gate drivers with discrete components.

The spreadsheet tool includes a guide to help users design the gate drivers of these MOSFETs.

8. Schematics

8.1. Powertrain Board Schematics

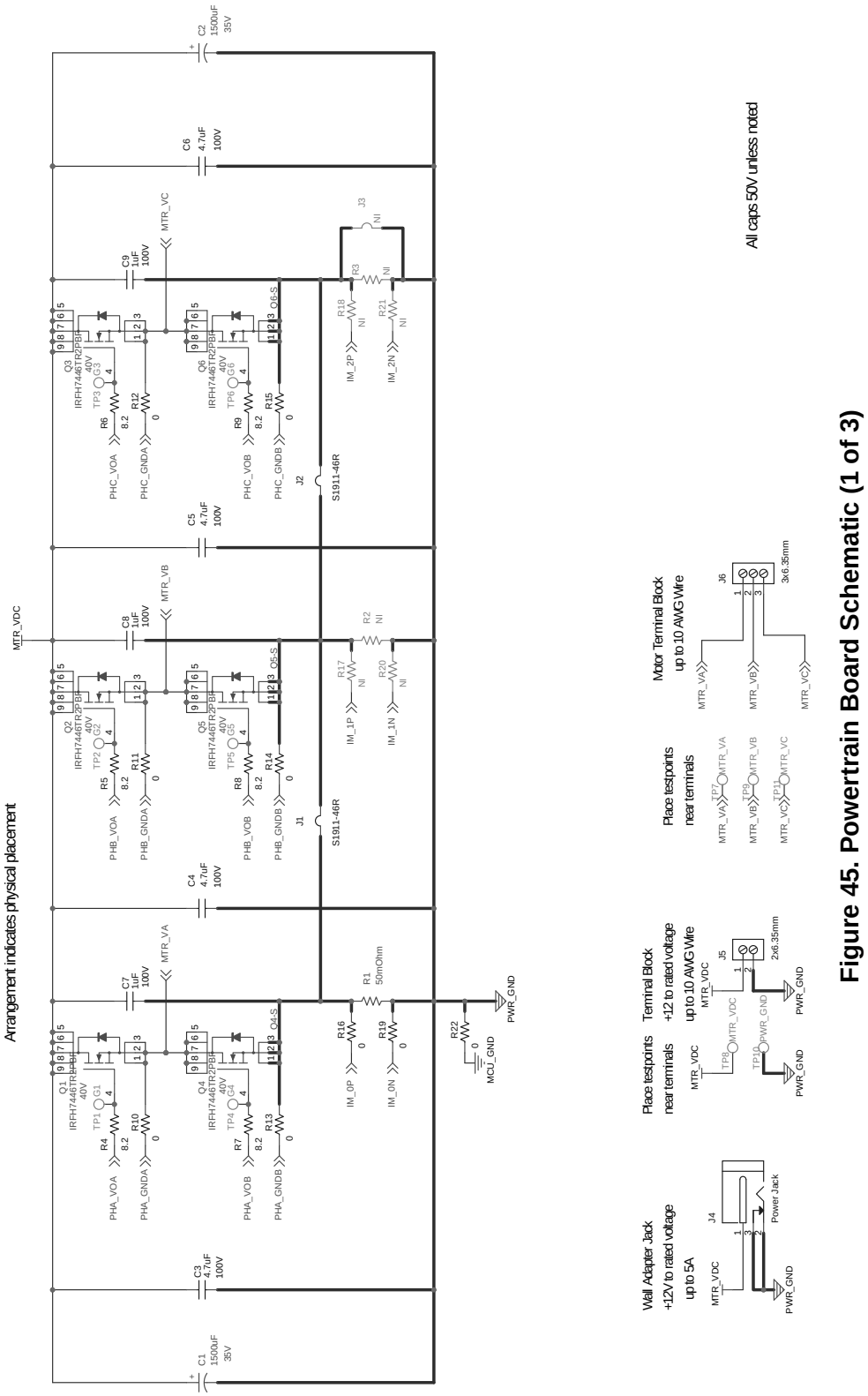


Figure 45. Powertrain Board Schematic (1 of 3)

]

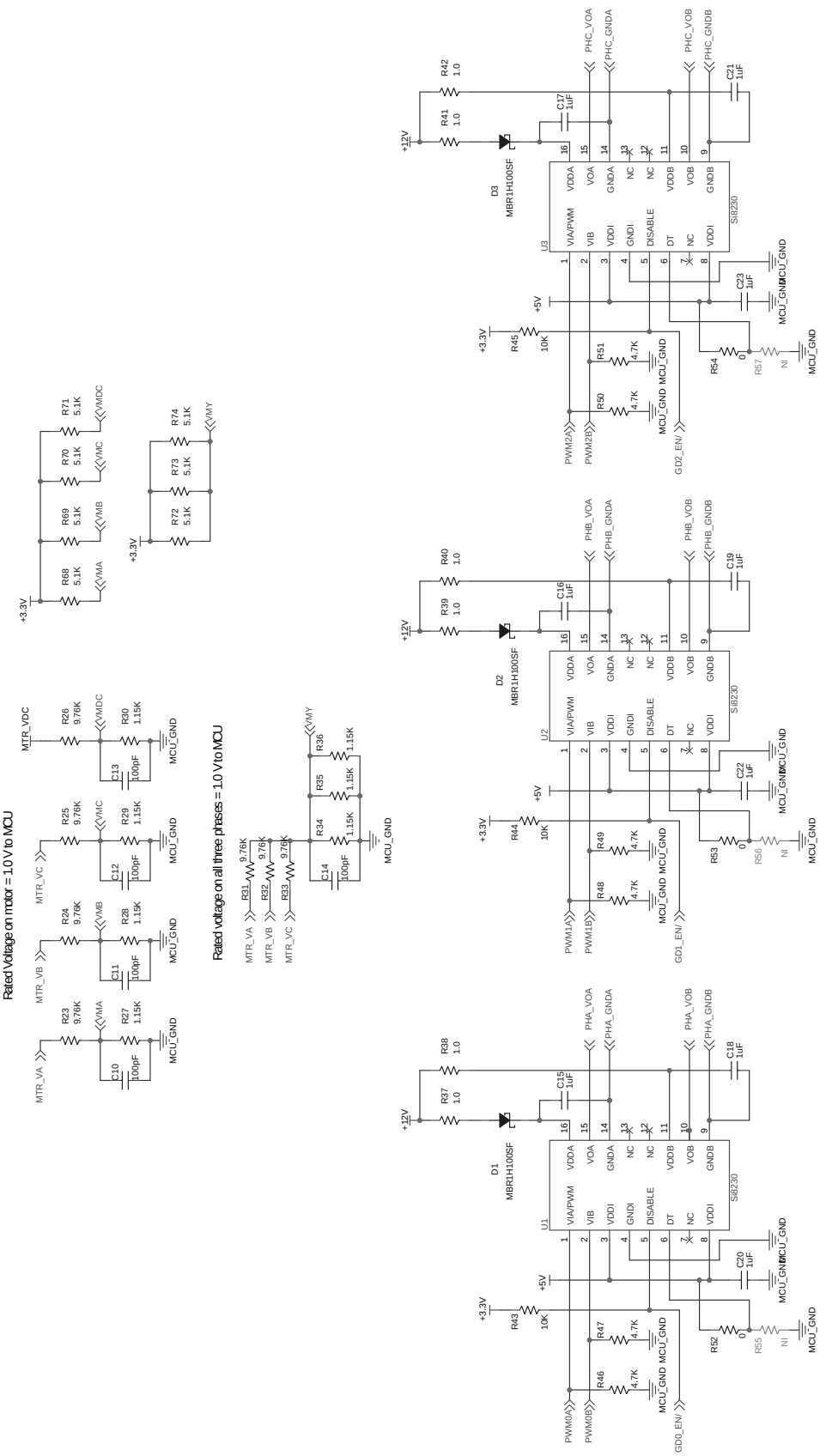


Figure 46. Powertrain Board Schematic (2 of 3)

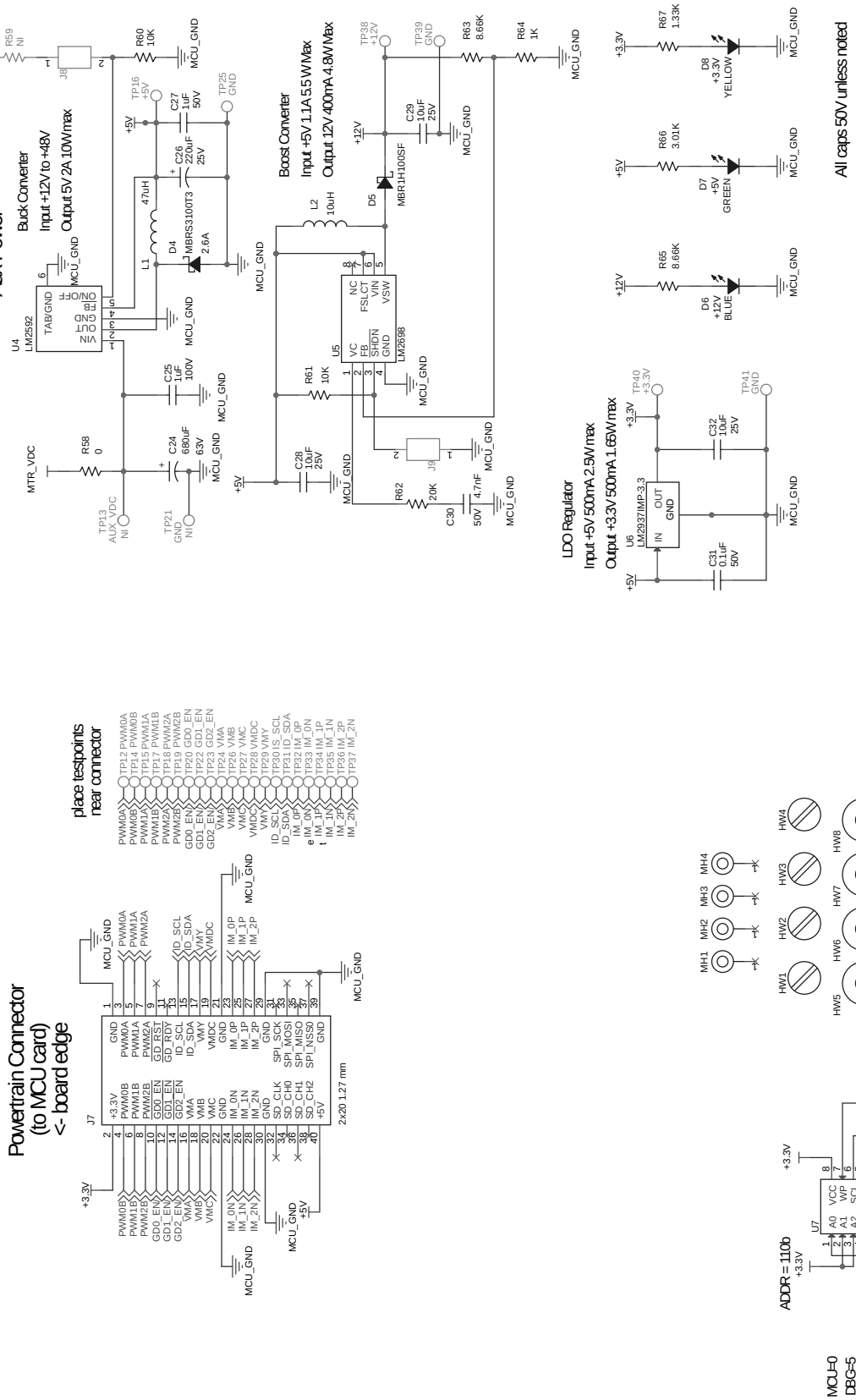


Figure 47. Powertrain Board Schematic (3 of 3)

8.2. MCU Board Schematics

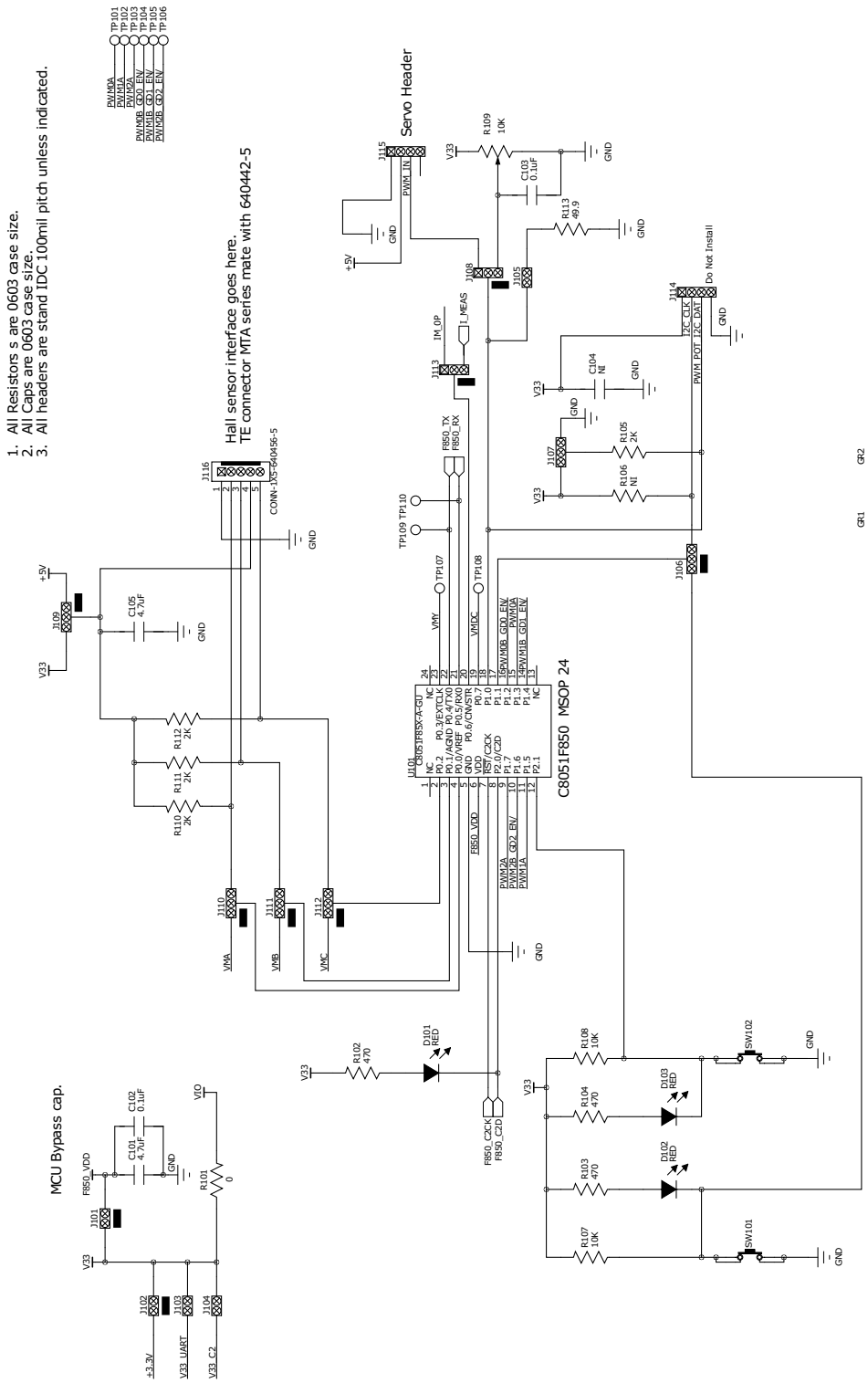


Figure 48. MCU Board Schematic

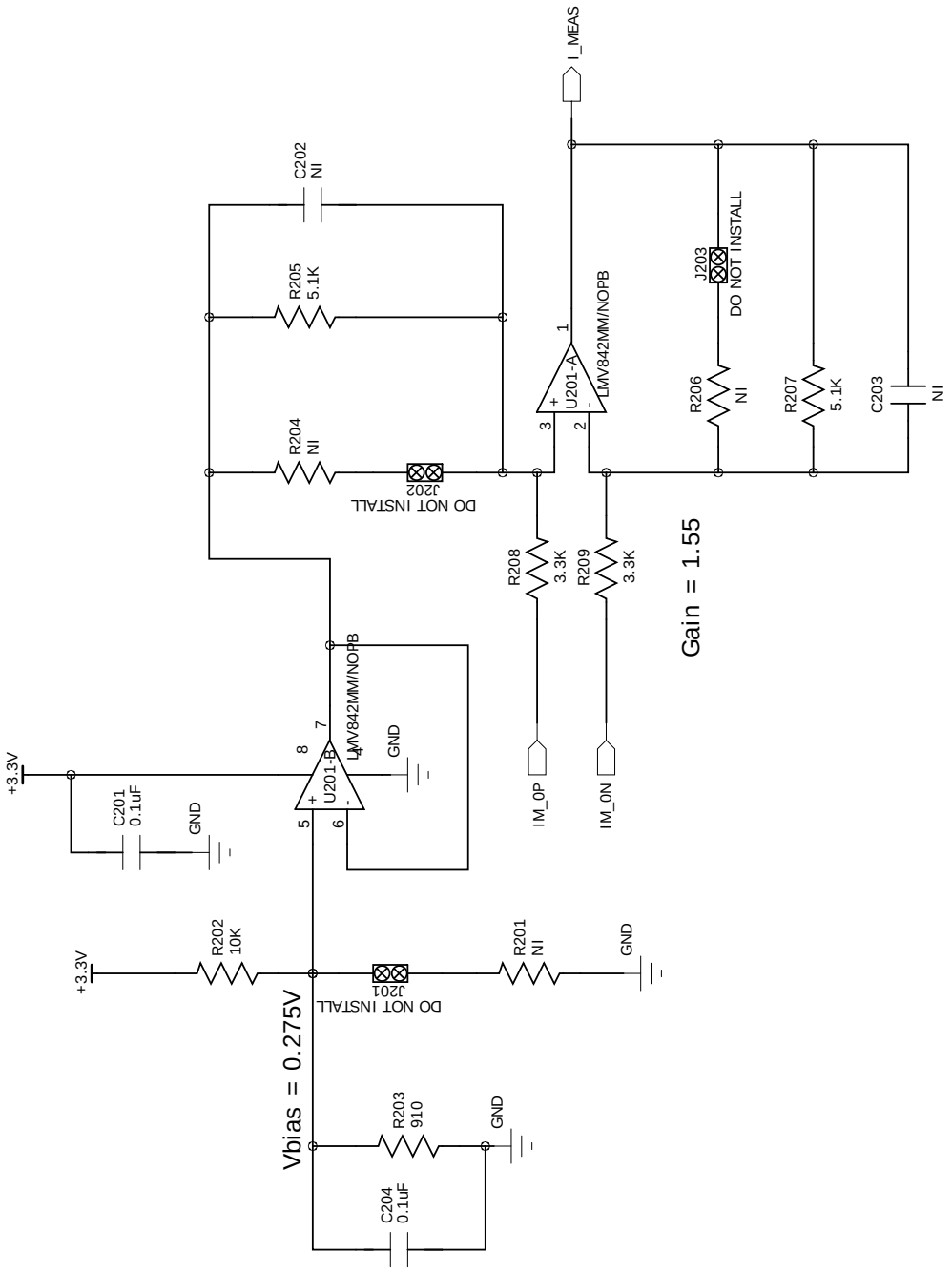


Figure 49. OpAmp Schematic

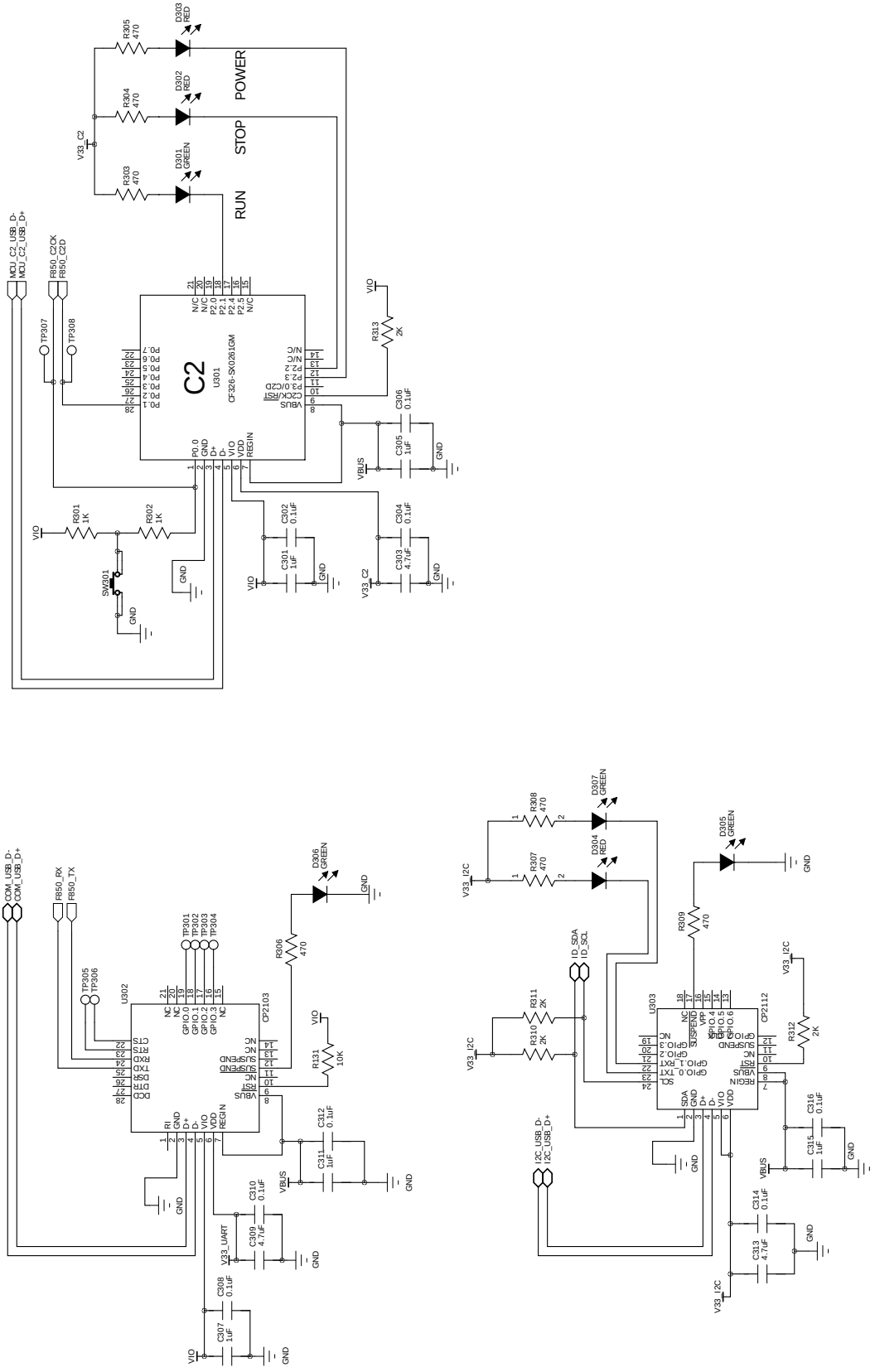


Figure 50. USB Bridge Schematic

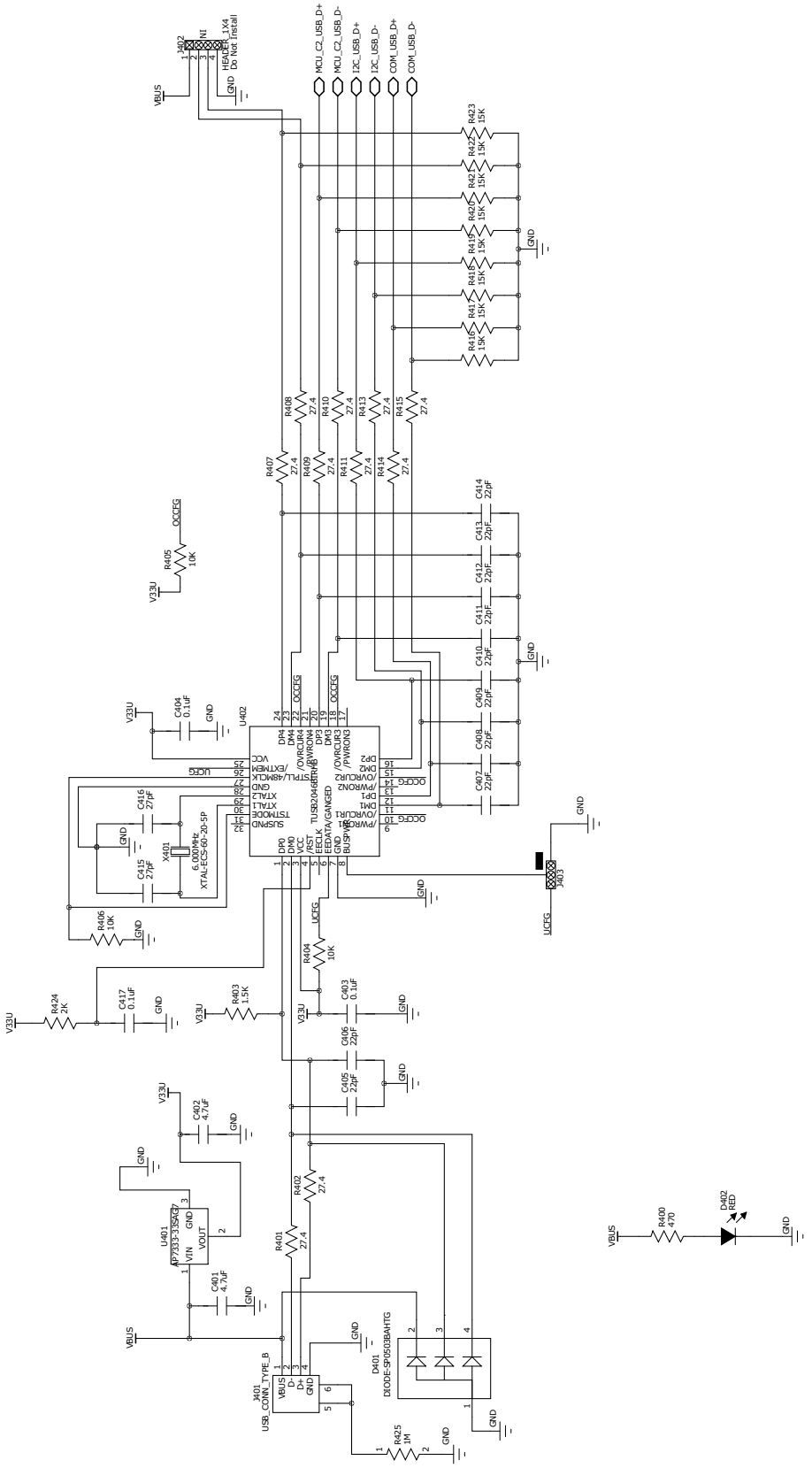
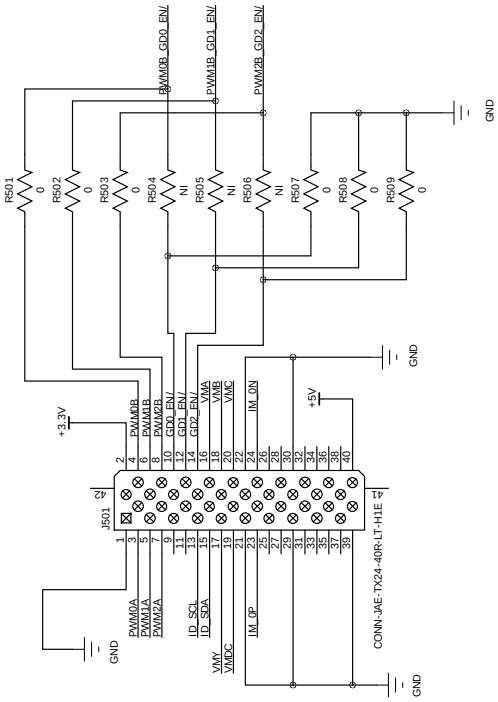
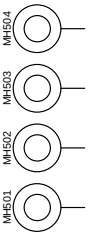


Figure 51. USB Hub Schematic





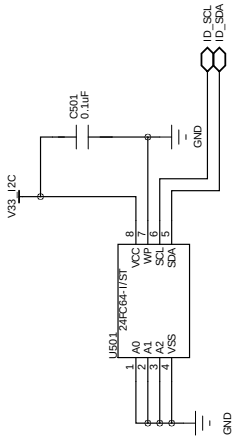


Figure 52. Connector Schematic

8.3. Low-Cost BLDC Motor Design Sample Schematic

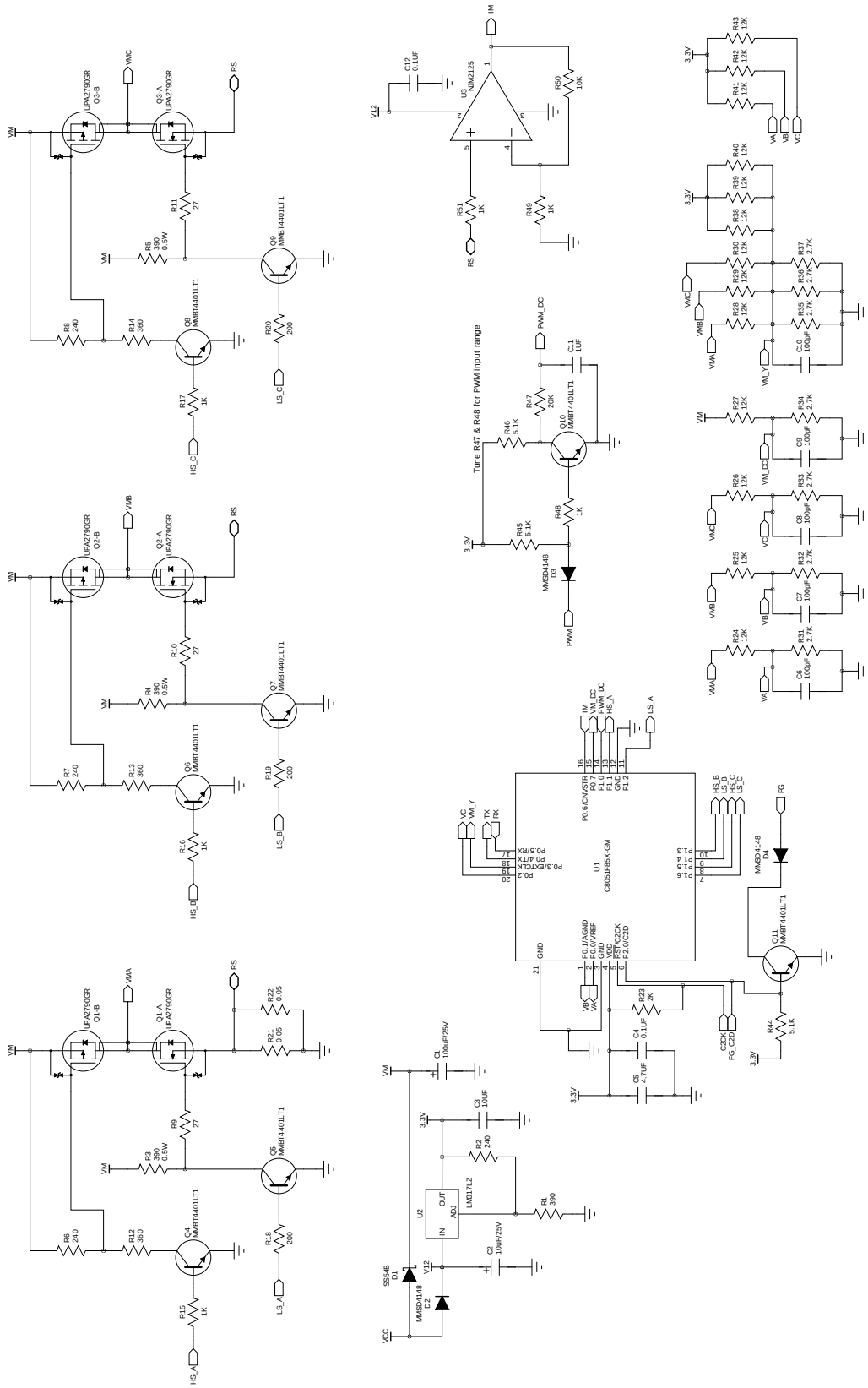


Figure 53. Low-Cost BLDC Motor Design Sample Schematic

9. Bill of Materials

9.1. C8051F850 Powertrain Bill of Materials

Table 15. Bill of Materials—C8051F850 Powertrain

Reference	Part Number	Source	Description
C1, C2	ECOS1VA152BA	Panasonic	1500 µF, 1.27 A, 35 V ±20% Alum_Elec
C10, C11, C12, C13, C14	C0603C0G500-101M	Venkel	100 pF, 50 V ±20% C0G
C15, C16, C17, C18, C19, C20, C21, C22, C23, C27	C1206X7R500-105K	Venkel	1 µF, 50 V ±10% X7R
C24	EEUFC1J681	Panasonic	680 µF, 2.16 A 63 V ±20% Alum_Elec
C26	EEUFC1E221	Panasonic	220 µF, 555 mA, 25 V ±20% Alum_Elec
C28, C29, C32	C1210X7R250-106M	Venkel	10 µF, 25 V ±20% X7R
C3, C4, C5, C6	C5750X7R2A475K230KA	TDK	4.7 µF, 100 V ±10% X7R
C30	C0603X7R500-472K	Venkel	4.7 nF, 50 V ±10% X7R
C31	C0603X7R500-104K	Venkel	0.1 µF, 50 V ±10% X7R
C7, C8, C9, C25	C1210X7R101-105K	Venkel	1 µF, 100 V ±10% X7R
D1, D2, D3, D5	MBR1H100SF	On Semi	1 A, 100 V Schottky Diode
D4	MBRS3100T3	On Semi	3 A, 100 V Schottky Diode
D6	LTST-C190TBKT	Lite-On Technology Corp.	Blue LED
D7	LTST-C190KGKT	Lite-On Technology Corp.	Green LED
D8	SML-LX0603SYW	Lumex Inc.	Yellow 30 mA LED
J1, J2	S1911-46R	Harwin Inc.	SMT Jumper Link
J4	RAPC722X	Switchcraft Inc.	Power Jack 5A BARREL
J5	1714955	Phoenix Contact	2 x 6.35 mm, 32 A, 300 V Terminal Block
J6	1714968	Phoenix Contact	3 x 6.35 mm, 32 A, 300 V Terminal Block
J7	TX25-40P-LT-H1E	JAE	2x20 1.27 mm Receptacle
L1	P0250.473NL	Pulse Electronics	47 µH, 2.6 A ±15% Unshielded
L2	P0770.103NL	Pulse Electronics	10 µH, 1.1 A ±20% Unshielded
LBL1	OL5400SP-F850 24 V, 10 A	OnlineLabels.com	F850 24 V, 10 A Label

Table 15. Bill of Materials—C8051F850 Powertrain (Continued)

Reference	Part Number	Source	Description
PCB1	MCRD-PWR-NLV REV 2.0	Silicon Labs	MCRD-PWR-NLV REV 2.0 BARE PCB
Q1, Q2, Q3, Q4, Q5, Q6	IRFH7446TR2PBF	IR	3.3 mΩ, 40 V N-CHNL MOSFET
R1	WSR5R0500FEA	VishayDale	0.05 Ω, 5 W ±1% Metal
R10, R11, R12, R13, R14, R15, R16, R19, R22, R52, R53, R54, R58	CR0603-16W-000	Venkel	0 Ω, 1/16 W ThickFilm
R23, R24, R25, R26, R31, R32, R33	CR0603-16W-9761F	Venkel	9.76 kΩ, 1/16 W ±1% ThickFilm
R27, R28, R29, R30, R34, R35, R36	CR0603-10W-1151F	Venkel	1.15 kΩ, 1/10 W ±1% ThickFilm
R37, R38, R39, R40, R41, R42	CR0603-10W-1R00F	Venkel	1.0 Ω, 1/10 W ±1% ThickFilm
R4, R5, R6, R7, R8, R9	CR0603-16W-8R2F	Venkel	8.2 Ω, 1/16 W ±1% ThickFilm
R43, R44, R45, R60, R61	CR0603-16W-1002F	Venkel	10 kΩ, 1/16 W ±1% ThickFilm
R46, R47, R48, R49, R50, R51	CR0603-10W-4701F	Venkel	4.7 kΩ, 1/10 W ±1% ThickFilm
R62	CR0603-10W-2002F	Venkel	20 kΩ, 1/10 W ±1% ThickFilm
R63, R65	CR0603-16W-8661F	Venkel	8.66 kΩ, 1/16 W ±1% ThickFilm
R64	CR0603-16W-1001F	Venkel	1 kΩ, 1/16 W ±1% ThickFilm
R66	CR0603-16W-3011F	Venkel	3.01 kΩ, 1/16 W ±1% ThickFilm
R67	CR0603-10W-1331F	Venkel	1.33 kΩ, 1/10 W ±1% ThickFilm
R68, R69, R70, R72, R73, R74	CR0603-10W-5111F	Venkel	5.11 kΩ, 1/10 W ±1% ThickFilm
U1, U2, U3	Si8230BB-B-IS	Silicon Labs	2500 Vrms, 24 V ISOdriver
U4	LM2592HVS-5.0	TI	2 A, 5 V Step-Down Voltage Regulator
U5	LM2698MM-ADJ/NOPB	TI	1.35 A Boost Regulator
U6	LM2937IMP-3.3	National Semiconductor	500 mA Low Dropout Regulator
U7	24FC64F-I/ST	Microchip	64 kbit I^2C Serial EEPROM

Table 15. Bill of Materials—C8051F850 Powertrain (Continued)

Reference	Part Number	Source	Description
Components Not Installed			
HW1, HW2, HW3, HW4	PMS 440 0025 SL	B&F Fastener Supply	Screw
HW5, HW6, HW7, HW8	2204	Keystone Electronics	Standoff
HW9, HW10, HW11, HW12	3358	Keystone Electronics	Flat Washer #4
J3	S1911-46R	Harwin Inc.	SMT Jumper Link
J8, J9	TSW-102-07-L-S	Samtec	Conn Header 2POS
R17, R18, R20, R21	CR0603-16W-000	Venkel	0 Ω, 1/16 W ThickFilm
R2, R3	WSR5R0100FEA	VishayDale	10 mΩ 5 W ±1% Metal
R55, R56, R57	CR0603-10W-2002F	Venkel	20 kΩ, 1/10 W ±1% ThickFilm
R59	CR0603-16W-1002F	Venkel	10 kΩ, 1/16 W ±1% ThickFilm
R71	CR0603-10W-5111F	Venkel	5.11 kΩ, 1/10 W ±1% ThickFilm
TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13, TP14, TP15, TP16, TP17, TP18, TP19, TP20, TP21, TP22, TP23, TP24, TP25, TP26, TP27, TP28, TP29, TP30, TP31, TP32, TP33, TP34, TP35, TP36, TP37, TP38, TP39, TP40, TP41	151-203-RC	Kobiconn	Test Points

9.2. C8051F850 MCU Board Bill of Materials

Table 16. Bill of Materials—C8051F850 MCU Board

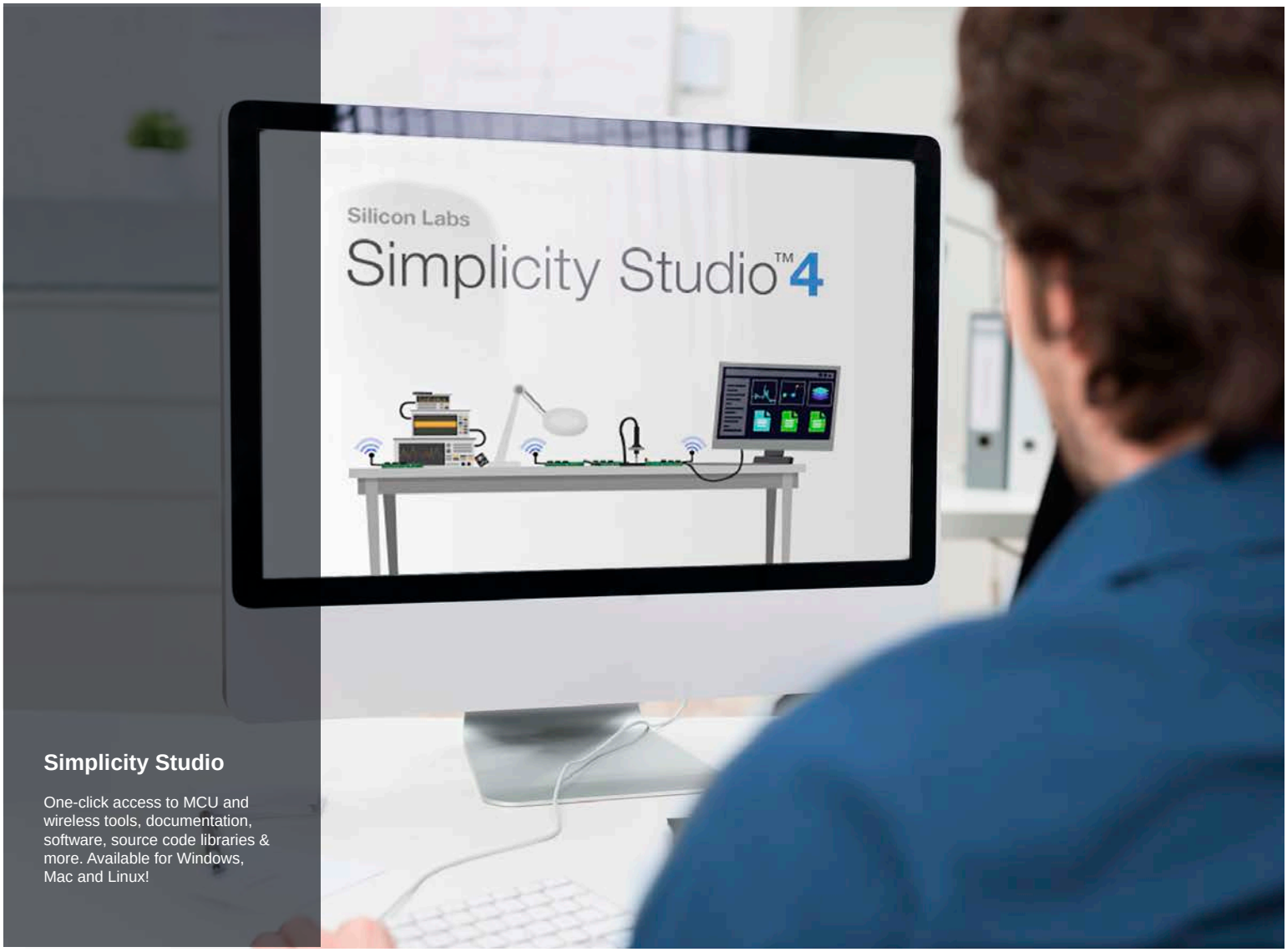
Reference	Part Number	Source	Description
U501	24FC64-I/ST	Microchip	IC EEPROM I2C 64 kbit (8Kx8bit) 1 Mbps
U401	AP7333-33SAG7,3.3V	Diodes Inc	300 mA, 3.3 V LDO Linear Regulator
U101	C8051F850-C-GU	Silicon Labs	C8051F850-C-GU
C102, C103, C201, C204, C302, C304, C306, C308, C310, C312, C314, C316, C403, C404, C417, C501	CAP-0.1UF-50V-Y7R-0603, 0.1 µF, 10–80%	Murata	CAP-CER-0.1 µF, 50 V, 10%-Y7R-0603
C301, C305, C307, C311, C315	CAP-1UF-25V-X5R-0603, 1 µF,10%	Murata	CAP-CER-1UF-25V-10%-X5R-0603
C405, C406, C407, C408, C409, C410, C411, C412, C413, C414	CAP-22PF-50V-COG-0603, 22 pF,10%	Venkel	CAP-CER-22 pF, 50 V, 10%-C0G-0603
C415, C416	CAP-27PF-250V-COG-0603, 27 pF,5%	Murata	CAP-CER-27 pF, 250 V, 5%-C0G-0603
C101, C105, C303, C309, C313, C401, C402	CAP-4.7UF-10V-Y5R-0603, 4.7 µF, 10–80%	Murata	CAP-CER, 4.7 µF, 10 V, 10%-Y5R-0603
C104, C202, C203	CAP-NI-XXX-XXX-0603, NI, 10–80%		CAP-NI-0603
U301	CF326-SX0261GM,C8051F326		
J116	CONN-1X5-640456-5	TE	CONN-1x5-640456-5
J501	CONN-JAE-TX24-40R-LT-H1E	JAE	JAE - TX24-40R-LT-H1E
U302	CP2103		
U303	CP2112		
D401	DIODE-SP0503-BAHTG,SP0503BAHTG		
GR1, GR2	GND_RAIL,WIRE LOOP		
J101, J102, J103, J104, J105	HEADER_1X2		1 x 2 100 mil Header

Table 16. Bill of Materials—C8051F850 MCU Board (Continued)

Reference	Part Number	Source	Description
J106, J107, J108, J109, J110, J111, J112, J113, J403	HEADER_1X3		1 x 3 100 mil Header
J115	HEADER_1X4		1 x 4 100 mil Header
U201	LMV842MM/NOPB	TI	IC OpAmp Dual RRIO MSOP8
MH501, MH502, MH503, MH504	MH-125		Mechanical Hole 125
D301, D305, D306, D307	OPTO-LED-GREEN-CLR-0603, GREEN	Lite-On	OPTO-LED-GREEN-CLR-0603
D101, D102, D103, D302, D303, D304, D402	OPTO-LED-RED-CLR-0603, RED	Lite-On	OPTO-LED-RED-CLR-0603
R109	POT-10K-THUMBWHEEL-LIN-EAR, 10 kΩ, 20%	Alpha (Taiwan)	POT-10 kΩ, Thumbwheel Linear 30 mW, 20% RoHS
R101, R501, R502, R503, R507, R508, R509	RES-0-1%-0.1W-0603, 0 k,1%	Yageo	RES-0 Ω, 1/10 W, 1%, 0603
R403	RES-1.5K-1%-0.1W-0603, 1.5 kΩ, 1%	Venkel	RES-1.5 kΩ, 1/10W, 1%, 0603
R107, R108, R131, R202, R404, R405, R406	RES-10K-1%-0.1W-0603, 10 kΩ, 1%	Yageo	RES-10 kΩ, 1/10W, 1%, 0603
R416, R417, R418, R419, R420, R421, R422, R423	RES-15K-1%-0.1W-0603, 15 kΩ, 1%	Yageo	RES-15 kΩ, 1/10W, 1%, 0603
R301, R302	RES-1K-1%-0.1W-0603, 1 kΩ, 1%	Panasonic	RES-1 kΩ, 1/10 W, 1%, 0603
R425	RES-1M-1%-0.1W-0603, 1 MΩ, 1%	Venkel	RES-1 MΩ, 1/10 W, 1%, 0603
R401, R402, R407, R408, R409, R410, R411, R413, R414, R415	RES-27.4-1%-0.1W-0603, 27.4 Ω,1%	Venkel	RES-27.4 Ω, 1/10 W, 1%, 0603
R105, R110, R112, R310, R311, R312, R313, R424	RES-2K-1%-0.1W-0603, 2 kΩ,1%	Yageo	RES-2 kΩ, 1/10 W, 1%, 0603
R208, R209	RES-3.3K-1%-0.1 W-0603, 3.3 kΩ,1%	Yageo	RES-3.3 kΩ, 1/10 W, 1%, 0603

Table 16. Bill of Materials—C8051F850 MCU Board (Continued)

Reference	Part Number	Source	Description
R102, R103, R104, R303, R304, R305, R306, R307, R308, R309, R400	RES-470-1%-0.1 W-0603, 470 Ω,1%	Yageo	RES-470 Ω, 1/10 W-1%, 0603
R113	RES-49.9-1%-0.1 W-0603, 49.9 Ω,1%	Yageo	RES-49.9 Ω, 1/10 W, 1%-0603
R205, R207	RES-5.1K-1%-0.1 W-0603, 5.1 kΩ,1%	Yageo	RES-5.1 kΩ, 1/10 W, 1%-0603
R203	RES-910-1%-0.1 W-0603, 910 Ω,1%	Yageo	RES-910 Ω, 1/10 W, 1%-0603
SW101, SW102, SW301	SW-PUSH-BUTTON-6X6		SW-Push-Button-6x6mm
U402	TUSB2046BIRHB	TI	4 Port Full-Speed USB Hub,QFN
J401	USB_CONN_TYPE_B,TYPE B		
X401	XTAL-ECS-60-20-5P, 6.000 MHz, 30 ppm	ECS	HC49 SMD 6.000 MHZ Crystal
Components Not Installed			
J201, J202, J203	HEADER_1X2		1 x 2 100 mill Header, NI
J114, J402	HEADER_1X4		1 x 4 100 mill Header
R106, R201, R204, R206, R504, R506	RES-NI-1%-0.1W-0603,NI		RES-NOT-INSTALL-0603
TP101, TP102, TP103, TP104, TP105, TP106, TP107, TP108, TP109, TP110, TP301, TP302, TP303, TP304, TP305, TP306, TP307, TP308	TESTPOINT_WHITE,White	Keystone Electronics	Test Point - White



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 and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Silicon Laboratories:

C8051F850-BLDC-RD