AN56541

Sensorless BLDC Motor Controller Implementation Using PSoC®

Author: Rajiv Badiger
Associated Project: Yes
Associated Part Family: CY8C24533
Software Version: PSoC® Designer™ 5.0
Related Application Notes: None

If you have a question, or need help with this application note, contact the author at rjvb@cypress.com.

This application note demonstrates about implementing Sensorless BLDC motor controller using PSoC® device.

Contents

Introduction

Nowadays, BLDC motors are gaining popularity due to numerous advantages when compared to traditional DC and AC motors. The technology to drive BLDC motors is also advancing. These days, Sensorless BLDC motors are in the limelight due to its lower cost and increased reliability.

In general, to detect the rotor position in a BLDC motor, Hall sensors are used. But this increases the cost of the motor and also the reliability issues. Hence, the trend in the market is with the Sensorless BLDC motors. The controller implementation is required to handle the additional burden of sensing the rotor position without the hall sensors.

This application note covers

- BLDC motor
  - Architecture
- Detecting the position of rotor in a BLDC motor using back EMF
- Algorithm to implement Sensorless BLDC controller
- PSoC Designer Project
  - Hardware Configuration
  - Firmware architecture
- Test Results
- Appendix: Schematics of design

BLDC Motors

BLDC motors consists of two mechanically isolated sections

1. Rotor
2. Stator

The rotor is made using a permanent magnet and the stator is wound with copper wire and is fitted to the motor case. In general, the rotor is made using copper coils and the stator is made using permanent magnet in DC motors.

Figure 1 shows the architecture difference of BLDC motors and DC motors.

Figure 1. Architecture of BLDC motors and DC motors

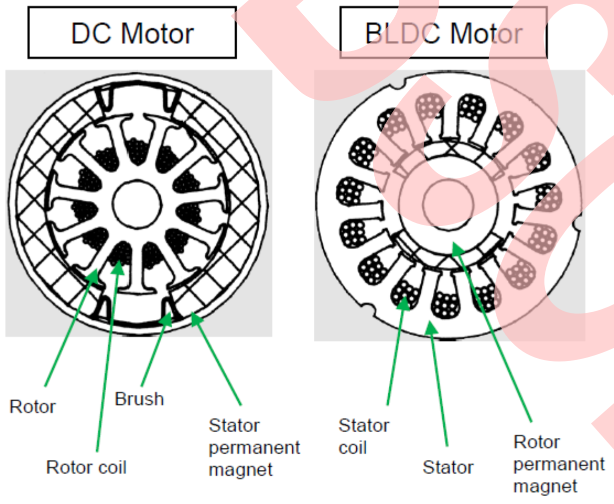
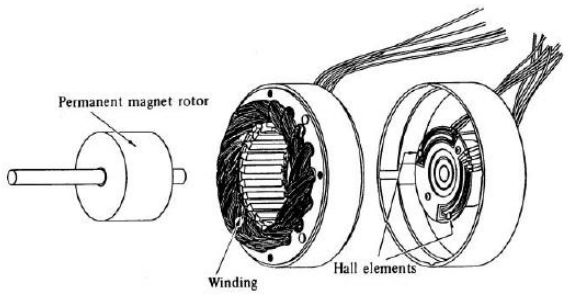


Figure 2 shows the disassembled view of the Sensored BLDC Motor.

Figure 2. Disassembled view of Sensored BLDC Motor



Hall sensors are used to detect the position of rotor which helps in initiating next commutation. This application note focuses on implementing BLDC control without the help of these hall sensors.

3 phase BLDC motors are widely used. These motors have 3 stator windings per single stator pole case.

Multiple stator pole motors have multiples of 3 phase stator windings. These windings are connected in parallel and a single terminal is taken out from each phase. Hence, for a 3 phase motor, only 3 terminals are taken out. Drive signals are applied to these terminals.

General Driver Architecture for BLDC Motors

Figure 3. Typical driver architecture for BLDC motor

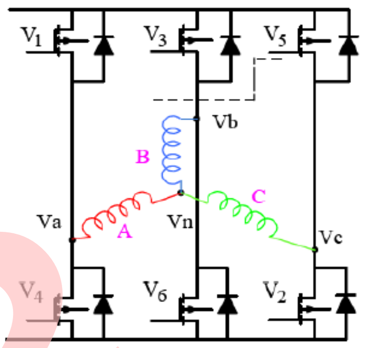


Figure 3 shows the typical driver architecture for BLDC motor. There are two sets of MOSFETs. One set, TOP MOSFETs, connecting the stator coil terminal to +V and the other set, BOTTOM MOSFETs, connecting the stator coil to –V. No two MOSFETs for the same end of the stator coil are turned ON at the same time. For example, MOSFET 1 and 4 are never turned ON simultaneously. MOSFETs are turned ON depending on the direction of current needed through the stator coils.

Operating BLDC Motor

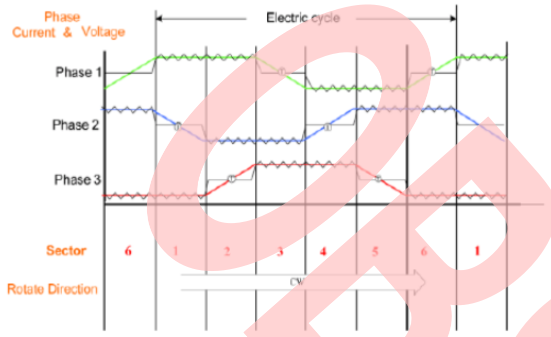
This section provides the operating procedure details of the motor. There are mainly two states in operating BLDC motor.

1. Sense the position of rotor
2. Initiate commutation at appropriate time

BLDC motors are available with hall sensors to provide the position information of the rotor. But the presence of hall sensors leads to higher motor cost. The position can also be sensed by detecting the zero crossing of the back EMF.

Figure 4 shows the general operating waveforms of phase voltages and currents.

Figure 4. General operating waveforms of phase voltages and currents



For 3 phase stator windings, there are six sectors or stages of commutation in a 360º electrical cycle. One sector corresponds to 60º. At one instance, voltage is applied across two stator terminals and the third stator terminal is left open. As shown in previous figure, phase voltage clamped to some value (high or low), is due to the voltage applied. The non-zero slope portion of the waveform is obtained when the terminal is kept open.

As seen in the waveform, the phase voltage crosses zero exactly in between two commutation periods; that is where there is 30º phase difference for the next commutation after the zero crossing. If you could provide a time delay equivalent to this 30º phase, then the motor can be commutated in a similar way as it is done with hall sensors.

This is the basis of implementing sensorless BLDC controller implementation.

Since the back-EMF is a function of rotation speed of the rotor it is too less to detect in the beginning. Therefore, initially, the system is made to run in open loop, in which commutation is done after fixed time delay. This makes the motor to get started and generates the back-EMF. Later on, the main control loop takes care of detecting the zero crossing of back-EMF and initiates commutation accordingly.

This kind of control system has major disadvantages as given below

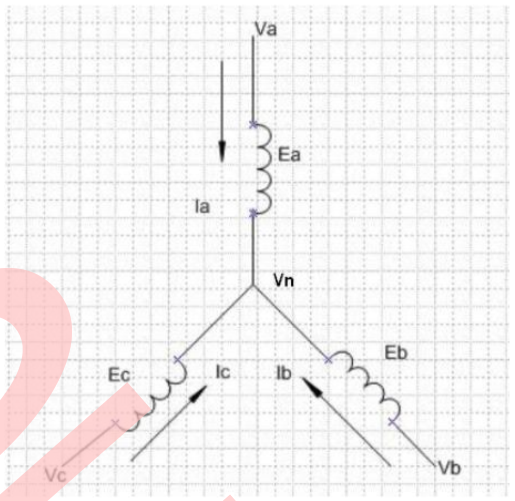
1. Requires one more state in the firmware in the beginning to get the motor started and generate back-EMF
2. Requires comparators for detecting the zero crossing of back-EMFs.
3. Detection of zero crossing of back-EMF requires the creation of a virtual neutral point (common terminal of star connection which is not available outside the motor).

This application note demonstrates the control strategy without having the state of running the system in open loop in the beginning and without creating virtual neutral point.

Detecting Zero Crossing

This section shows the method to detect the zero crossing of back-EMF.

Figure 5. Detecting Zero Crossing



The following equations show the voltages observed at the star connected stator coil terminals.

$$Va = R \times Ia + L\frac{dIa}{dt} + Ea + Vn$$ **Equation 1**

$$Vb = R \times Ib + L\frac{dIb}{dt} + Eb + Vn$$ **Equation 2**

$$Vc = R \times Ic + L\frac{dIc}{dt} + Ec + Vn$$ **Equation 3**

Where E is the back-EMF, I is the phase current, L is the inductance and R is the resistance of the stator coil, which are assumed to be same for all the phase coils of the stator.

As explained earlier, at one instance, voltage is applied to two stator terminals. The third one is left open.

For example, voltage is applied across terminals A and B, causing current to flow through phase coils A and B. Moreover these coil currents are the same. If terminal A is pulled high (VBUS) and B is pulled low (0 V).

$$Ia = -Ib = I$$

$$Ic = 0$$

The equation for terminal voltage at all the three phases can be calculated as,

$$Va = R \times I + L\frac{dI}{dt} + Ea + Vn = VBUS$$ **Equation 4**

$$Vb = -R \times I + L\frac{d(-I)}{dt} + Eb + Vn = 0$$ **Equation 5**

$$Vc = Ec + Vn$$ **Equation 6**

Adding Equation 4, 5 and 6 gives,

$$VBUS + Vc = Ea + Eb + Ec + 3Vn$$

Now, the summation of back-EMFs at any time is zero.

That is, $Ea + Eb + Ec = 0$

This makes the equation as,

$$VBUS + Vc = 3Vn$$

Implementing this finding in equation 6 gives,

$$Vc = Ec + \frac{VBUS + Vc}{3}$$

Simplifying further gives,

$$\frac{Ec}{2} = Vc - \frac{VBUS}{2} = VDIF$$ **Equation 7**

The above equation gives the formula for back-EMF for the phase coil whose terminal is kept open for the commutation cycle.

You can easily find the zero crossing by comparing the terminal voltage Vc and half of the VBUS voltage.

Thus, by detecting the zero crossing using this technique eliminates the requirement of comparator and the virtual neutral point. These voltages can be measured using an ADC which can also be used to read speed command from you, measure motor current and for many other functions.

Commutation Control Strategy

This section provides the details of the method used to determine the commutation rate based on the voltage difference as given in Equation 7. Before going into the control strategy, it is necessary to discuss the motor behavior, when the commutation rate is less or more than the ideal value. This ideal value depends on the voltage applied, the load on the motor, and the motor characteristics.

If the commutation rate is less than the ideal value

This creates stepper motor sort of behavior, wherein rotor moves to new position with the speed proportional to the applied voltage to the stator windings. After arriving at the new position, rotor gets locked to that particular position. This causes a decrease in the back-EMF and the stator starts sinking more current. If the next commutation is not initiated quickly, this can damage the stator coils due to excessive current, if protection network is not in place.

If the commutation rate is more than the ideal value

This causes loss of synchronization between the rotating magnetic field created by the stator coils and rotor.

It is acceptable that the motor be operated with a lesser commutation rate, but it is at the expense of a higher current.

The control scheme used here is to initially run the motor at a lesser commutation rate (because the ideal value is not known yet at the given operating voltage and the load) and then keep updating the commutation rate based on the voltage difference (VDIF) as given in Equation 7. If the expected slope of the back-EMF is positive and the voltage difference is found to be positive, then it means that the commutation rate is slower than the ideal value and further steps are taken to increase this commutation rate. This increase in commutation rate depends on the VDIF. As the commutation rate starts approaching the ideal value for the given voltage, VDIF drops to 0 and commutation rate is held constant.

On the other hand, if the expected slope of back-EMF is positive and if the VDIF is found negative, then the commutation rate is faster than the ideal value. The commutation rate is then reduced. Again this decrease in commutation rate depends on the VDIF.

A timer can be used to represent the commutation rate. The timer period value is updated based on the VDIF value. The PI control can be introduced here to provide stability in the system.

PSoC Designer Project

The project is based on Cypress PSoC CY8C24533.

Motor used for this application has following characteristics.

Table 1. Characteristics of Motor

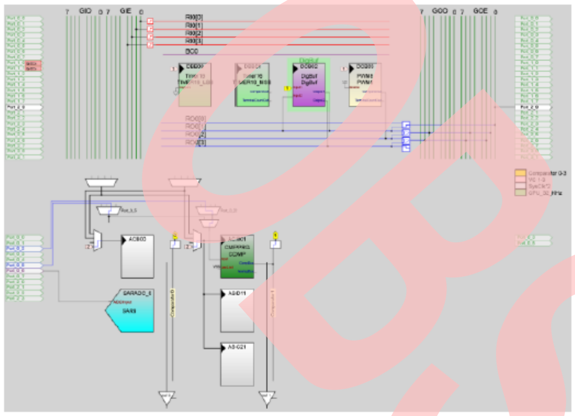
Parameter	Value
Power	80 W
Vin	24 VDC
Maximum Speed	3000 rpm
Torque	0.255 N-m

Appendix A: Schematics gives the schematic for the project.

Hardware Configuration

Figure 6 shows the snapshot of Device Editor View of PSoC Designer.

Figure 6. Device Editor View of PSoC Designer



The design uses the following modules.

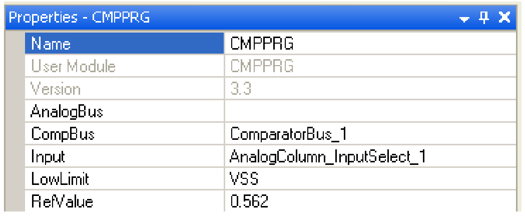
Module	Purpose
PWM8	To drive the high side and lower side MOSFETs
CMPPRG (comparator)	To detect high motor current
DAC	To provide reference to the comparator
ADC	To read speed input from POT, measure VBUS and stator terminal voltage (Va/Vb/Vc)
Timer 16 bit	To decide the time of commutation
DigBuf	To route the comparator output to digital section. This clips the PWM off when comparator output goes high.

Module Configurations

This section gives the properties of each module used in the project.

CMPPRG

This module senses the current flowing through the motor. For this, a small value resistance of 0.1 Ω is placed in series with the lower MOSFET source terminal. The voltage developed across this resistance is fed to the input of comparator.



Properties - CMPPRG

Name	CMPPRG
User Module	CMPPRG
Version	3.3
AnalogBus	
CompBus	ComparatorBus_1
Input	AnalogColumn_InputSelect_1
LowLimit	VSS
RefValue	0.562

This equation determines what to set the threshold of comparator to.

$$Vth = LowLimit + (Vdd - LowLimit) \times RefValue$$

$$Vth = 5 \times 0.562 = 2.81V$$

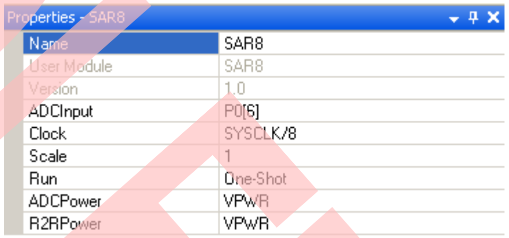
When all MOSFETs are turned OFF, the voltage across this resistance is 2.5 V. Hence, the motor current limit set here is,

$$Motor\ Current\ Limit = (Vth - 2.5V)/0.1\Omega$$

$$Motor\ Current\ Limit = 3.1A$$

SAR8

This is used to measure the stator terminal voltage and the VBUS voltage for detecting the zero crossing of back-EMF.



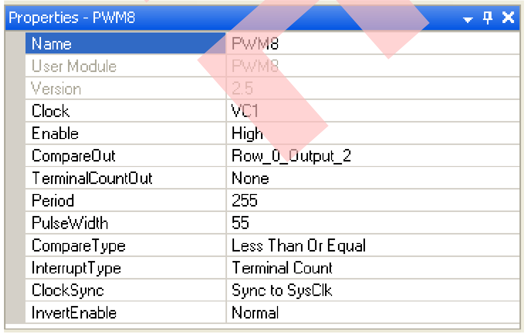
Properties - SAR8

Name	SAR8
User Module	SAR8
Version	1.0
ADCInput	P0[6]
Clock	SYSCLK/8
Scale	1
Run	One-Shot
ADCPower	VPWR
R2RPower	VPWR

The conversion time is set to 2.6 µs and the mode of ADC is set to one-shot.

PWM8

This module generates the PWM signal for driving the gate of TOP and BOTTOM MOSFETs. PWM frequency is set to 18 KHz.

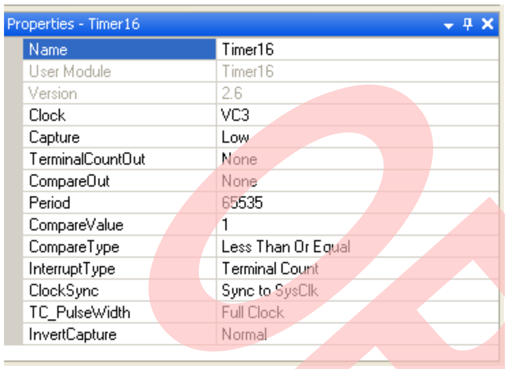


Properties - PWM8

Name	PWM8
User Module	PWM8
Version	2.5
Clock	VC1
Enable	High
CompareOut	Row_0_Output_2
TerminalCountOut	None
Period	255
PulseWidth	55
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

TIMER

Timer is used to decide the commutation rate of the motor.

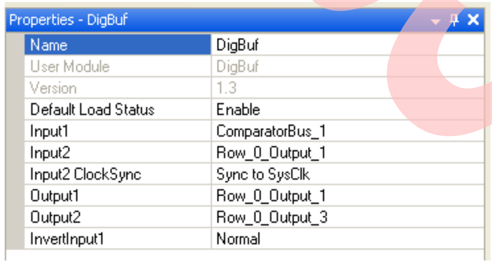


Properties - Timer16

Name	Timer16
User Module	Timer16
Version	2.6
Clock	VC3
Capture	Low
TerminalCountOut	None
CompareOut	None
Period	65535
CompareValue	1
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal

DIGBUF

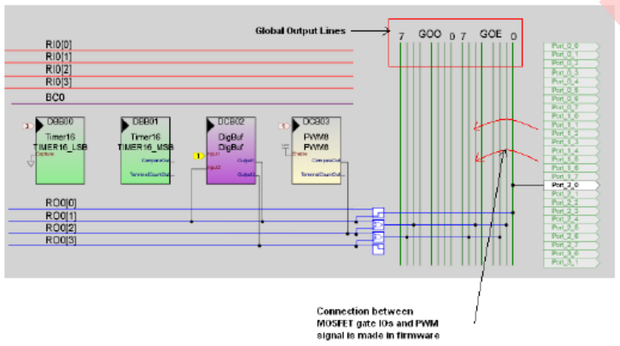
This module routes the comparator output to the LUT NOR Gate.



Properties - DigBuf

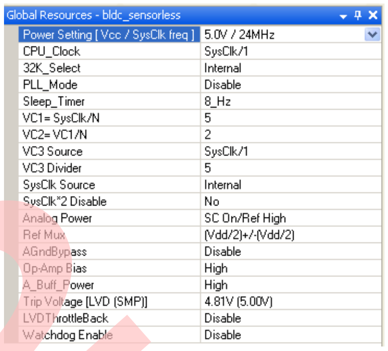
Name	DigBuf
User Module	DigBuf
Version	1.3
Default Load Status	Enable
Input1	ComparatorBus_1
Input2	Row_0_Output_1
Input2 ClockSync	Sync to SysClk
Output1	Row_0_Output_1
Output2	Row_0_Output_3
InvertInput1	Normal

Figure 7. PWM Routing



In the project, both HIGH side and LOW side MOSFETs are driven with the PWM generated from a single PWM8 module. This is possible due to the flexible routing scheme of PSoC device. At one point of time, only two lines require the PWM. The rest of the MOSFETs are turned off by grounding the I/Os which drive their gates. The connection from the pin to the global output line is set in firmware. The other input to the NOR gate is the DigBuf output which reflects the status of the comparator output (in the analog section). This Comparator output is logic 1 when the motor current exceeds the threshold, thereby pulling the NOR gate output low. This disables the PWM and provides protection for the MOSFET against higher currents.

Global Resource Settings



Global Resources - bldc_sensorless

Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/1
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	8_Hz
VC1= SysClk/N	5
VC2= VC1/N	2
VC3 Source	SysClk/1
VC3 Divider	5
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	High
A_Buff_Power	High
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

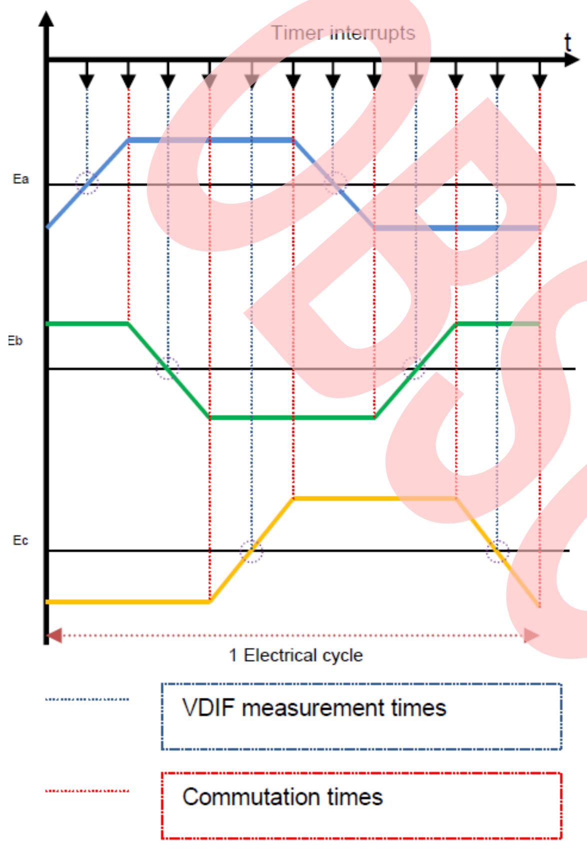
Firmware Architecture

As mentioned earlier, the rotor position sensing, that is, the detection of the back-EMF zero crossing and the commutation are the two main tasks of the controller. The back-EMF zero crossing is detected by measuring the difference of VBUS/2 and open terminal voltage using ADC. VBUS is scaled down by 2 in firmware after analog to digital conversion (VBUS>>2).

To decide the commutation rate, a 16 bit timer is used in the application. The timer period value is updated based on the VDIF value. The commutation rate is kept as half of the timer overflow rate. For one interrupt of the timer, a commutation is carried out and for the next interrupt of timer, VDIF is measured. This automatically creates 30º delay that was required for commutation after the detection of zero crossing of back-EMF.

Figure 8 shows the execution of commutation and VDIF measurement on timer interrupts.

Figure 8. Execution of commutation and VDIF measurement on timer interrupts



Commutation

The commutation involves connecting and disconnecting PWM lines and pulling some of the MOSFET gates low.

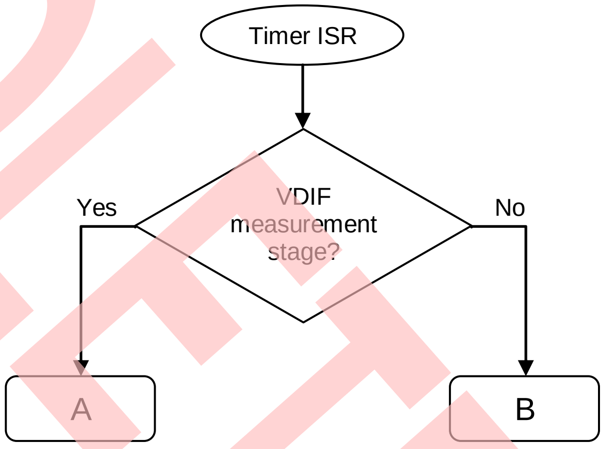
This is done with the help of PRTXGS (port global select) registers and PRTXDR (port data) register, where X defines the port number. Each port has one 8 bit wide PRTXGS and PRTXDR register. Each bit in these registers corresponds to one I/O pin for that port. Setting the bit in the PRTXGS register connects the pin to a particular global output bus. The status of the I/O pin then depends on the status of the bus. The data in PRTXDR reflects at the I/O port, given that the I/O pin is disconnected from the global output bus.

For example, if a pin requires a PWM, the bit in PRTXGS corresponding to that pin, is set to 1. This connects the pin to the global output lines to which the PWM output is already connected. Data in the PRTXDR corresponding to this pin does not matter now as it is connected to a global output line.

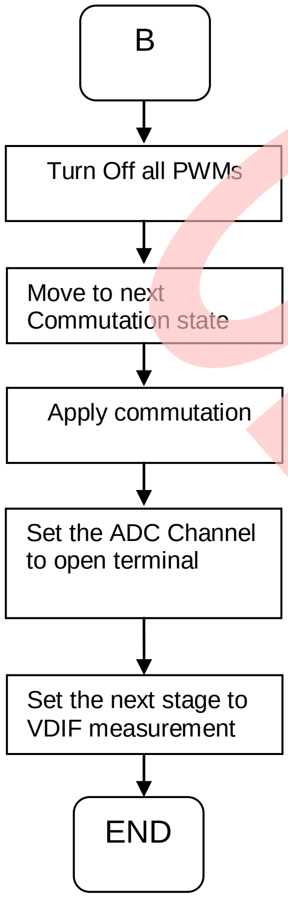
If the pin requires logic 0, the pin is disconnected from the global output bus by setting the corresponding bit in PRTXGS and PRTXDR bit to 0.

Firmware Flowchart

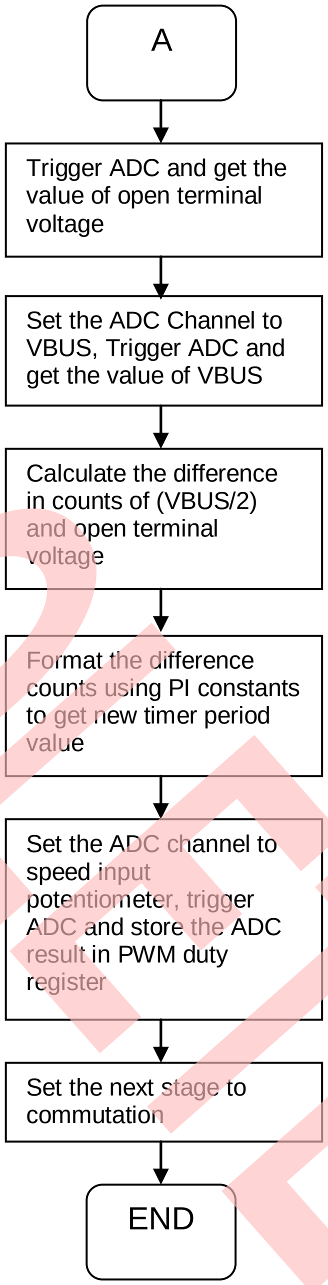
The entire control code, that is, the commutation logic and the zero crossing detection, is written in Timer ISR.



Commutation Stage


B
Turn Off all PWMs
Move to next Commutation state
Apply commutation
Set the ADC Channel to open terminal
Set the next stage to VDIF measurement
END

VDIF Measurement Stage


A
Trigger ADC and get the value of open terminal voltage
Set the ADC Channel to VBUS, Trigger ADC and get the value of VBUS
Calculate the difference in counts of (VBUS/2) and open terminal voltage
Format the difference counts using PI constants to get new timer period value
Set the ADC channel to speed input potentiometer, trigger ADC and store the ADC result in PWM duty register
Set the next stage to commutation
END

Tuning the Control System

This section guide you in one level deep to understand how the timer input frequency is set based on the motor speed limits and how the timer period value is derived based on the difference between error counts.

The timer input frequency needs to be determined from the motor maximum speed parameter and the timer resolution.

Consider a 16 bit timer with the motor and your specifications as follows.

Maximum Speed – Smax rps

Number of Poles – P

Commutation frequency at maximum speed is given by,

$$\mathit{Commutation\ Frequency} = \mathit{Smax} \times \left(\frac{P}{2}\right) \times 6$$

6 denotes the commutation stages in an electrical cycle.

As explained earlier, the timer interrupt frequency is twice the commutation rate.

So,

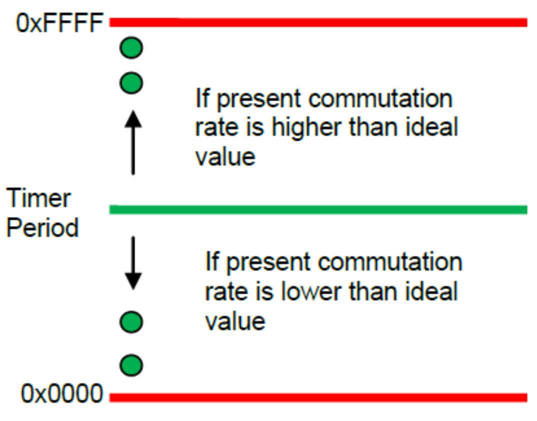
$$\mathit{Timer\ interrupt\ rate\ at\ max\ speed} = \mathit{Commutation\ Frequenc}$$

For a motor of 10 poles and maximum speed of 3,000 rpm, the timer interrupt rate can be found to be 3,000 Hz.

The timer period is derived based on the difference in error counts. Initially the timer period is set to the maximum timer count. In this project, 16 bit timer is used and hence the timer period is initially set to 0xFFFF. As the difference count grows on the positive side and if the expected slope is positive, the timer period is made lower than the present value. Following table shows actions taken on timer period under conditions of expected slope of back-EMF and the measured difference count.

Difference count	Expected slope of Back EMF	Timer period
Positive	Positive	Decreased
Positive	Negative	Increased
Negative	Positive	Decreased
Negative	Negative	Increased

Figure 9. Actions taken on timer period

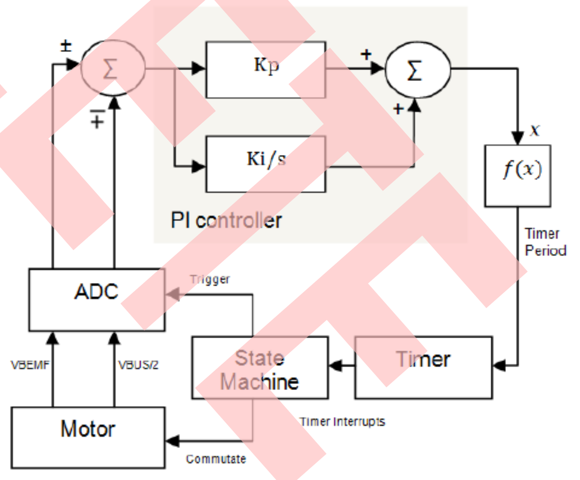

0xFFFF
If present commutation rate is higher than ideal value
Timer Period
If present commutation rate is lower than ideal value
0x0000

Now, you know what is done to the timer period when the difference count is found to be positive or negative. But, what you do not know is, by how much the Timer period is to be changed from the present value. This value is a function of the number of difference counts.

$$\mathit{Timer\ Count} = f(\mathit{Difference\ Counts})$$

This function comprises of passing the difference counts to PI controller logic and final scaling and offset correction to get timer period. This is shown in the Figure 10.

Figure 10. Getting timer period



The three unknown parameters here are Kp, Ki and the function $f(x)$. Proportional constant Kp and integral constant Ki needs to be determined based on the requirements of maximum overshoot tolerance, settling time, and steady state error. The function $f(x)$ involves scaling and offset correction. This is done to keep the generated timer period value within the limits for all values of motor speed.

$$Timer\ Period = f(x) = \frac{(KSpeed \times Prescalar)}{x}$$

The terms KSpeed and Prescalar bring the value x from the PI controller output to map within the limits of timer period.

In the project, KSpeed is set to 0xFFFF and Prescalar is set to 32. Also the PI constants are defined in such a way that the ratio x/Prescalar always lie between 2 to 250 for complete range of motor speed. In the project, x/Prescalar is named as SpeedCount.

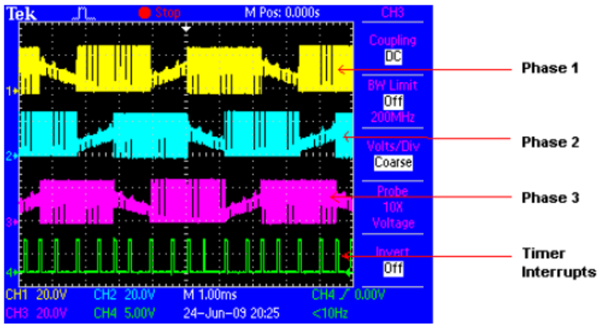
Timer input frequency can now be calculated from this assumed value of SpeedCount and timer interrupt rate at maximum speed of motor. You can refer the excel sheet provided with this application note, for calculating timer input frequency for different motor speed ratings.

Test Results

Commutation Times

Figure 11 shows the zero crossing detected by the firmware. As it is shown in the figure, the timer provides the interrupts when terminal voltage is around VBUS/2 that is, 12 V.

Figure 11. Zero crossing detected by the firmware



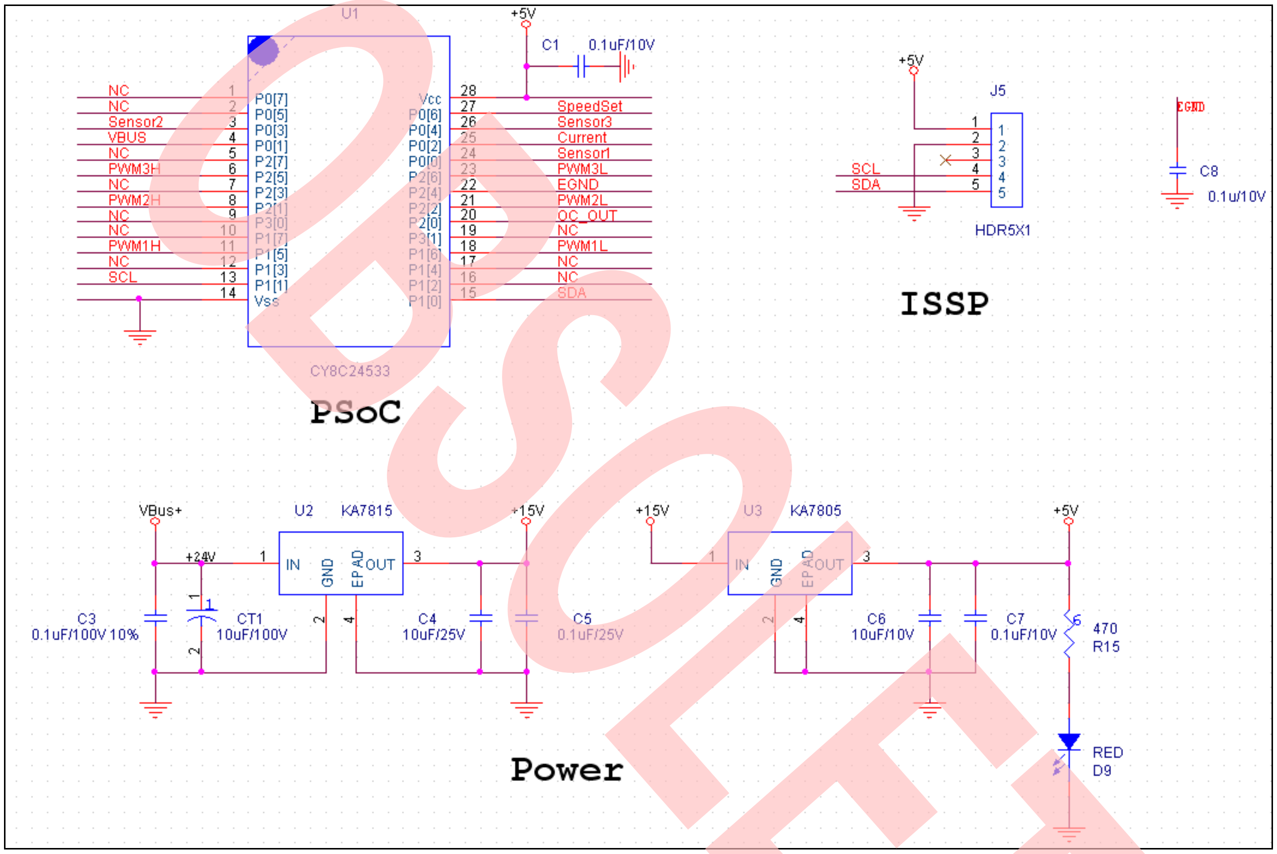
Summary

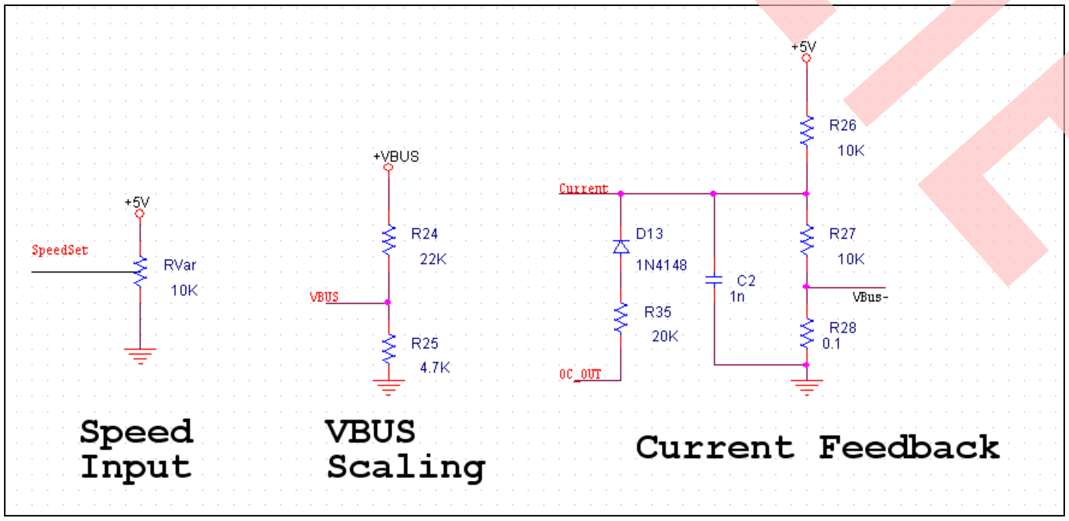
This application note demonstrates the unique way to implement Sensorless BLDC Controller. The solution has lesser external components, thus improving the reliability and decrease in cost of the whole solution.

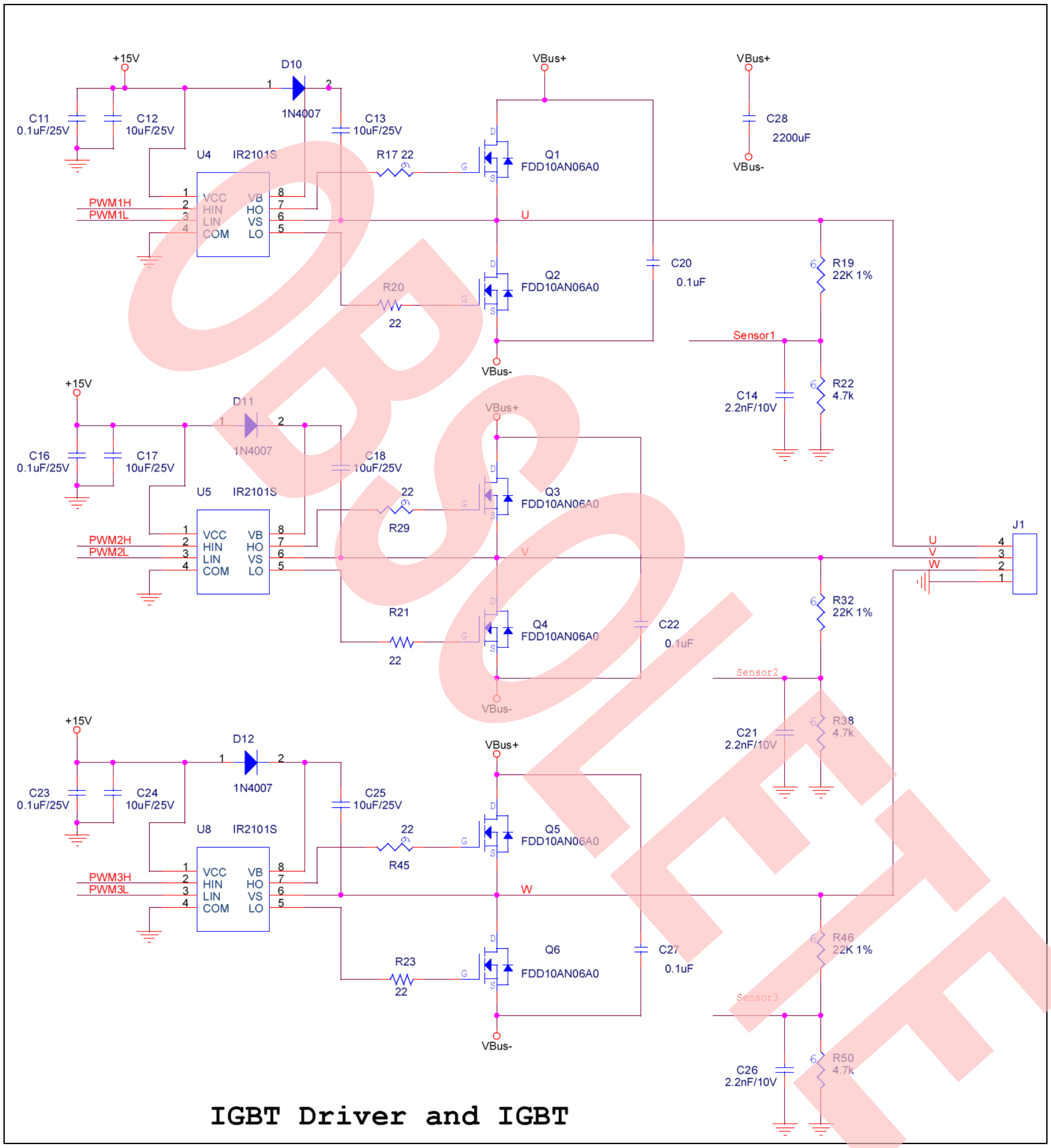
About the Author

Name:	Rajiv Badiger
Title:	Sr. Applications Engineer
Background:	B.E. Electronics and Communication, GHRCE, Nagpur, India
Contact:	rjvb@cypress.com
	+91 - 80 - 67073437

Appendix A: Schematics






+15V
D10
1N4007
C11
0.1uF/25V
C12
10uF/25V
C13
10uF/25V
U4
IR2101S
VCC
HIN
LIN
COM
VB
HO
VS
LO
PWM1H
PWM1L
R17 22
R20
22
VBus+
Q1
FDD10AN06A0
Q2
FDD10AN06A0
U
C20
0.1uF
VBus-
VBus+
C28
2200uF
VBus-
R19
22K 1%
Sensor1
C14
2.2nF/10V
R22
4.7k
D11
C16
0.1uF/25V
C17
10uF/25V
C18
10uF/25V
U5
IR2101S
PWM2H
PWM2L
R29
22
R21
22
Q3
FDD10AN06A0
Q4
FDD10AN06A0
V
C22
0.1uF
J1
W
R32
22K 1%
Sensor2
C21
2.2nF/10V
R38
4.7k
D12
C23
0.1uF/25V
C24
10uF/25V
C25
10uF/25V
U8
IR2101S
PWM3H
PWM3L
R45
22
R23
22
Q5
FDD10AN06A0
Q6
FDD10AN06A0
C27
0.1uF
R46
22K 1%
Sensor3
C26
2.2nF/10V
R50
4.7k
IGBT Driver and IGBT

Document History

Document Title: Sensorless BLDC Motor Controller Implementation Using PSoC® - AN56541

Document Number: 001-56541

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2769278	RJVB	09/25/09	New Spec.
*A	3772742	RJVB	10/10/2012	Updated in new template.
*B	4821795	RLIU	07/02/2015	Obsolete.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone	: 408-943-2600
Fax	: 408-943-4730
Website	: www.cypress.com