C8051F850 BLDC REFERENCE DESIGN KIT

Contents
1 Introduction .................................................................................................................. 4
  1.1 Kit Contents ........................................................................................................... 4
2 Theory of Operation ..................................................................................................... 4
  2.1 System Model ........................................................................................................ 4
  2.2 Driving 3-phase BLDC Motors .............................................................................. 6
    2.2.1 High-side PWM ............................................................................................... 7
    2.2.2 Low-side PWM .............................................................................................. 10
    2.2.3 Complementary high-side PWM ................................................................... 10
    2.2.4 Complementary low-side PWM ..................................................................... 11
    2.2.5 Mixed mode PWM ......................................................................................... 11
  2.3 Determining commutation instant ......................................................................... 12
    2.3.1 High-side PWM – Part 1 of PWM off cycle .................................................. 13
    2.3.2 High-side PWM – Part 2 of PWM off cycle .................................................. 15
    2.3.3 Low-side PWM – Part 1 of PWM off cycle .................................................... 15
    2.3.4 Low-side PWM – Part 2 of PWM off cycle .................................................... 16
    2.3.5 Summary of terminal voltage comparisons for ZCP detection ....................... 16
3 System Implementation ............................................................................................... 16
  3.1 Powertrain board .................................................................................................. 16
    3.1.1 Attenuated Motor Voltage Circuit ................................................................. 17
    3.1.2 Back-EMF Filter Circuit ................................................................................ 17
    3.1.3 Virtual neutral Filter Circuit ......................................................................... 18
  3.2 MCU Board .......................................................................................................... 18
    3.2.1 Gate Drive pin connections .......................................................................... 19
    3.2.2 Motor sensor connections ............................................................................ 19
    3.2.3 Current sensing circuit ................................................................................. 20
  3.3 Back-EMF Zero Crossing Point detection technique .............................................. 20
  3.4 PWM and Comparator Configuration Sequences ................................................. 22
  3.5 BLDC Motor startup technique ............................................................................ 24
  3.6 Hyperdrive mode .................................................................................................. 25
4 Firmware .................................................................................................................... 25
7.3 Low-cost BLDC motor design sample schematic...47
1 Introduction

This design kit provides a complete system-level solution for sensorless brushless DC (BLDC) motors. This document includes complete schematics, PCB layout and firmware.

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

- Trapezoidal back-EMF
- Up to 24V DC voltage
- 24 kHz PWM frequency
- Maximum average current of 10A
- Maximum speed of 200000 RPM or lower for a 2-pole BLDC motor
- Overcurrent detection capability – stops motor when average current exceeds 10A
- Motor Stall detect capability – stops motor when detecting motor has stalled or under extreme load.
- Tachometer Frequency Generator (FG) output signal

The kit aims to demonstrate the capabilities of the C8051F850 for operating BLDC motor. 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:

1 MCU Board: MCRD-MCU-C8051F850 with the motor control firmware pre-programmed into the MCU
1 Powertrain Board: MCRD-PWR-NLV-F85X
1 BLDC Motor: Turnigy 450 Series 3800KV Brushless Outrunner Helicopter Motor
1 Motor Mount Board
1 CD containing the Motor Control Interface Utility – this is a GUI PC Application to aid user in firmware configuration and operation. [TBD: CD may not be available in BETA version]

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.
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. $R_x$ 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 as:

\[
V_A - V_N = I_A R + L \frac{dI_A}{dt} + e_A \tag{1}
\]
\[
V_B - V_N = I_B R + L \frac{dI_B}{dt} + e_B \tag{2}
\]
\[
V_C - V_N = I_C R + L \frac{dI_C}{dt} + e_C \tag{3}
\]
\[
e_A = K \omega_m F(\theta_e) \tag{4}
\]
\[
e_B = K \omega_m F(\theta_e - \frac{2\pi}{3}) \tag{5}
\]
\[
e_C = K \omega_m F(\theta_e - \frac{4\pi}{3}) \tag{6}
\]
\[
\omega_m = \frac{2}{N_p} \frac{d\theta_e}{dt} \tag{7}
\]

where

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

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

$e_A, e_B, e_C$ denote the phase back-EMF (BEMF) associated with terminals A, B, 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
$\theta_i$ is the electrical angle of the motor
$\omega_m$ is the angular speed of the motor
$F(\theta_e)$ represents the BEMF reference waveform as a function of rotor electrical angle

The electromagnetic torque equation is given by:

$$T_e = K I_A F(\theta_e) + K I_B F(\theta_e - \frac{2\pi}{3}) + K I_C F(\theta_e - \frac{4\pi}{3})$$  \hspace{1cm} (8)

BLDC and PMSM (permanent magnet synchronous motor) motors are differentiated by the BEMF reference waveform $F(\theta_e)$. BLDC motors are identified by their trapezoidal BEMF reference waveforms on each phase as given below:

![Figure 2 BEMF reference waveforms of all phases](image)

### 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 follows:
The conducting interval of each phase is $120^\circ$ and only 2 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

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 do not receive sufficient heat dissipation. The sequence of current flow through the inverter circuit is illustrated in the following 3 diagrams:

![High-side PWM scheme diagram](image-url)
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 re-circulation current remains flowing:

![Diagram of current flow: High-side PWM, part 2 of PWM off cycle](image)

Figure 6  Current flow: High-side PWM, part 2 of PWM off cycle

And if the PWM off cycle is long enough, the remaining re-circulation current will decay to zero. The oscilloscope capture below shows the voltage of an open terminal for all the phases of the PWM off cycle:

![Oscilloscope capture of open terminal](image)

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

![Low-side PWM scheme](image)

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 will not be suitable. As in the high-side PWM scheme, low-side PWM is simple to implement as it requires only 1 PWM signal to be active at any time. And like the high-side PWM, the disadvantage is the potential heat problems due to the re-circulation current.

### 2.2.3 Complementary high-side PWM

![Complementary High-side PWM](image)

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 2 complementary PWM signals with dead time between the enable periods of the 2 signals. Another disadvantage is that this method is not suitable for applications where the load is light – this causes the re-
circulation 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. The braking effect lowers the efficiency of the motor.

2.2.4 Complementary low-side PWM

![Figure 10: Complementary Low-side PWM](image)

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](image)

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 30° 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 ($R_p$) is typically used to construct a virtual neutral point:

![BLDC motor drive circuit with virtual neutral circuit](image)

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

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

During commutation phase 1 ($0^\circ$ to $60^\circ$), 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:
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:

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

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

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:

![Commutation Phase 1 – High-side PWM Off majority current flow](image)

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:
If the minimum forward diode voltage required for current to flow is $V_{F_{\text{min}}}$, then current will flow in the open phase if the following condition is met:

\[
V_{Ac} < -V_{F_{\text{min}}}
\]

\[
e_A < \frac{V_{FCL}}{2} - V_{F_{\text{min}}}
\]

The current flow is illustrated below 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 $0^0$ to $30^0$ 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:

\[
V_A = -V_{F_{\text{AL}}} \leq -V_{F_{\text{min}}}
\]

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

\[
V_Y = -\frac{1}{3}(V_{F_{\text{AL}}} + V_{FCL})
\]

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

\[
V_Y - V_A = -\frac{1}{3}(-2 \cdot V_{F_{\text{AL}}} + V_{FCL})
\]

This is a small valued current that flows in the open terminal for a short while until it becomes zero, or when the PWM returns to the PWM on cycle – whichever occurs first. Hence, when current flows in the open terminal, it is impossible to measure the zero-crossing point. And this problem can be further exacerbated parasitic capacitance across the drain-source of the power MOSFETs that 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:

\[
V_A = V_{Ac} = e_A - V_{FCL}/2 \quad (25)
\]
\[
V_Y = \frac{1}{3}(e_A - 3 \times V_{FCL}/2) \quad (26)
\]
The comparison relationship is still the same as when PWM is on:

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

2.3.3 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 below:

![Diagram of Low-side PWM - PWM Off with current flow in open terminal](image)

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:

\[
V_C = V_{DC} \quad (27)
\]
\[
V_B = V_{DC} + V_{FCH} \quad (28)
\]
\[
V_N = V_{DC} + V_{FCH}/2 \quad (29)
\]
\[
V_{Ac} = V_{N} + e_A = e_A + V_{DC} + V_{FCH}/2 \quad (30)
\]

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

\[
V_{Ac} > V_{DC} + V_{Fmin}
\]
\[
e_A > V_{Fmin} - \frac{V_{FCH}}{2} \quad (31)
\]
And the terminal voltage 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:
\[
V_Y - V_A = \frac{1}{3} (-2 \cdot V_{FAH} + V_{FCH}) \quad (34)
\]

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:
\[
V_A = V_{ae} = e_A + V_{DC} + \frac{V_{FCH}}{2} \quad (35)
\]
\[
V_Y = V_{DC} + \frac{1}{3} (e_A + 3 \cdot \frac{V_{FCH}}{2}) \quad (36)
\]
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

This table presents a summary of the terminal voltage comparison with the virtual neutral point voltage:

<table>
<thead>
<tr>
<th>Active PWM state</th>
<th>VY-VA equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM on</td>
<td>(V_Y - V_A = -\frac{2}{3} e_A) (16)</td>
</tr>
<tr>
<td>High-side/Low-side PWM off, zero current flows in open terminal A</td>
<td>(V_Y - V_A = -\frac{2}{3} e_A) (16)</td>
</tr>
<tr>
<td>High-side PWM off, non-zero current flows in open terminal A</td>
<td>(V_Y - V_A = -\frac{1}{3} (-2 \cdot V_{FAL} + V_{FCH})) Conditional on: (e_A &lt; \frac{V_{FCH}}{2} - V_{Fmin}) (21)</td>
</tr>
<tr>
<td>Low-side PWM off, non-zero current flows in open terminal A</td>
<td>(V_Y - V_A = \frac{1}{3} (-2 \cdot V_{FAH} + V_{FCH})) Conditional on: (e_A &gt; V_{Fmin} - \frac{V_{FCH}}{2}) (31)</td>
</tr>
</tbody>
</table>

Table 2 Terminal voltage comparison equations

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. The schematics for the design kit can be found in Appendix. The kit consists of 2 boards - an MCU board and a powertrain board.

3.1 Powertrain board

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

- Motor supply voltage range of 7V to 24V
- 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 3.3V required by the MCU board
- 50 mOhm current sensing resistor rated for 10W
- 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:

![Attenuated Motor Voltage Circuit Diagram]

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:

![Back-EMF Filter Circuit Diagram]
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:

\[ V_{VMA} = \frac{+3.3/R68}{R23} + \frac{V_{MTR,VA}/R23}{R25} \]  

(37)

This works out to an offset voltage of 0.55V and a gain of 0.0877. C10 is a 100pF capacitor that filters sharp 
ringing in the voltages caused by parasitic capacitance between the drain and source of the power MOSFETs.

3.1.3 Virtual neutral Filter Circuit

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

![Virtual neutral filter circuit](image)

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

\[ V_{VMY} = \frac{+3.3/R72}{R31} + \frac{(V_{MTR,VA}+V_{MTR,VB}+V_{MTR,VC})/(3+R31)}{R34} \]  

(38)

Recall that equation 9 is:

\[ V_Y = \frac{1}{3}(V_A + V_B + V_C) \]  

(9)

This shows that the voltage at VMY is the virtual neutral voltage that has scaled and offset in the same manner as 
VMA. Hence, the VMx signals from the resistor network circuits can be used for back-EMF zero crossing detection 
by the MCU’s comparator. C14 is a 100pF capacitor that filters sharp ringing in the voltages caused by parasitic 
capacitance between the drain and source of the power MOSFETs.

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 – but this will be explained in section 3.3 (Back-EMF Zero Crossing Point detection technique) later.

### 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 the table below:

<table>
<thead>
<tr>
<th>MCU Pin</th>
<th>Connection name</th>
<th>Controlled motor phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.2</td>
<td>PWM0B_GD0_EN</td>
<td>A</td>
</tr>
<tr>
<td>P1.3</td>
<td>PWM0A</td>
<td>A</td>
</tr>
<tr>
<td>P1.4</td>
<td>PWM1B_GD1_EN</td>
<td>B</td>
</tr>
<tr>
<td>P1.5</td>
<td>PWM1A</td>
<td>B</td>
</tr>
<tr>
<td>P1.6</td>
<td>PWM2B_GD2_EN</td>
<td>C</td>
</tr>
<tr>
<td>P1.7</td>
<td>PWM2A</td>
<td>C</td>
</tr>
</tbody>
</table>

**Table 3**  
MCU Motor Gate Drive Pin Connection

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

![Figure 19](image)

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 (refer to 3.1.2). 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 the 3.1.3. 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.

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 2, 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 the diagram below:

![Diagram](image)

**Figure 21 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 the diagram below:

![Diagram](image)

**Figure 22 Peripheral configuration for ZCP detection of falling BEMF signal**

Recall from Table 2 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. Hence, 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 17 and Figure 18.
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 Figure 21 and Figure 22.
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 the diagram:

![Diagram of CEX1 and CEX0 synchronization for low duty cycle](image1.png)

**Figure 23**  
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.

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 the diagram:

![Diagram of CEX1 and CEX0 synchronization for high duty cycle](image2.png)

**Figure 24**  
Active high motor PWM - tracking synchronization for high duty cycle

The advantage of this technique is that it incurs MCU processing overhead to ignore the undesirable parts of the BEMF signal. Otherwise, without the tracking signal, 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. The next 3 tables specifies the peripheral assignment sequences for the 3 different PWM modes in a specific direction.
<table>
<thead>
<tr>
<th>State</th>
<th>AH</th>
<th>BH</th>
<th>CH</th>
<th>AL</th>
<th>BL</th>
<th>CL</th>
<th>Open phase</th>
<th>Slope</th>
<th>CP0+</th>
<th>CP0- and CEX0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>C</td>
<td>\</td>
<td>VMC</td>
<td>VMY</td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>A</td>
<td>/</td>
<td>VMY</td>
<td>VMA</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>B</td>
<td>\</td>
<td>VMB</td>
<td>VMY</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>/</td>
<td>VMY</td>
<td>VMC</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>A</td>
<td>\</td>
<td>VMA</td>
<td>VMY</td>
</tr>
<tr>
<td>5</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>B</td>
<td>/</td>
<td>VMY</td>
<td>VMB</td>
</tr>
</tbody>
</table>

Table 4  High-side PWM peripheral configuration sequence

<table>
<thead>
<tr>
<th>State</th>
<th>AH</th>
<th>BH</th>
<th>CH</th>
<th>AL</th>
<th>BL</th>
<th>CL</th>
<th>Open phase</th>
<th>Slope</th>
<th>CP0+</th>
<th>CP0- and CEX0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>C</td>
<td>\</td>
<td>VMC</td>
<td>VMY</td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>A</td>
<td>/</td>
<td>VMY</td>
<td>VMA</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>B</td>
<td>\</td>
<td>VMB</td>
<td>VMY</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>/</td>
<td>VMY</td>
<td>VMC</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>A</td>
<td>\</td>
<td>VMA</td>
<td>VMY</td>
<td>VMY</td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>B</td>
<td>/</td>
<td>VMY</td>
<td>VMB</td>
<td>VMB</td>
</tr>
</tbody>
</table>

Table 5  Low-side PWM peripheral configuration sequence

<table>
<thead>
<tr>
<th>State</th>
<th>AH</th>
<th>BH</th>
<th>CH</th>
<th>AL</th>
<th>BL</th>
<th>CL</th>
<th>Open phase</th>
<th>Slope</th>
<th>CP0+</th>
<th>CP0- and CEX0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>C</td>
<td>\</td>
<td>VMC</td>
<td>VMY</td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>A</td>
<td>/</td>
<td>VMY</td>
<td>VMA</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>B</td>
<td>\</td>
<td>VMB</td>
<td>VMY</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>PWM</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>/</td>
<td>VMY</td>
<td>VMC</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>PWM</td>
<td>A</td>
<td>\</td>
<td>VMA</td>
<td>VMY</td>
<td>VMY</td>
</tr>
<tr>
<td>5</td>
<td>PWM</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>B</td>
<td>/</td>
<td>VMY</td>
<td>VMB</td>
</tr>
</tbody>
</table>

Table 6  Mixed mode PWM peripheral configuration sequence

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 pre-determine 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 pre-determined level regardless of the motor load or motor supply voltage. During motor startup, the comparator and PWM are configured as shown in the following diagram:

![Diagram showing 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 1.8V. 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 20. The gain of the op-amp = R207/R209. R208 & R209 should always have the same value; R205 & 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. We use 20A as the trigger limit in our design because the current sense resistor is rated at 5W. 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:
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:

$$T_{k+1} = \left( \frac{1}{2} \sum_{i=0}^{k} T_i \right) \cdot \sqrt{\frac{1 + \frac{4\pi}{\Sigma_{i=0}^{k} T_i}}{\Sigma_{i=0}^{k} T_i} - 1}$$  \hspace{1cm} (39)

The first 3 terms can be worked out offline 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:

$$T_{k+1} \approx \frac{\frac{1}{2} \sum_{i=0}^{k} T_i}{\Sigma_{i=0}^{k} T_i}$$  \hspace{1cm} (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:

$$T_e = K_I A F(\theta_e) + K_I B F(\theta_e - \frac{2\pi}{3}) + K_I C F(\theta_e - \frac{4\pi}{3})$$  \hspace{1cm} (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 especially applicable for motors where the maximum speed is significantly below the maximum speed supported by the firmware (200000 RPM for 2-pole motor).

### 4 Firmware

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. The block diagram below shows a high-level overview of this organization.
This framework simplifies the organization and implementation of user-specific application code. Users can write application code to access to 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](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>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Files</th>
</tr>
</thead>
</table>
| User Application       | This is the user application code. Users should reimplement the code here for their own applications. The default firmware is complex because it communicates with the Motor Control Interface Utility 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  
blcdck.h  
btn.h  
MCP_Core.h  
MCP_Registers.h  
mtrapp.h  
UART_Driver.h |
| Standard C8051F85x     | Standard header files to access C8051F85x SFRs in a compiler-independent manner.                                                                                                                                 | C8051F850_defs.h                                                        |
Table 7 Categorization of BLDC reference design firmware

4.1 Motor Control API

The motor control API consists of the following functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL_MTR_init(void)</td>
<td>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.</td>
</tr>
<tr>
<td>SL_MTR_motor(void)</td>
<td>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.</td>
</tr>
<tr>
<td>SL_MTR_start_motor(void)</td>
<td>This function starts the BLDC motor spinning if the motor is in the MOTOR_STOPPED 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.</td>
</tr>
<tr>
<td>SL_MTR_stop_motor(void)</td>
<td>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_STOPPED state.</td>
</tr>
<tr>
<td>U16 SL_MTR_time(void)</td>
<td>Gets the high 16-bit of the 32-bit running time of the system. Each time unit is 2.7ms.</td>
</tr>
<tr>
<td>SL_MTR_GET_32BIT_TIME(x)</td>
<td>A macro to read the 32-bit running time of the system. x is a UU32 type variable.</td>
</tr>
<tr>
<td>SL_MTR_change_pid_gain(pg, ig)</td>
<td>Initializes the PI proportional gain and PI integral gain parameters. The proportional gain is represented in units of 256/(SPEED_UNIT RPM). Refer to section 4.4.9 for definition of SPEED_UNIT. The integral gain has the same units based on a time resolution unit of 1.</td>
</tr>
<tr>
<td>SL_MTR_change_num_poles(poles)</td>
<td>Change the number of poles in the motor. This is normally not used in an actual...</td>
</tr>
</tbody>
</table>
The motor control API also exports 3 read-only variables for the user application. These variables are prefixed by SLR_.

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

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>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLW_target_rpm</td>
<td>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).</td>
</tr>
<tr>
<td>SLW_target_pwm_duty</td>
<td>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</td>
</tr>
<tr>
<td>SLW_acceleration_step_size</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_deceleration_step_size</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_oc_debounce</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_current_limit</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_motor_max_rpm</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_user_direction</td>
<td>This is a 1-bit variable used by the user application to indicate to the motor control state machine the desired direction of rotation.</td>
</tr>
<tr>
<td>Variable name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SLW_rpm_updated</td>
<td>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.</td>
</tr>
<tr>
<td>SLW_pwm_updated</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 10  Motor Control API writable variables

4.2  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 this code sample:

```c
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:
               - monitor motor temperature
               - balance motor windings
            */
        }
    }
}
```
- 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 ...

4.3 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 done by using the Motor Control Interface Utility provided with BLDC kit. Note that some of configurations may require changes to the jumper settings.

4.3.1 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.

4.3.2 Speed Control Command Source

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

- UART-received 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:

HOST_SPEED_SOURCE – An application running on Host PC sets the target value of Speed Control Parameter.
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.
4.3.3 Direction Command Source
The firmware supports two sources of rotation direction command:

- UART-received command
- SW101 button.

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

```c
#define BLDC_RD_DIR_SRC DIRECTION_BY_XXX
```

where DIRECTION_BY_XXX can be one of the following:

- **DIRECTION_BY_HOST** – An application running on Host PC can change the direction of rotation.
- **DIRECTION_BY_BUTTON** – The SW101 button on the BLDC kit will be used to toggle direction of rotation.

Note that configuring for **DIRECTION_BY_BUTTON** requires J106 jumper to be set in the default position (1-2).

Also note that **DIRECTION_BY_HOST** can only be selected if firmware configuration has also enabled both **BUILD_FOR_PROTOCOL** and **BUILD_FOR_UART**.

This configuration is defined in **BLDC_RD_Build_Params.h**.

4.3.4 Build for Protocol
The Motor Control Interface Utility tool is a PC-based GUI tool that can be used with the demo firmware during development. The firmware implements a protocol to interact with PC GUI application. This protocol can be enabled at build-time by defining:

```c
#define BUILD_FOR_PROTOCOL
```

If the firmware is built with this parameter disabled, then the speed control command source (section 4.3.2) and direction command source (section 4.3.3) must not select a UART-received command.

This configuration is defined in **BLDC_RD_Build_Params.h**.

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

```c
#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:

```c
#define IMEA_ADMX 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 Op-Amp can be configured as follows:
#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.

### 4.3.6 Motor Stall Detection

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

```c
#define FEATURE_RPMSTALL_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:

```c
#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 the value of ‘1UL’ in COMPENSATION_CONSTANT_FACTOR or increase the value of ‘70UL’ in DELTA_CURRENT_FACTOR_K.

### 4.3.7 Motor Voltage Measurement

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

```c
#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:

```c
#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.

### 4.3.8 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:

```c
#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:

```c
#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.
4.3.9 Motor Specific Configurations

In addition to BLDC_RD_NUM_POLES, following configurations depend on motor selected by user:

```plaintext
#define MOTOR_KV 3800
#define MOTOR_MAX_CURRENT 10.0
#define MOTOR_SUPPLY_VDD 8.5
#define MOTOR_MAX_RPM ((U16)((U32)MOTOR_KV * MOTOR_SUPPLY_VDD * 1.2 / SPEED_UNIT))
```

where MOTOR_KV, MOTOR_MAX_CURRENT and MOTOR_SUPPLY_VDD are appropriate values given in the motor specifications and system design. Alternatively, users can replace these with the minimal set of MOTOR_MAX_CURRENT and MOTOR_MAX_RPM definitions. Users should define additional headroom of 10% to 20% for the MOTOR_MAX_RPM definition – this allows an opportunity for hyperdrive to kick in if this feature is enabled.

These configurations are defined in BLDC_RD_System.h.

4.3.10 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:

```plaintext
#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:

```plaintext
#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:

```plaintext
CONFIGBTN0()
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.

4.4 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 for desired configuration.
4.4.1 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:

```c
#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.

4.4.2 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:

```c
#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.

4.4.3 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:

```c
#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.

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

```c
#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:

```c
#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.

4.4.5 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 (section 4.3.5). 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.

4.4.6 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.

4.4.7 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.

4.4.8 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.

4.4.9 Rotation Speed Resolution
The BLDC kit supports motor rotation speeds of up to 200000 RPM. However, the only 16-bit variables are used in the internal firmware 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 GUI tool assumes this value to be 10.

5 Motor Control Interface Utility
The Motor Control Interface Utility 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 Create Project tab. When Generate Project button is clicked (after selecting appropriate configurations), a C language header file is generated by the utility. The generated header file should be included in BLDC firmware (via bldcdk.h).

![Motor Control Interface Utility Build-time configuration](image)

Most of the build-time configuration has been covered in section 4.3. The following sub-sections describe the configuration parameters that were not covered earlier.

5.1.1 Project Name
In this text box, user has to specify project name which becomes part of header file name (BLDC_RD_<project name>.h).
5.1.2 Project Directory
In this text box, user must specify BLDC firmware include directory. Alternately, user can use browse button to select the include directory.

5.2 Runtime Monitor and Control
Run-time parameters can be read, plotted and set to desired value in the Debug Project tab.

![Runtime Monitor and Control page](image)

This tab consists of three main areas:
1. Parameters
2. Plot
3. Dump

After the BLDC kit is powered and connected to host PC, user can establish a session with BLDC kit. This is done by specifying COM Port and clicking the Connect button. Baud rate can also be selected from main menu (Edit->Preferences) but it should always match baud rate of BLDC kit.

5.2.1 Parameters
The parameters are grouped into configuration parameters (fixed by BLDC kit at build-time), status parameters and control parameters. There are three operations that can be performed on a parameter:
1. Get – To read a parameter value from BLDC kit. All parameters support this operation
2. Set – To write new value to a parameter in BLDC kit. Only control parameters support this operation
3. Enable Update – To read a parameter automatically from BLDC kit in real-time. All parameters support this operation (though, enabling some of parameters doesn’t make sense). Enabled parameters are also plotted in plot area. Auto update works only when streaming mode is activated by clicking Enable Debug Stream.
All parameters can also be read at once by clicking Update All Values button. BLDC kit can also be reset to its initial state by clicking Reset Protocol button. In streaming mode, Get, Set, Update All Values and Reset Protocol buttons are deactivated.

All these parameters have been discussed in sections 4.3 and 4.4.

6 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 comes with a spreadsheet tool to guide the user to select the optimal resistor values to use in the design. The spreadsheet is at DesignTools\blcdesign.xlsx.

6.1 Design of Op-amp Gain for Current Sense Circuit

Section 3.2.3 discussed the operation of the current sense circuit, while section 3.5 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 1.8V output at the op-amp. The design goal is to match the op-amp voltage output to the internal C8051F850 comparator voltage reference (1.8V) 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.

6.2 Design of BEMF Zero-Crossing Detection Circuit

Sections 3.1.2 and 3.1.3 describe the Back-EMF filter circuit and the Virtual neutral filter circuit respectively. The design goals of these 2 circuits are:

- The filtered voltages do not exceed the MCU operating voltage even when the motor supply voltage is at maximum voltage. Some headroom for voltage spikes should also be accounted for in the design.
- 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. Ideally, designs should use the 9 resistor option because it gives the widest flexibility in the operating range of motor supply voltage.
7 Appendix A

7.1 Powertrain Board Schematic
7.3 Low-cost BLDC motor design sample schematic
DOCUMENT CHANGE LIST

Revision 0.1
  - Initial revision

Revision 0.2
  - Changed Low Cost BLDC Motor Design sample schematic
  - Modified sections 6.1 and 6.2
CONTACT INFORMATION
Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
and register to submit a technical support request

Patent Notice
Silicon Labs invests in research and development to help our customers differentiate in the market with innovative low-power, small size, analog-intensive mixed-signal solutions. Silicon Labs' extensive patent portfolio is a testament to our unique approach and world-class engineering team.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories, Silicon Labs, and Ember are registered trademarks of Silicon Laboratories Inc.
Other products or brand names mentioned herein are trademarks or registered trademarks of their respective holders.