CY3253-BLDC DEMO KIT Design Note

Ver1.0 Rev A

TABLE OF CONTENT

CH1. BLDC CONTROL BACKGROUND

A BLDC motor is constructed with a permanent magnet rotor and wire wound stator poles. Electrical energy is converted to mechanical energy by the magnetic attractive forces between the permanent magnet rotor and a rotating magnetic field induced in the wound stator poles.

Most BLDC motors have a three-phase winding topology with star connection. A motor with this topology is driven by energizing two phases at a time, and the other one phase is keeping float. The key to BLDC commutation is to sense the rotor position, then energize the phases that will produce the most amount of torque. The rotor travels 60 electrical degrees per commutation step. The appropriate stator current path is activated when the rotor is 120 degrees from alignment with the corresponding stator magnetic field, and then deactivated when the rotor is 60degrees from alignment, at which time the next circuit is activated and the process repeats.

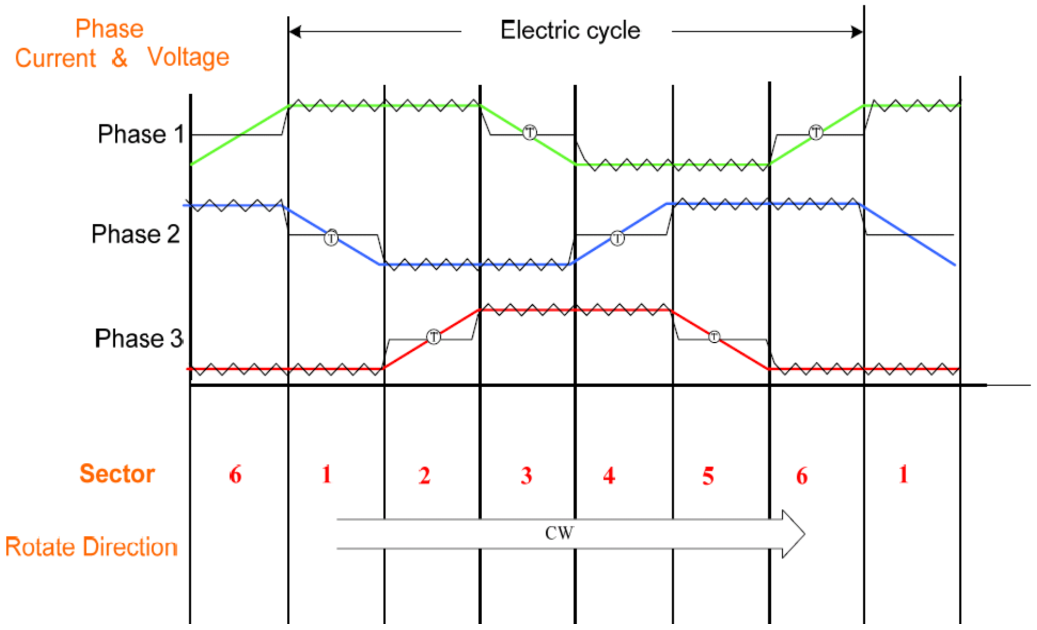


Figure1 BLDC BEMF and Current WaveForm

To implement this sequence, it is important to know the rotor position. This is done by using sensors, such as Hall-Effect Sensors (sensored control), or by sensing back EMF (sensorless control).

When the motor is spinning, the permanent magnet rotor moving past the stator coils induces an electrical potential in the coils called Back Electromotive Force, or BEMF. BEMF is directly proportional to the motor speed.

Every commutation sequence has one of the windings energized positive, the second negative and the third left open. The voltage polarity of back EMF crosses from a positive to negative or from negative to positive (so call zero-crossing) between two commutations. In ideal cases, the zero-crossing of BEMF happens 30 electrical degrees after last commutation and 30 electrical degrees before the next commutation. By measuring the zero-crossing of BEMF and the 30 degrees time interval, the controller can use a timer to do the 30 degrees delay after the zero-crossing point, then do the commutation without a HALL sensor.

CY3253 kit supports both sensored control and sensorless control. The two control scenarios share the same HW resource.

Please refer to Cypress Application Notes AN42102 more detail.

CH2. BLDC CONTROL WITH HALL SENSOR ON CY3253 KIT

The block diagram of BLDC motor sensored control based on PSoC is shown in 错误！未找到引用源。2.

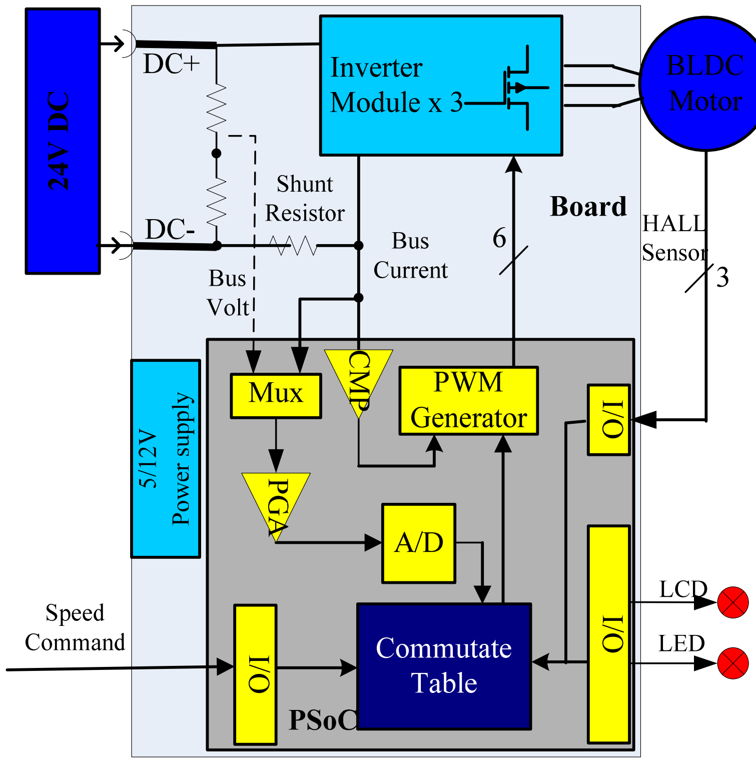


Figure2 Block Diagram of BLDC control with HALL Sensor

Algorithm of BLDC control with hall position sensor is based on Cypress Application Note AN42102- BLDC--Closed-Loop Speed Control Based on CY8C24x33. Please find out the application note for more details about BLDC control with hall effect sensor.

The following chapters describe the BLDC sensor-less control implementations in details.

CH3. BACK EMF ZERO-CROSSING DETECTION THEORY

For 3-phase BLDC, we often use 3-phase power inverter circuit for 3-phase BLDC control, as figure 3 shows:

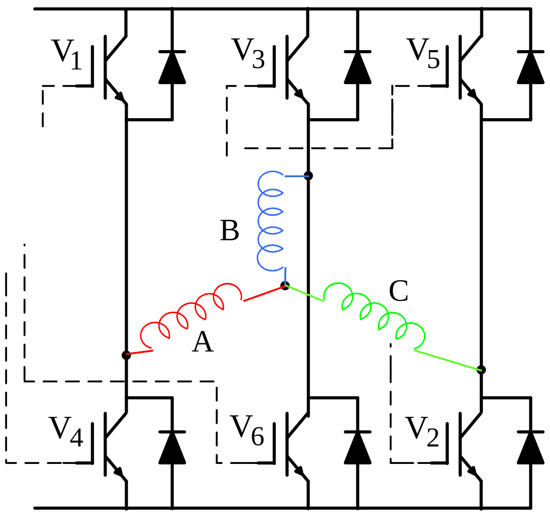


Figure3 3-Phase Power Inverter

It is not easy to detect backEMF signal from PWM noise. Moreover, BackEMF is referenced to neutral voltage (Vn). Vn is not a constant, while terminal voltages (Va,Vb,Vc) are easy to measure. We can deduce their relationship of BEMF and terminal voltage.

Assuming that phase C is the non-fed phase it is possible to write the following equations for the three terminal voltages:

$$Va = RIa + L(dIa/dt)+ Ea + Vn \qquad (1)$$
$$Vb = RIb + L (dIb/dt) + Eb + Vn \qquad (2)$$
$$Vc = Ec + Vn \qquad (3)$$

As only two currents flow in the stator windings at any one time, two phase currents are equal and opposite. Therefore,

$$I_a = -I_b$$

Thus, by adding the three terminal voltage equations we have,

$$V_a + V_b + V_c = E_a + E_b + E_c + 3V_n$$

It is evident that at the Bemf zero crossing points the sum of the three backEMFs equals to zero. Therefore the last equation reduces to,

$$V_a + V_b + V_c = 3V_n \qquad (4)$$

For the non-fed phase (zero current flowing), the stator terminal voltage can be rewritten as follows:

$$3E_c = 3V_c - 3V_n \qquad (5)$$

From (4) and (5):

$$3E_c = 2V_c - (V_a + V_b) \qquad (6)$$

From above equation, we can draw a conclusion: To detect the zero crossing of BEMF Ec, we need simultaneously sample 3-phase current, then, compare Vc with half of (Va+Vb).

But it is a heavy task for MCU to sample three phase current at some applications. There are extra conditions for equation (6). Suppose that PWM drive signals are arranged only on high side power switches, extra conditions for equation (6) are:

Sample at PWM-Off time

Assuming at a particular step, phase A and B are conducting current, and phase C is floating. The high-side switch of phase A is controlled by the PWM and the low-side switch of phase B is on during the whole step. The terminal voltage Vc is measured.

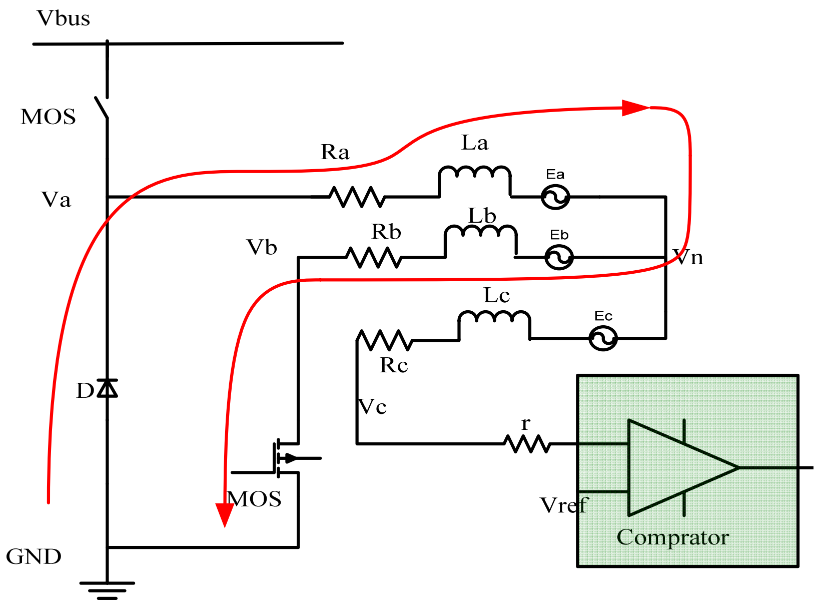


Figure4 BEMF and Terminal Voltage During PWM-Off

From figure 4, ignoring voltage drop on Vmos and Vd, we can seen that,

$$Va = Vb = 0 \qquad (7)$$

Combine (7) and (6)

$$3Ec = 2Vc$$

So, to decect zero crossing of backEMF Ec, just need to sample terminal voltage Vc and compare the value with zero.

Sample at PWM-On time

If sample at PWM-On time, in figure 3, high-side switch is on. Therefore:

$$Va = Vdc; \; Vb = 0 \qquad (8)$$

Combine (8) and (6)

$$3Ec = 2Vc - Vdc$$

So, to detect zero crossing of backEMF Ec, just need to sample terminal voltage Vc and compare it with half of DC bus.

Sample with filter

In some applications, filters are using to eliminate PWM signals. In such circumstances:

$$Va + Vb = Dp^{*} \; Vdc \qquad (9)$$

Here, Dp is duty-cycle of PWM signal.
Combine (9) and (6)

$$3E_c = 2V_c - D_p * V_{dc}$$

So, to detect zero crossing of backEMF Ec, just need to sample terminal voltage Vc and compare the value with half of Dp*Vdc. Usually, Dp is not a constant value. In some applications, deem Dp as “one” and make compensations to eliminate the error.

CH4. CY3253 KIT BEMF ZERO-CROSSING DETECTION PRINCIPLE

Based on above description, different kinds of BEMF zero-crossing detection algorithms are wildly used in applications:

- Compare Virtual Neutral Point with Zero Voltage
- Compare Terminal Voltage with Zero Voltage at PWM-Off
- Compare Terminal Voltage with Half DC Bus-Voltage at PWM-On
- Compare Filtered Terminal Voltage with Compensated Voltage

These algorithms either need a high speed ADC or external PWM latch circuit.

Thanks for versatility of PSoC chip, CY3253 kit implements a BEMF zero-crossing detection algorithm - Terminal Voltage Compared with Half Bus-Voltage via internal PWM Synced sample/hold circuit.

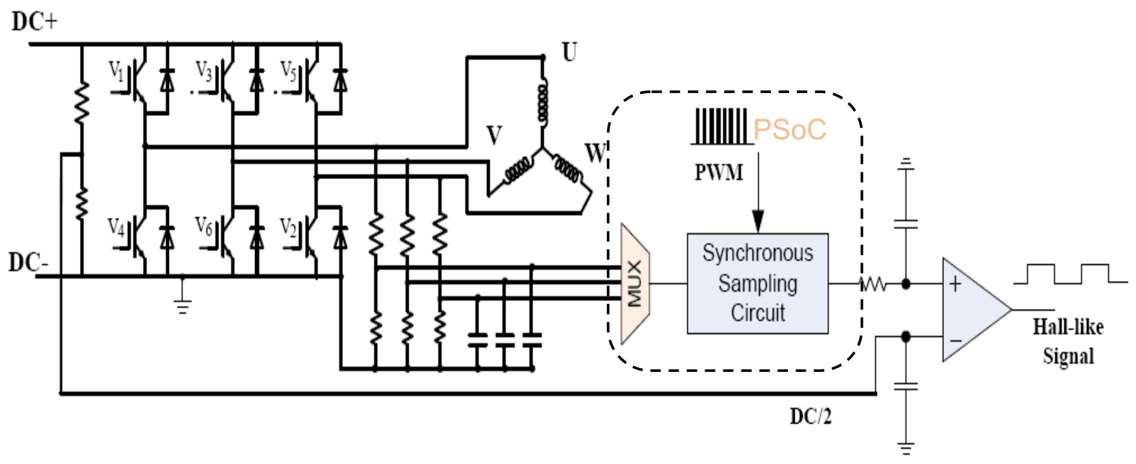


Figure5 BEMF Zero-crossing Detection

The circuit for BEMF zero-crossing detection is shown in figure 5. This circuit includes a multiplexer, PWM synchronous sample/hold circuit and a comparator.

- The multiplexer can switch among 3-channel BEMF signals into the sampling circuit according to the commutation status.
- The synchronous sample circuit can sample the BEMF signal to RC filter when PWM on, when PMW off, the sampling circuit will turn off to keep high-Z status, and voltage on the comparator input will keep the same as PWM turning off time.
- A delay can be added after PWM rising edge to disable the influence from power devices current transition.
- A comparator to compare the analog signal after synchronous sampling circuit with the DC bus scale down voltage. Then output the signal without PWM carrier waveform to indicate the commutation.

CH5. PSOC EMF ZERO-CROSSING DETECTION IMPLEMENTATION

Figure 6 shows the PSoC implementation:

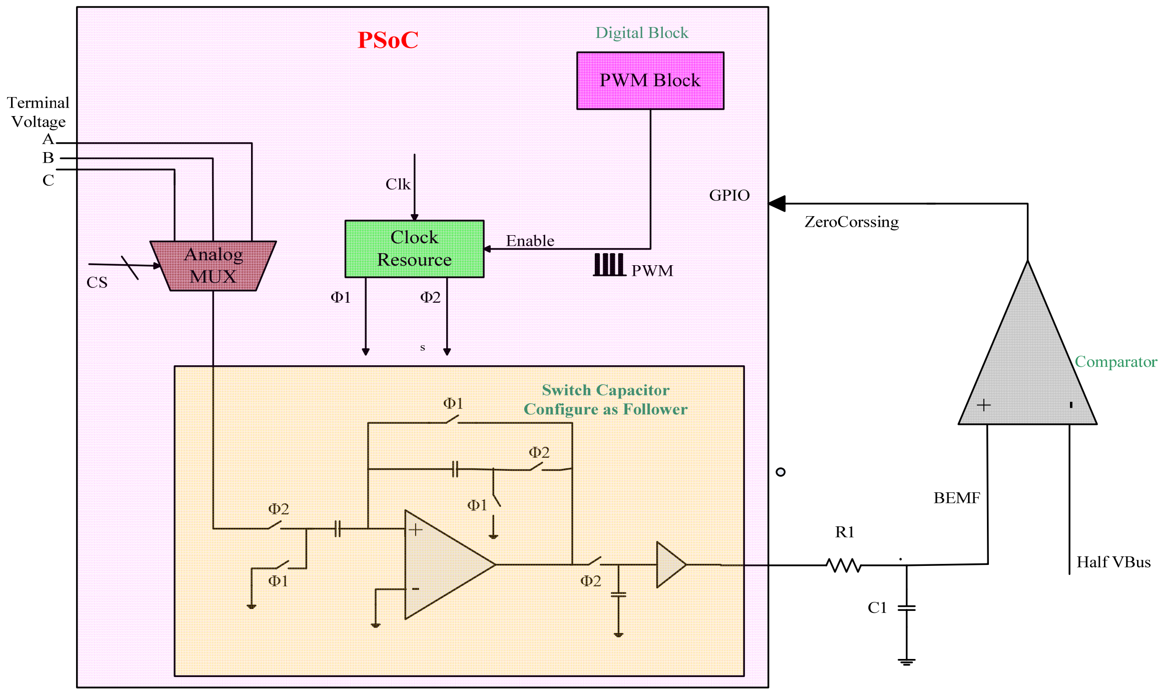


Figure6. PSoC Implementation of BEMF Zero-crossing Detection

One AMUX8 User Module multiplexes 3-channel terminal voltage signals into the sampling circuit.

One PWM8 User Module works as a clock resource, the input clock frequency is much higher than PWM frequency and is enabled by the PWM signal.

One SC (Switch Capacitor) UM works as a 1:1 amplifier, it is driven by high frequency clock Φ1 and Φ2.

An external RC circuit filters glitch.

-These User Modules construct a synchronous sampling circuit. When PWM on, the clock source that provides to the SC block will be enable, and the BEMF signal can charge to C1; when PWM off, the voltage on C1 keeps same.

-The demodulated BEMF is compared with half Bus-Voltage by an external comparator. It is obvious that the output signal out the comparator is the BEMF zero-crossing signal. Then, PSoC chip samples the BEMF zero-crossing signal via a GPIO pin.

CH6. BEMF ZERO-CROSSING DETECTION & COMMUTATION

Figure 7 illustrates BEMF zero-crossing detection procedure in software. The upper picture is the BEMF signal of one of the three channels, and the second picture is the algorithm implementation for all three phase.

- At the commutating moment, due to the flywheel of diode, there will be an incorrect pulse impact on BEMF zero-crossing signal. To neglect this impact, FW discards the first few samples of BEMF after commutation. This period of is called “Blank” period.
- After “Blank” period, FW reads zero-crossing signal in every PWM period. But, usually, near the zero-crossing point of BEMF, due to the voltage ripple, the output of comparator probably has several flip-flops. To validate the zero-crossing, PSoC judges a continuous high/low level following a low/high level as the correct zero-crossing.
- Zero-crossing validation will cause a little delay against the real zero-crossing moment. FW will do compensation the delay based on calculating and experiments.
- In ideal cases, according to figure1, the zero-crossing of BEMF is happened 30 electrical degrees after last commutation and 30 electrical degrees before the next commutation. Hence, when a zero-crossing checked, a 30 electrical degrees delay time should be memorized to evoke next commutation. A Timer16 user module does this work.

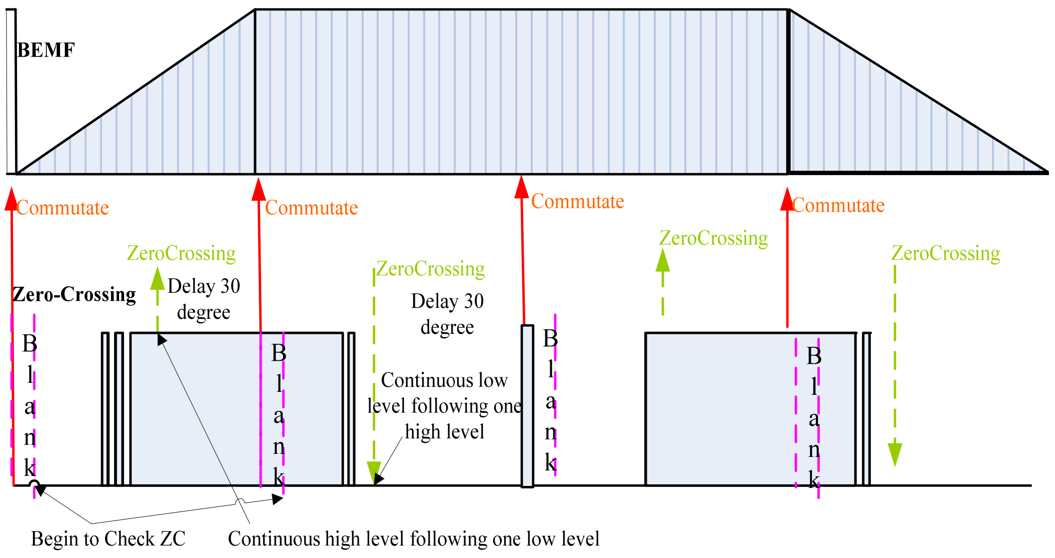


Figure7 BEMF Zero-crossing Detecting Procedure

Parameter tuning in FW:

Array ***iRisingEdgeCompensation[ ]*** is used to adjust the compensation for "Delay 30 electrical degree" (rising edge) at different BLDC speed;

Array ***iFallingEdgeCompensation[ ]*** is used to adjust the compensation for "Delay 30 electrical degree" (falling edge) at different BLDC speed;

Array ***bBlankCntThreshold[ ]*** is used to set "Blank" period at different BLDC speed;

Function ***Check30DegreePassed()*** counts the delay of 30 electrical degrees by enabling timer interrupt.

CH7.CONFIGURE MULTI-FUNCTIONAL TIMER

In CY8C24533, there only one digital row, or 4 digital block totally. While for sensorless BLDC control, we need several 16-bit timers for commutation, period measurement etc. This section introduces how to share one 16bit timer.

As figure7 shown, in BLDC sensor-less control, the zero-crossing and commutating events should be time-stamped to measure the period value and do the commutaion. In this design， one free-running 16-bit timer is used to timestamp the event so as to calculate the period. The clock frequency of Timer16 is 1MHz, with a high input clock frequency, we can get better resolution when period measurement. Meanwhile, a 16-bit resolution will make sure that the period measurement will be not overflow.

From the introduction in CH1, the commutation of the motor is occurring at the center of two zero-crossing events, or 30 electrical degrees delay after a zero-crossing event. To do the commutation, the firmware should measure the delay time for the 30 degree electrical cycle. When detect a valid zero-crossing point, the firmware will set up a 30 degree delay timer. The commutation is trigger by the interrupt of the 30 degrees delay occurring. Here is the introduction of sharing the period measurement and 30 degree delay timer into a single 16-bit timer.

Difficulties

1. Side Effects of the API function ***Timer16_wReadTimer()***: Compare register contents are lost. The compare condition becomes true immediately or on the next timer input clock cycle depending on whether the Compare Type parameter is set to "Less than or Equal to," or "Less Than," respectively. If (or when) the user module and global interrupts are enabled, the interrupt will be serviced, quite possibly before this API function has returned control to its caller. The A and X registers may be modified by this or future implementations of this function.
2. Overflow effect: When ***(counter value - delay)*** is less than zero. It will overflow, thus, compare register of Timer16 is set as relatively large value. For example, if currently counter value is 0x0080, and the desired delay value is 0x0090, then ***(counter value - delay)*** value is less than zero. Due to effect of overflow, the compare register will be set as 0xFFF0. If compare type interrupt is enabled, the system jumps to interrupt ISR immediately. Obviously, this is an error.

By elaborate design, these difficulties can be overcome.

Measures

1. Use a variable to separate the ISR into two types. One type is generated by calling the function ***Timer16_wReadTimer()***. We called this ISR is useless or call it “blind”. Another ISR is the desired one, generated by compare interrupt.
2. When the Overflow occurred, the delay can be divided into two periods. First, enable the “Terminal Count” interrupt. Use another variable to identify this ISR. Second, in this “Terminal Count” ISR, enable “Compare True” interrupt by setting relative register. When next interrupt occurred, PSoC will jump to use desired ISR. Figure 8 shows this trick.

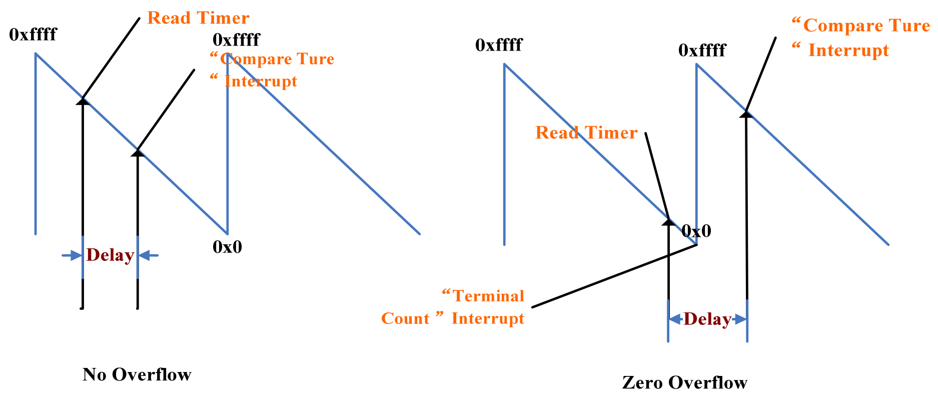


Figure8 Time16 Count Flow

CH8. FREE_RUNNING STAGE OF BLDC

BLDC phase BEMF amplitude is directly proportional to the motor speed. When motor doesn't run, there is no BEMF at all. Or when motor running at low speed, the BEMF is too weak to be detected. So before the BEMF can be measured by the firmware, there is a stage, the free-running stage, which is purposed to drive BLDC motor step by step to acquire initial BEFM zero-crossing signal prior to running sensorless control.

When running at a constant speed with open-loop stepping energization, the rotor position will be approximately 90 electrical degrees in advance of that when running correctly under sensorless control. As a result, the BEMF zero crossings can't be sensed. To make the zero crossings observable, it is necessary to accelerate the motor at a certain rate.

Therefore, in this stage, the PWM duty-cycle increased gradually, and the commutating period is longer than normal circumstance. A table ***baStartUpTimeTbl[]*** in FW defines the timeout of every commutating period. The value of table content is resulted from experiment based on different motor type. When enough valid zero-crossing events detected, FW will enter into normal synchronous running stage. If FW can't detect enough valid zero-crossing event during a period of time, an error is generated and BLDC have to stop and waiting for next round of free-running stage.

CH9. CHANGE BLDC ROTATION DIRECTION

For BLDC sensorless control, rotor position is acquired by detecting zero-crossing point of BEMF signals. According to commutation state, the rotor position is divided into six vectors. In each vector, commutation state is identical with others. Increasing or decreasing the sector (so called commutation) one by one generates continuous torque for rotation. One commutation event is followed by one zero-crossing detection event.
If increasing the sector one by one lets the rotor rotating in clockwise direction, then decreasing the sector one by one will rotate the rotor in Counter-ClockWise(CCW) direction. Figure 1&9 illustrates the relationship. It should be notified that the zero-crossing edge (falling/rising) is reversed when direction changed.

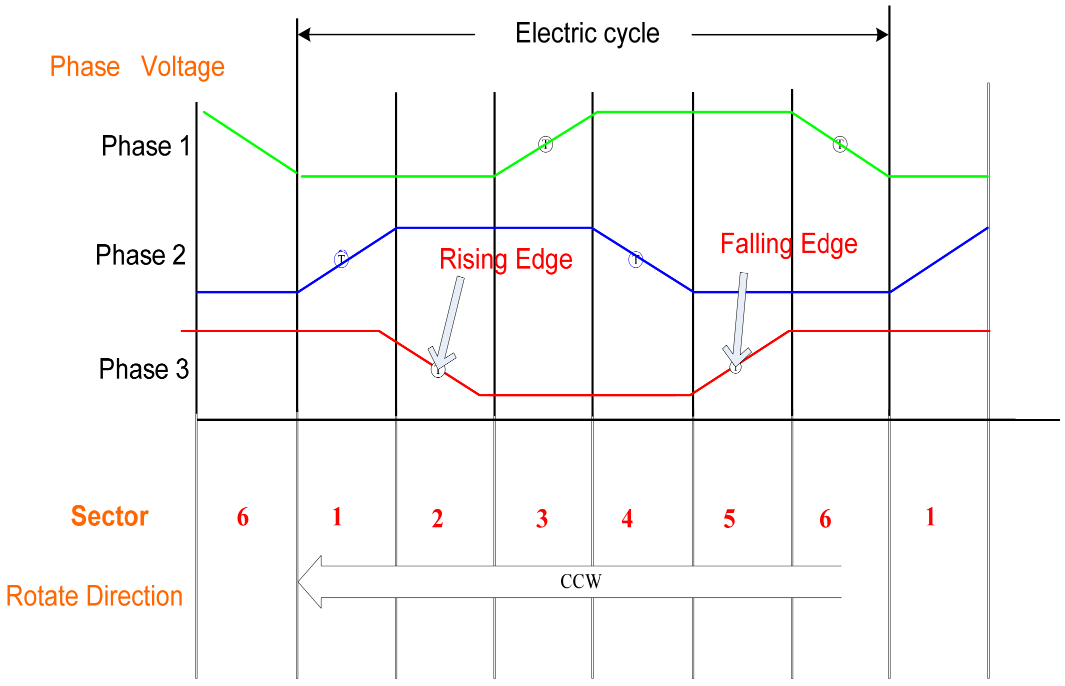


Figure9 Senorless BLDC CCW Commutation

CH10. PI SPEED CLOSE LOOP

The PI control algorithm is very useful in a continuous control system. There are two basic PI control algorithms: absolute mode and increment mode PI control algorithm. The following equation is a discrete expression of position mode of the PI algorithm.

$$u_k = K_P * e_k + K_I * \sum_{i=1}^{k-1} e_i + u_0 \qquad (10)$$

The disadvantages of the absolute mode PI algorithm are:

1. Switching between closed-loop and open loop has system impulsive force, which results in an unstable running of motor.
2. Output of integration action is related to all status in the past. This increases the workload of MCU, and also accumulates systematic error.

The disadvantages can be solved by using the increment mode PI algorithm. The formula is shown in the following equation.

$$\Delta u_k = u_k - u_{k-1} = K_P * (e_k - e_{k-1}) + K_I e_k \qquad (11)$$

The control increment is Output, which is added to the current control input. MCU implementation also becomes easier.

In closed-loop speed PI control system: e_k is speed error.

Figure10 is the block diagram of PSoC PI closed-loop speed control.

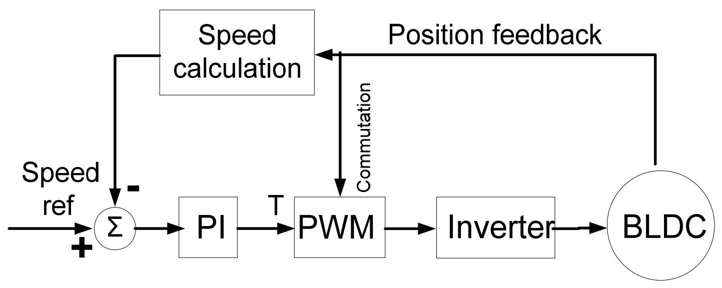


Figure10 Diagram of Speed PI Close Loop

PSoC system resources include an on-chip register based Multiply Accumulate (MAC). The MAC block is a fast 8-bit multiplier or a fast 8-bit multiplier with 32-bit accumulator. To execute a multiply, write the value to any internal multiplicand register. Immediately after the write of the multiplicand, the product is available at product registers. The Multiply Accumulate calculation has the same operation. The MAC block is a very useful resource for the PI algorithm.

CH11. FIRMWARE ARCHITECTURE

There are one main loop and 2 interrupt service routines: PWM ISR, Timer ISR.

The PWM frequency is 16.7 KHz. System enters PWM ISR every 67 us. In PWM ISR, 67us software timer is generated to supply rough time base.

Timer ISR is serviced when delay 30 degree completed. In this ISR, BLDC commutation is enabled, BLDC motor commutates to next sector.

FW Project Directory

```
|---Source
|       |----boot.asm                 ….. M8C Boot Code for CY8C24xxxB controller family
|       |----commutate.c              …..  BLDC commutation control function
|       |----freerun.c                ….. Stepping BLDC for initial zerocross event checking
|       |----key_det.c                …..  Detecting key events on key_mode&key_start
|       |----main.c                   …..  Main function
|       |----pwmctrl.c                …..  Assign PWM to different  I/Os on different ports
|       |----check30degreepassed.c     ….. Set interrupt for enabling next commutate
|       |----checkzerocross.c         ….. Check and verify zerocross event
|       |----spdloop.c                …..  Speed regulator for speed close loop
|       |----timestamp.c              …..  Timer16 interrupt handler and PWM Int handler
|       |----spdref_adc.c             ……Sample speed reference value on pototiometer
|
|----Headers
|       |----adc.h                    ….. Header  file for ADC sampling
|       |----flags.h                  ….. Header  file for key event flags
|       |----interface.h              ….. Header file for variables definition
|       |----led.h                    ….. Header file for LED on/off control
|       |----main.h                   ….. Header file for main.c
|       |----memory.inc               …… Memory Model and Stack Parameter Definitions
|       |----pwm_ctrl.h               …..  Header  file for PWM control
|       |----motorstage.h            …… Header file for defining motor run state
|       |----parameter.h             …… Header file for defining key constant
|       |----speed.h                  …… Header file for defining speed level
|
|----Library Source
|----Library Headers
|----External Headers
|----flashsecurty.txt
```

FW FlowChart – MainLoop

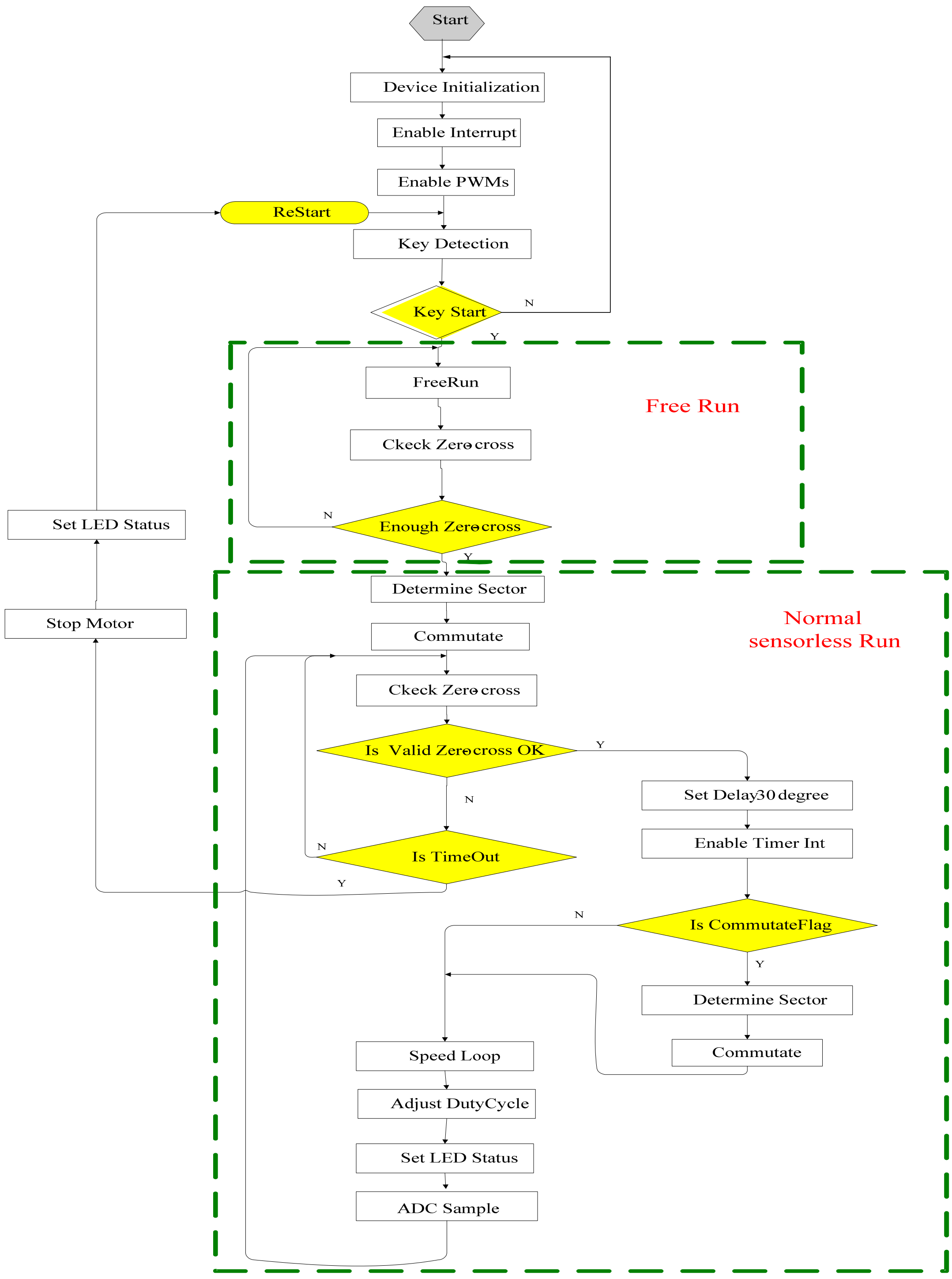


Figure 11 Firmware FlowChart

CH12. OVER CURRENT PROTECTION

As the CY8C24533 resource limitation, CY3253 kit implements over current protection by external hardware circuit.

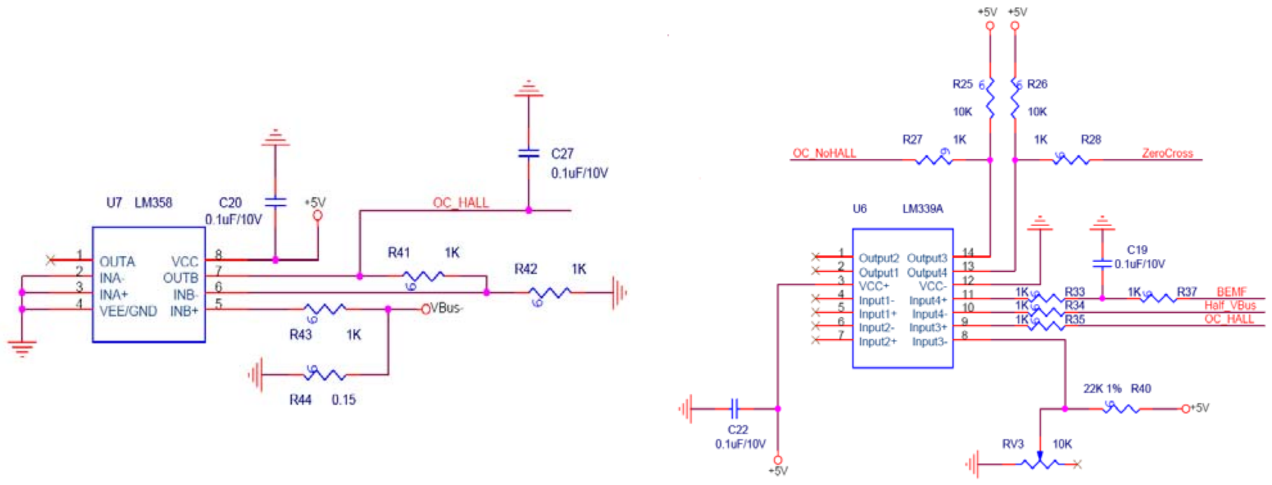


Figure12 Over Current Protection Circuit

As shown in figure 12, R44 is the current shunt resistor. OC_Hall signal, which is amplified bus current signal, is compared with a reference voltage via a comparator. The reference voltage is given by a potentiometer RV3.

Usually, the output signal (OC_Nohall) of comparator is low. When bus current is large than the threshold, i.e. OC_Hall voltage larger than the reference voltage, OC_Nohall comparator will be high.

OC_Nohall is routed to the GlobleInput pin connection of PSoC. This signal is inverted and routed into Look-up Table (LUT). In LUT, PWM signal and the inverted OC_Nohall form an AND gate. So, when OC_Nohall is high, PWM output is off. Motor current hereby is limited. Figure 13 is PSoC internal connection:

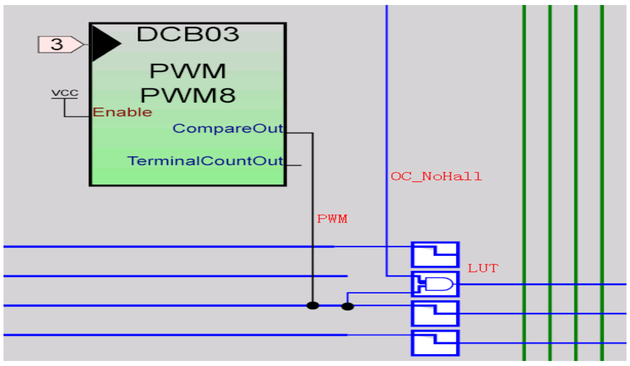


Figure 13 Inner Configure for Over Current Protection

CH13. WAVEFORM

A. Terminal Voltage VS Filtered BEMF

The yellow curve is one channel terminal voltage, and green one is filtered BEMF of three channels.

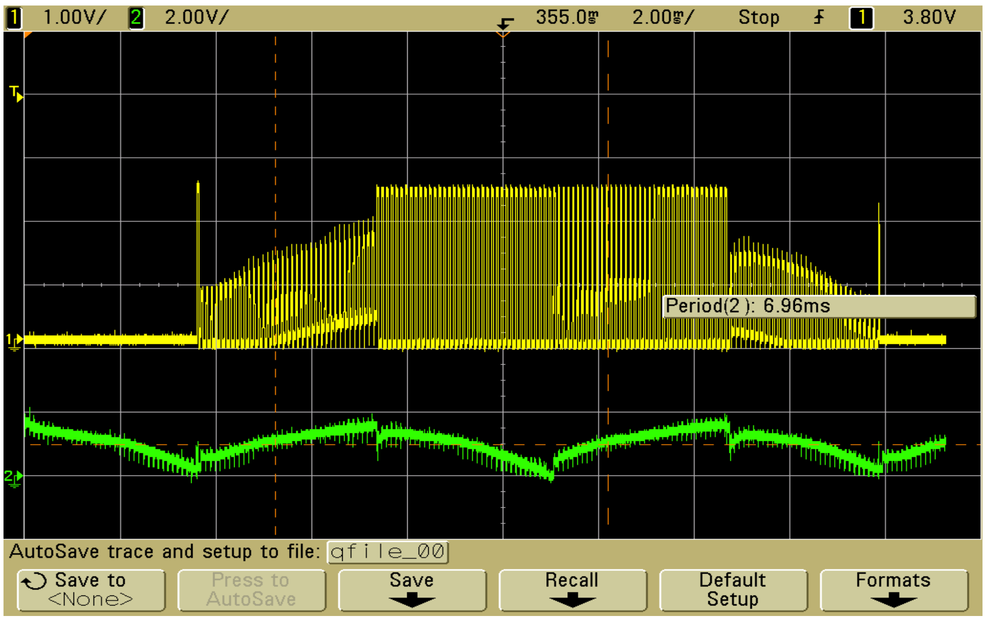


Figure14 Terminal Voltage VS Filtered BEMF

B. Terminal Voltage VS Zero-Crossing

The yellow curve is one channel terminal voltage, and green one is filtered BEMF of three channels.

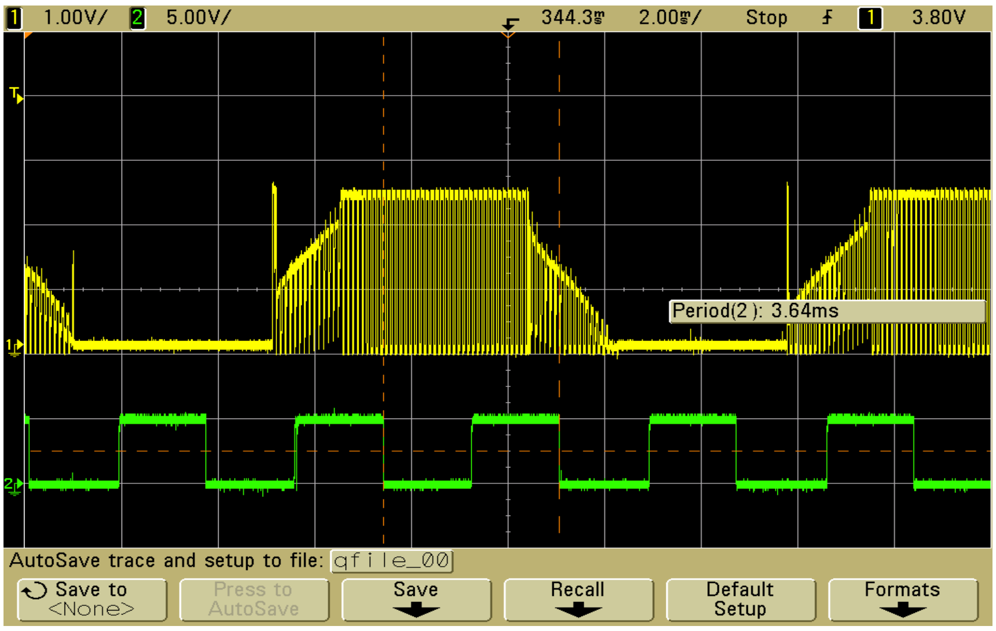
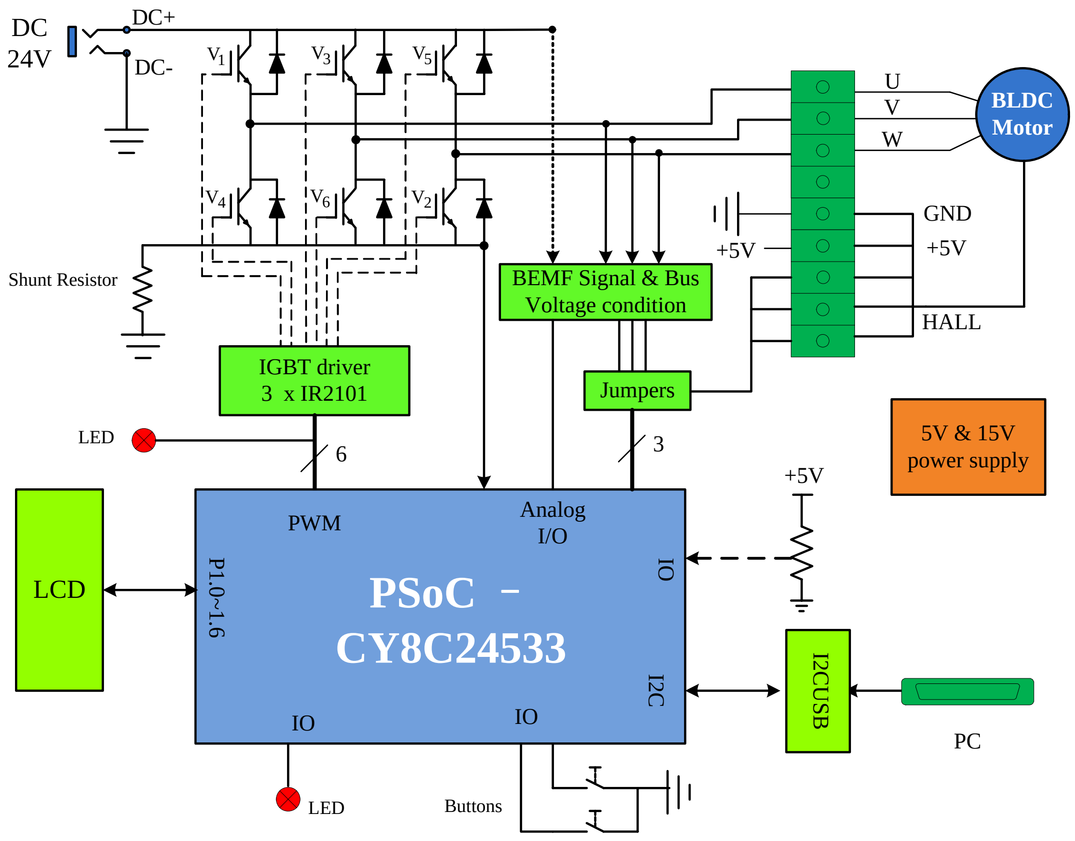
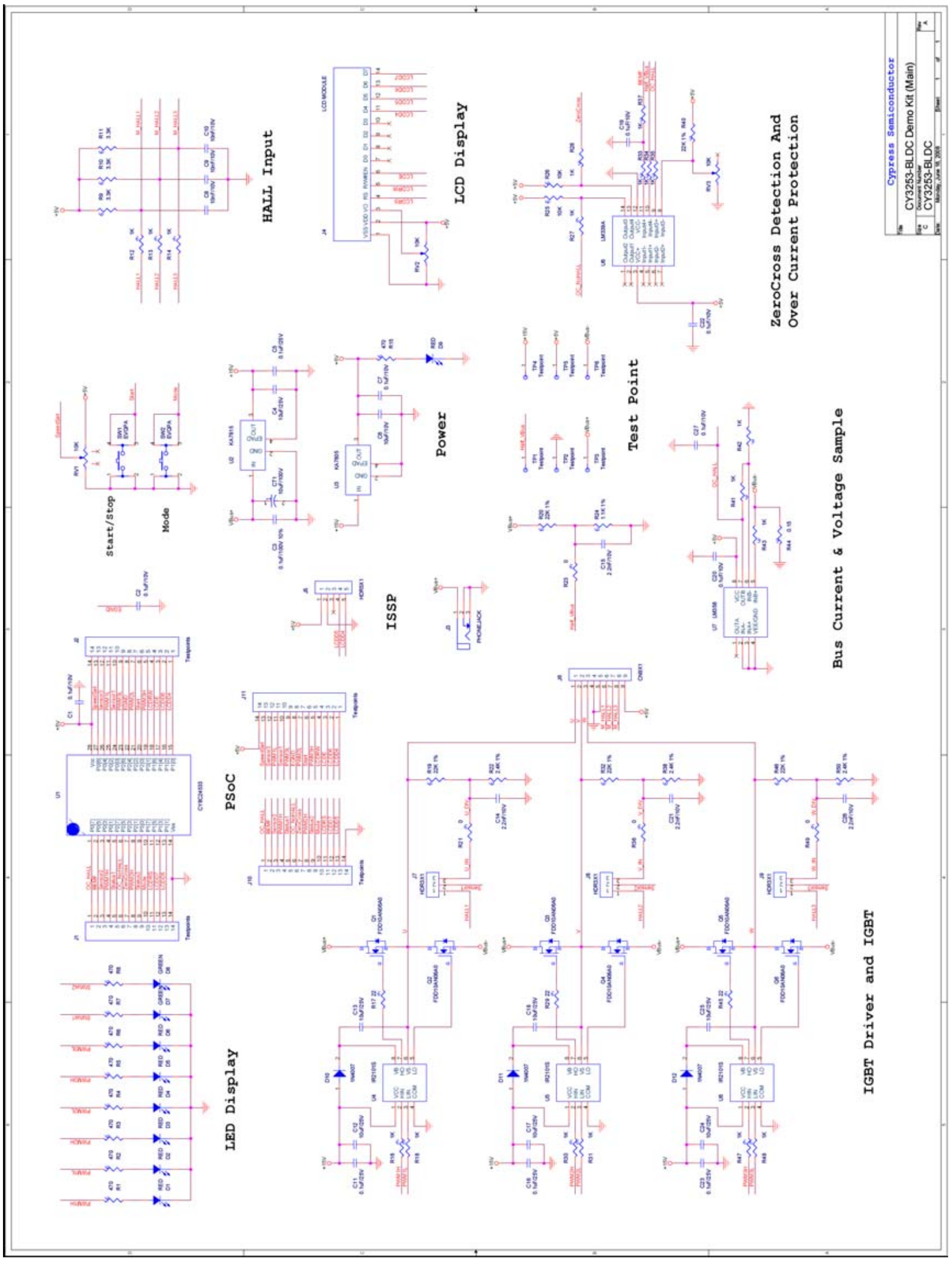


Figure15 Terminal Voltage VS Zero-Crossing

14.1 System Diagram



14.2 Schematic



CH15. REFERENCES

1. **BLDC Closed Loop Control Based On CY8C24x33**, Bill Jiang and Jemmey Huang. AN42102, Cypress
2. **Brushless DC Motor Control**, Andrey Magarita, AN2227, Cypress
3. **Using the DsPIC30F for Brushless DC Motor Control**，Charlie Elliote, AN901, MicroChip
4. **A Novel Direct Back EMF Detection for Sensorless Brushless DC (BLDC) Motor Drives**, Jianwen Shao, Dennis Nolan, and Thomas Hopkins, ST MicroElectronics
5. **Commutation Trigger for BLDC Drive**, DCS Group, Texas Instrument
6. **JHU#033**, Cypress Internal Correspondence
7. **QDGU#003**, Cypress Internal Correspondence
8. **QDGU#010**, Cypress Internal Correspondence
9. **QDGU#019**, Cypress Internal Correspondence