

3-phase Sensorless BLDC Motor Control Kit with the S12 MagniV MC9S12ZVM

by: Branislav Zigmund, Petr Staszko, and Anita Maliverney

1 Introduction

This application note describes the design of a 3-phase brushless DC (BLDC) motor control drive using a sensorless algorithm. The design is targeted at automotive applications. This cost-effective solution is based on the MC9S12ZVML128 chip, which is dedicated to automotive motor control.

The design exhibits the suitability and advantages of the MC9S12ZVML128 microcontroller for motor control. It serves as an example of a BLDC control design using Freescale's 16-bit S12 MagniV mixed-signal MCUs.

2 System Concept

The system is designed to drive a 3-phase BLDC motor. The application meets the following system concept:

- Targets the MC9S12ZVML128 microcontroller.

Contents

1	Introduction	1
2	System Concept	1
3	BLDC Sensorless Control	2
3.1	Brief overview of brushless DC motors	2
3.2	Complementary/independent unipolar PWM modulation technique	6
3.3	Position estimation based on BEMF zero-crossing detection	7
3.4	DC bus current measurement	11
3.5	States of BLDC sensorless control based on BEMF zero-crossing detection	12
4	Software implementation on the MC9S12ZVML128	13
4.1	MC9S12ZVM configuration	13
4.2	Software architecture	18
5	FreeMASTER user interface	29
6	Conclusion	29
7	References	29

- Runs on the hardware of the 3-phase Sensorless BLDC Motor Control Kit with the S12 MagniV MC9S12ZVM (for further information, see document number MTRCKTSBNZVM128). Details about the MC9S12ZVML128 evaluation board can be found in MC9S12ZVM128MCBUG, *MC9S12ZVML128 Evaluation Board User's Manual*.
- Incorporates a control technique with:
 - Six-step commutation control of a 3-phase BLDC motor
 - Position sensing by means of BEMF zero-crossing detection technique
 - Closed-loop speed control
 - Motor current limitation
 - Alignment at startup
- FreeMASTER software control interface (motor start/stop, speed setup)
- DC bus over-voltage, under-voltage, over-current, and overload protection

3 BLDC Sensorless Control

3.1 Brief overview of brushless DC motors

The BLDC motor is a rotating electric machine with a classic 3-phase stator similar to an induction motor. The phases mounted on the stator are connected to form a wye or delta connection. The rotor has surface-mounted permanent magnets. The motor can have more than one pole pair per phase. The pole pair per phase defines the ratio between the electrical revolution and the mechanical revolution.

The BLDC motor is equivalent to an inverted DC brushed motor, where the magnet rotates while the conductors remain stationary. In the DC brushed motor, the commutator and brushes reverse the current polarity in such a way that stator and rotor magnetic fields are perpendicular. However, in the brushless DC motor, a power transistor (which must be switched in synchronization with the rotor position) performs the polarity reversal. This process is also known as electronic commutation.

The displacement of the magnets on the rotor creates a trapezoidal back electromotive force (BEMF) shape when the rotor is spinning. Neglecting the higher-order harmonic terms, the BEMF in the motor phase (e_a, e_b, e_c) is as indicated in [Figure 1](#). Each BEMF has a constant amplitude for 120 electrical degrees, followed by a 60 electrical degree transition in each half-cycle. The developed torque at any instance is given by the following equation:¹

Eqn. 1

$$T = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega_{el}}$$

Where:

- ω_{el} is electrical angular velocity

The ideal current waveforms in each phase (i_a, i_b, i_c) need to be quasi-square waveforms of 120 electrical degrees of conduction angle in each half-cycle. The conduction of current in each phase must coincide with

1. Rashid, Muhammad H., ed. *Power Electronics Handbook*. 2nd ed. London: Academic Press, 2007.

the flat part of the BEMF waveforms; this guarantees that the developed torque is constant or ripple-free at all times. In order to align current conduction in each phase with the flat part of the BEMF, the rotor position must be known.

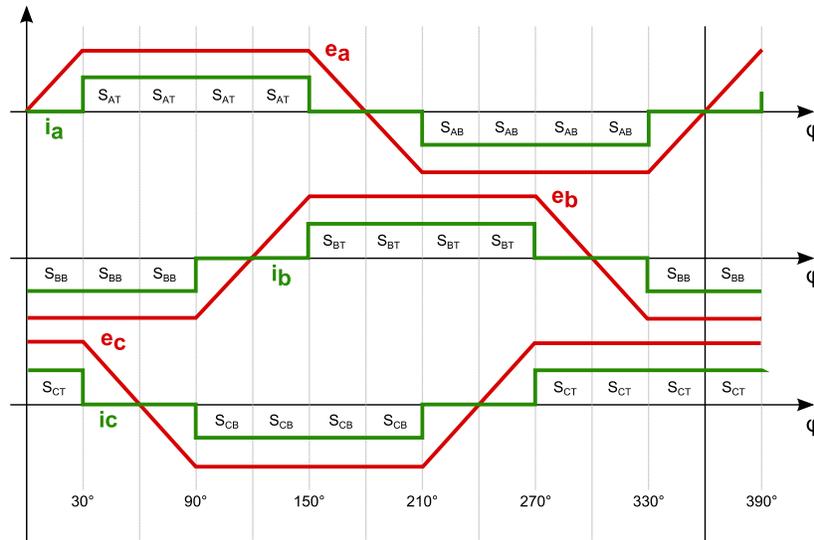


Figure 1. 3-phase BEMF voltages and phase currents of a BLDC motor

The position of the rotor can be obtained by a position sensor or a sensorless algorithm. Various kinds of position sensors are used; however, since the rotor is a permanent magnet, it is a very simple matter to determine where the physical pole edges are using a simple, reliable, and inexpensive Hall effect sensor.¹

The following techniques are commonly used to estimate rotor position in applications that rely on sensorless control of a BLDC motor:

- BEMF zero-crossing detection method
- Flux level detection method
- Various kinds of system state observers
- Signal injection methods

From a control perspective, two logical mechanisms must be employed:

- *Commutation control*, where the phases are energized according to rotor position with the quasi-square current waveforms.
- *Speed/torque control*, where the amplitude of the quasi-square current waveform applied to the phases is controlled to achieve the desired speed/torque performance.

The following sections discuss the concept of the BEMF zero-crossing detection method, as well as the methods and conditions for its correct evaluation.

¹.Ibid.

3.1.1 Electronic commutation control

The commutation process provides a mechanism to energize phases according to the rotor position with the quasi-square current waveforms. Since only six discrete outputs per electrical cycle are required (as shown in Figure 1), six semiconductor power switches are sufficient to create quasi-square current waveforms for the phases. Six semiconductor power switches form a 3-phase power inverter, designed using IGBT or MOSFET switches. The power for the system is provided by the DC-Bus voltage U_{DCB} . The semiconductor switches and diodes are modeled as ideal devices in Figure 2.

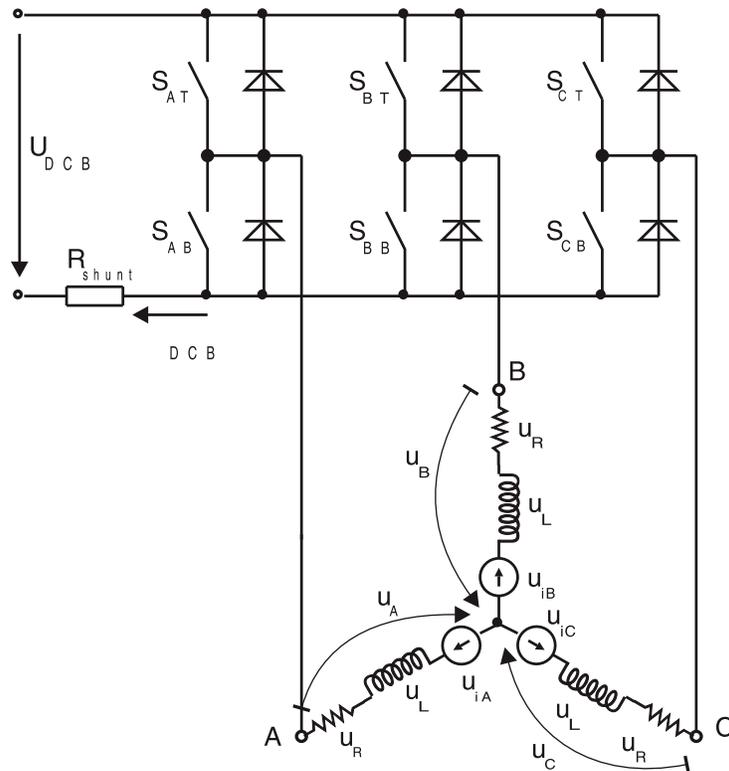


Figure 2. Power stage and motor topology

Six-step commutation is a very common method for driving a 3-phase way-connected BLDC motor. In six step commutation control, the BLDC motor is operated in a two-phase model. Two phases are energized while the third phase is disconnected as the space between the magnet poles passes over it and produces a zero BEMF voltage. Selection of the two energized phases is carried out by a position sensor or a position observer. Table 1 shows the output current waveforms for a 3-phase inverter and the switching devices that conduct during the six switching intervals per cycle.

Table 1. Six step switching sequence

Rotor position	Sector number	Switch closed		Phase current		
				A	B	C
330°–30°	1	S _{AT}	S _{BB}	+	–	off
30°–90°	2	S _{AT}	S _{CB}	+	off	–
90°–150°	3	S _{BT}	S _{CB}	off	+	–
150°–210°	4	S _{BT}	S _{AB}	–	+	off
210°–270°	5	S _{CT}	S _{AB}	–	off	+
270°–330°	6	S _{CT}	S _{BB}	off	–	+

3.1.2 Speed/torque control

Commutation ensures the proper direction of the phase current according to the rotor position of the BLDC motor, while the motor torque/speed only depends on the amplitude of the quasi-square current waveform. Continued control of the amplitude of the quasi-square current waveform for each phase of the motor is ensured by hysteresis or PWM control.

PWM control is commonly used in applications where microcontrollers are employed. The duty-cycle for the PWM modulator is obtained by the speed PI controller. The speed PI controller amplifies the error between the required and actual speeds, and its output, appropriately scaled, is assigned to the PWM modulator.

The actual mechanical speed can be calculated as a time derivative of the shaft position φ_{mech} .

Eqn. 2

$$\omega_{mech} = \frac{d\varphi_{mech}}{dt} = \frac{1}{p} \frac{d\varphi_{el}}{dt} \approx \frac{1}{p} \frac{\Delta\varphi_{el}}{\Delta T}$$

Since the shaft travels exactly 1/6th of one electrical revolution (2π in radians) between two commutations, the above equation can be rewritten to the following form:

Eqn. 3

$$\begin{aligned} \omega_{mech} &\approx \frac{1}{p} \frac{\Delta\varphi_{el}}{\Delta T} = \frac{1}{p} \frac{360^\circ}{T_{CM}} \\ &= \frac{1}{p} \frac{360^\circ}{T_{(330^\circ \rightarrow 30^\circ)} + T_{(30^\circ \rightarrow 90^\circ)} + T_{(90^\circ \rightarrow 150^\circ)} + T_{(150^\circ \rightarrow 210^\circ)} + T_{(210^\circ \rightarrow 270^\circ)} + T_{(270^\circ \rightarrow 330^\circ)}} \\ &= \frac{360^\circ}{p \sum_{n=1}^6 T_{CM}^n} \end{aligned}$$

Where:

- p is the number of pole-pairs
- T_{CM} is the time between two following commutations
- T_{CM}^n is the time between commutations in sector $n = 1, 2, 3, 4, 5, 6$
- φ_{el} is the electrical position

3.2 Complementary/independent unipolar PWM modulation technique

There are different methodologies for powering and switching the phases. The unipolar PWM control technique combines commutation control and torque control. While the state of the switches is determined by commutation control, the torque is controlled by the applied duty cycle. An application with BLDC control where the unipolar PWM control technique is employed benefits from a reduction in the MOSFET switching losses and an improvement in the system's EMC robustness.

The unipolar PWM control means that the motor phase sees only the positive polarity of the voltage. To achieve the unipolar PWM pattern, one phase is in complementary PWM mode while the second phase is grounded and the third phase stays unpowered, as shown in Figure 3. This PWM pattern can be seen every 60° electrical degrees, and they differ only in phase order. The phase order is determined according to the shaft position by commutation control.

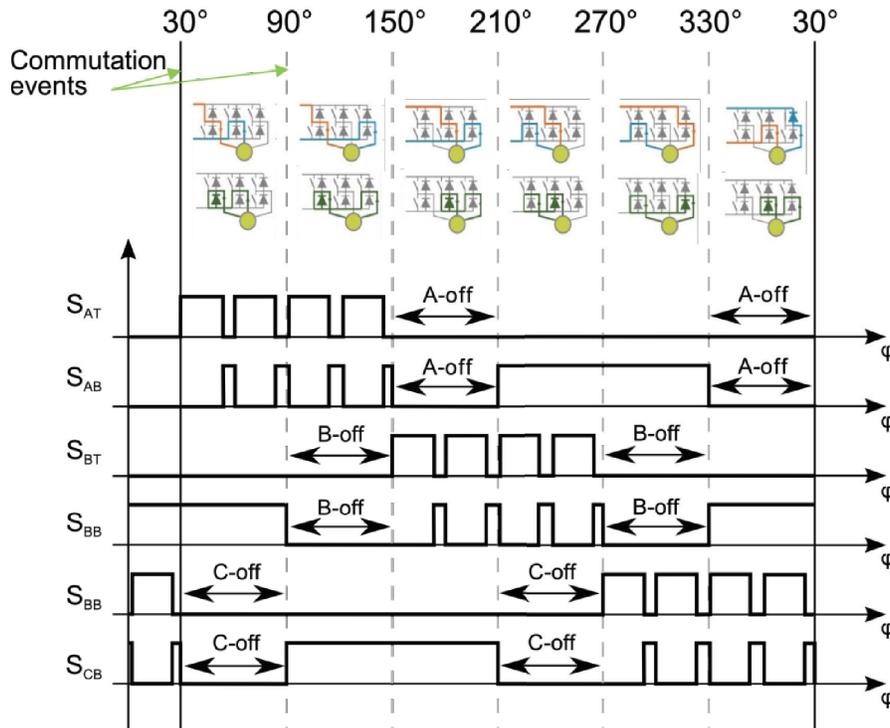


Figure 3. Complementary/independent unipolar PWM switching

For example, in the first cycle Phase A is powered by the complementary PWM signal while the bottom transistor of Phase B is grounded and Phase C is unpowered. After the commutation event at 90° electrical degrees, Phase A is still powered by the complementary PWM signal, Phase B is unpowered, and Phase C becomes grounded instead.

The control described in this application note is based on the complementary/independent unipolar PWM modulation technique.

The following section explains sensorless position estimation by means of BEMF zero-crossing detection for commutation control purposes.

3.3 Position estimation based on BEMF zero-crossing detection

Figure 4 shows ideal BEMF waveforms and depicts a commutation event occurring at a position of 30 electrical degrees after the point where a BEMF zero-crossing arises. The BEMF zero-crossing happens at a position of 30 electrical degrees after the point of the last commutation event. Let us assume that the motor is spinning at a constant velocity; in this case, the motor needs the same amount of time to travel from the position of the last commutation event to a BEMF zero-crossing and from the BEMF zero-crossing to the following commutation event. In the time domain, a BEMF zero-crossing is right in the middle of two commutation events. Therefore, the BEMF zero-crossing event, with help of a timer, can simply be used to estimate the right commutation point as well as the velocity of the rotor.

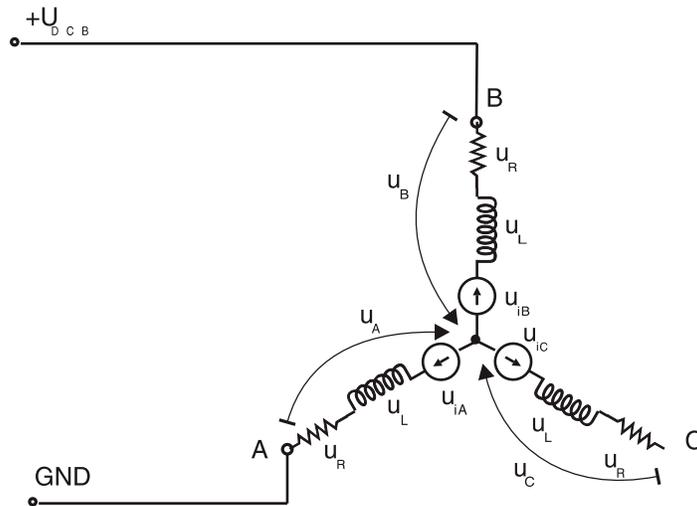


Figure 4. Zero-crossing detection and commutation diagram

3.3.1 BEMF zero-crossing principle

To explain and simulate the idea of BEMF sensing techniques, this document provides a simplified mathematical model based on the basic circuit topology (see Figure 4). The goal of the mathematical model is to identify dependencies between the measurable motor waveforms and a BEMF zero-crossing. The BEMF zero-crossing in turn helps to identify the commutation event.

BLDC Sensorless Control

The mathematical model is based on the fact that only two phases of a motor are energized and the third is disconnected. The natural voltage level of the whole model is referenced to half of the DC bus voltage, which simplifies the mathematical expressions. The mathematical model assumes that the motor phases are symmetrical (see [Figure 4](#)).

Eqn. 4

$$\left. \begin{aligned} u_N &= U_{DCB} - Ri_b - L \frac{di_b}{dt} - e_b \\ u_N &= Ri_a + L \frac{di_a}{dt} - e_a \end{aligned} \right\} \xrightarrow{i_a = i_b} u_N = \frac{U_{DCB}}{2} - \frac{e_b + e_a}{2}$$

For a symmetrical 3-phase motor, the sum of all BEMF voltages is zero; therefore:

Eqn. 5

$$e_c + e_b + e_a = 0 \rightarrow e_c = -(e_b + e_a)$$

The unpowered phase has the following voltage equation, since there is no current flowing:

Eqn. 6

$$u_N = u_C - e_c$$

By substituting [Equation 5](#) and [Equation 6](#) into [Equation 4](#), the phase voltage on the unpowered phase can be derived as:

Eqn. 7

$$u_C = \frac{u_{DCB}}{2} + \frac{3}{2}e_c$$

At the time of the BEMF zero-crossing, the BEMF voltage (e_c in this case) is zero as the name implies. Therefore by measuring voltage at the unpowered phase (e_c) and comparing it to half of the DC-Bus voltage ($\frac{U_{DCB}}{2}$), the BEMF zero-crossing can be accurately identified.

3.3.2 BEMF zero-crossing event detection, phase current measurement, and complementary/independent unipolar PWM switching

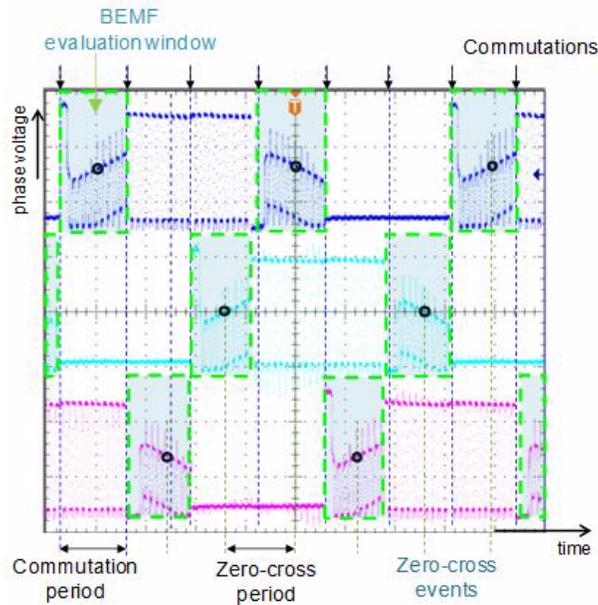


Figure 5. BEMF zero-crossing events, commutations, and their relationship to complementary/independent unipolar PWM switching

The exact position of the rotor can be sensed by measuring the BEMF voltage induced by the rotating permanent magnet in the unpowered phase.

In [Figure 5](#), the green windows mark the time periods in which the respective phase is unpowered. The voltage measured in this time window is the BEMF voltage. At the BEMF zero-crossing event, the permanent magnet is right in front of a coil and the rotor field is positioned 90° versus the stator field. This event happens in the middle of a commutation period and is marked as the black circles in the green BEMF window. At this time, the phase voltage is equal to half of the DC bus voltage as described in [Section 3.3.1](#), “[BEMF zero-crossing principle](#).” In the case of a constant shaft velocity, the period between two following zero-crossing events is equal to the commutation period.

[Figure 6](#) zooms in closer to one of the PWM cycles. At the top of the figure is the PWM pattern where Phase A is controlled by PWM and Phase C is grounded for the entire PWM period. During the PWM On cycle, the top switch of Phase A is turned on and the bottom switch of Phase C is grounded. Current flows from the DC bus into Phase A, and back through Phase C and the DC bus shunt resistor. In this cycle, the waypoint of the motor shows a voltage level of $\frac{U_{DCB}}{2}$. The BEMF voltage in the unpowered phase changes relatively to $\frac{U_{DCB}}{2}$ in the positive and negative directions, which means that the zero-crossing is detectable when the phase voltage on the unpowered phase is equal to $\frac{U_{DCB}}{2}$. Also, the phase current is measurable on the DC bus shunt.

During the Off cycle of the PWM period, both the Phase A and Phase C bottom switches are on. Therefore, phase current circulates through Phase A, Phase B, and two bottom switches back. During this cycle, the phase current is unable to reach the DC bus shunt resistor and the phase current cannot be measured. The

BLDC Sensorless Control

star point of the motor as well is connected to ground and the zero-crossing cannot precisely be measured in that cycle.

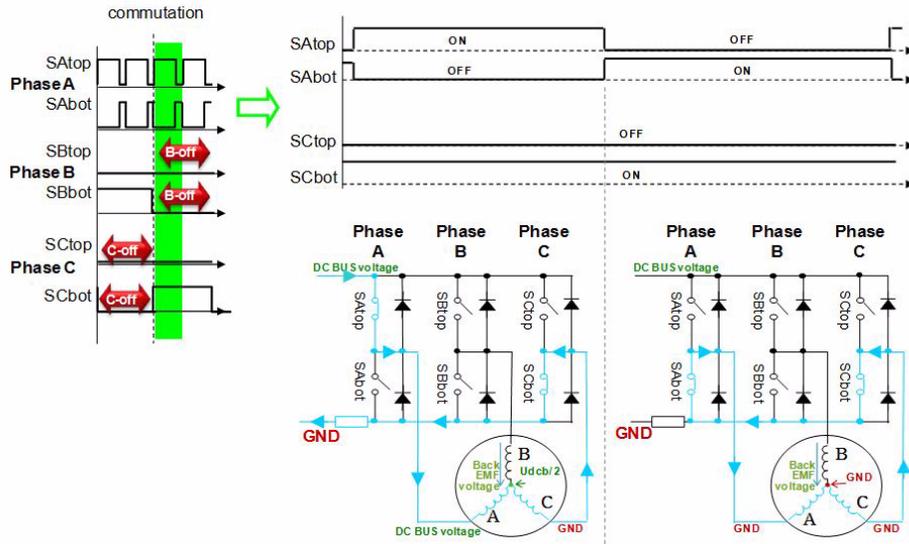


Figure 6. BEMF zero-crossing detection with complementary/independent unipolar PWM switching

Following on from the discussion above, phase current and BEMF voltage measurements must be performed in the active phase of the PWM cycle.

Moreover, right after the commutation event there is a commutation transient pulse measured on the phase voltages, which is produced by the current recirculation when the flyback diodes are conducting. The duration of this transient is dependent on the motor inductance, phase current, and speed. The BEMF voltage should not be measured during this time. The software must take care of this interval.

3.3.3 BEMF voltage measurement

As we learned earlier, the BEMF voltage can only be measured during the active phase of the PWM. Importantly, this is measured towards the end of the active cycle due to switching noises. In [Figure 7](#), the green marked area shows the window in which the BEMF should be measured.

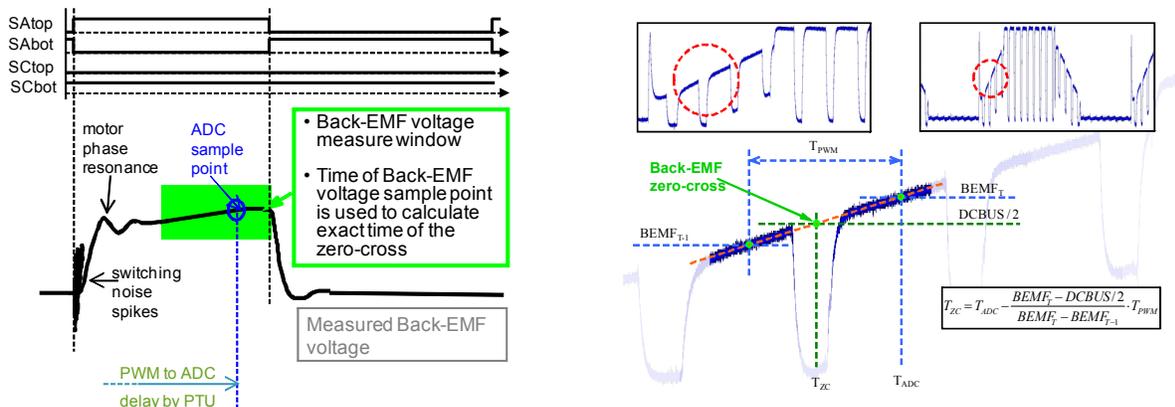


Figure 7. BEMF measurement

Figure 8. Precise BEMF zero-crossing identification

It should be noted that, depending on the motor and power stage parameters, the amplitude, period, and damping of the voltage ringing vary. As a result, it is recommended that you measure BEMF voltage close to the end of the window. The time of this sample point also needs to be stored, as it is used to enhance zero-crossing detection.

If we zoom out again and look at the BEMF voltage cycles (see Figure 8), it can be seen that the crossing of the BEMF voltage and $\frac{U_{DCB}}{2}$ level can take place wherever between two following BEMF voltage measurements. For accurate position estimation, an exact zero-crossing point has to be identified. This exact zero-crossing point identification is done by an approximation based on the interpolation of two following BEMF measurements.

Assuming that the shaft is not accelerating, actual BEMF voltage was measured at time T_{ADC} with the voltage level of e_T , and the previous measurement was taken at the time of $T_{ADC} - T_{PWM}$ with the voltage level of e_{T-1} , then the equation to calculate the exact time of the zero-crossing event could be derived as follows:

Eqn. 8

$$\frac{e_T - e_{T-1}}{T_{PWM}} = \frac{e_T - U_{DCB}/2}{T_{ADC} - T_{ZC}} \Rightarrow T_{ZC} = T_{ADC} - \frac{e_T - U_{DCB}/2}{e_T - e_{T-1}} T_{PWM}$$

This formula is calculated in the commutation period when two following comparisons of the BEMF voltage to half of the DC bus have the opposite signs.

In order to enhance the accuracy of the zero-crossing event even further, the DC bus voltage and BEMF voltage need to be measured simultaneously.

3.4 DC bus current measurement

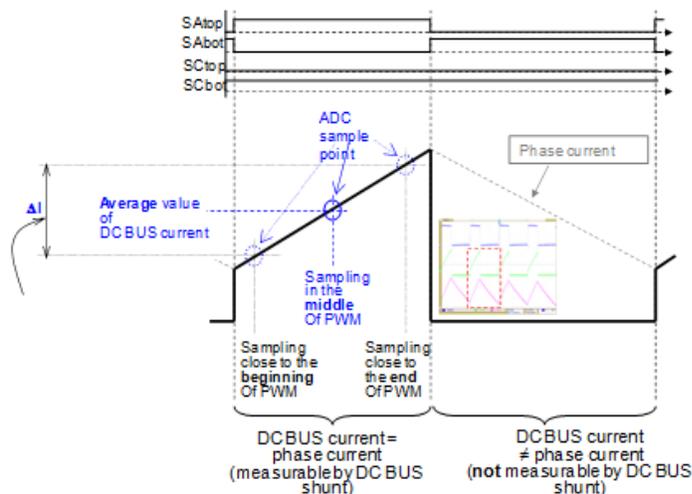


Figure 9. DC bus current measurement

As mentioned in the previous section, the DC bus current has to be measured in the active cycle of the PWM period due to the fact that the DC bus current equals the phase current only in the active cycle, as illustrated in [Figure 9](#).

During the active cycle of the PWM period, the phase current is rising. The slope of the rising current is defined by the motor phase coil inductance; the lower the phase inductance, the larger the gradient of the rising current. It is important to measure the current in the middle of the PWM active cycle to obtain the average phase current.

3.5 States of BLDC sensorless control based on BEMF zero-crossing detection

In order to start and run the BLDC motor, the control algorithm has to go through the following states:

- Alignment (initial position setting)
- Startup (forced commutation or open-loop mode)
- Run (sensorless running with BEMF acquisition and zero-crossing detection)

3.5.1 Alignment

As mentioned previously, the main task for sensorless control of a BLDC motor is position estimation. Before starting the motor, however, the rotor position is not known. The aim of the alignment state is to align the rotor to a known position. This known position enables starting the rotation of the shaft in the desired direction and generating the maximal torque during startup. During the alignment state, all three phases are powered in order to get the best performance behavior in either direction of shaft rotation. Phase A and Phase B are connected to the positive DC bus voltage and Phase C is grounded. The alignment time depends on the mechanical constant of the motor, including load, and also on the applied motor current. In this state, the motor current (torque) is controlled by the PI controller.

3.5.2 Startup

In the startup state, motor commutation is controlled in an open-loop mode without any rotor position feedback. The commutation period is controlled with an open-loop starting curve. The open-loop start is required only until the shaft speed is high enough (approximately 5% of nominal motor speed) to produce an identifiable BEMF voltage.

3.5.3 Run

The block diagram of the run state is represented by [Figure 10](#) and includes the BEMF acquisition with zero-crossing detection in order to control the commutations. The motor speed is estimated based on zero-crossing time periods. The difference between the demanded and estimated speeds is fed to the speed PI controller. The output of the speed PI controller is proportional to the voltage to be applied to the BLDC motor. The motor current is measured and filtered during the BEMF zero-crossing event and used as feedback into the current controller. The output of the current PI controller limits the output of the speed PI controller. The limitation of the speed PI controller output protects the motor current from exceeding the maximal allowed motor current.

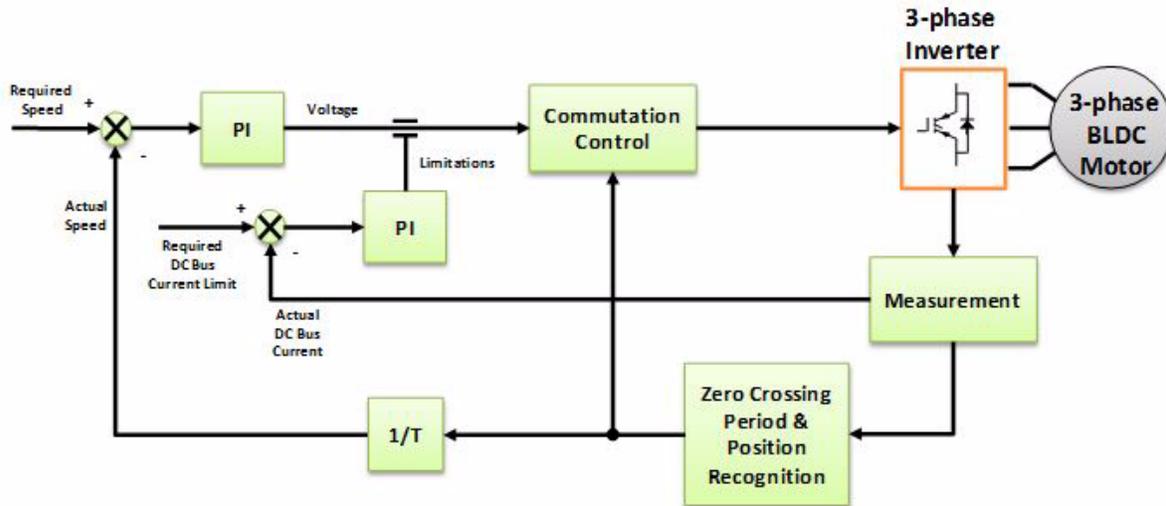


Figure 10. Speed control with current limitation

4 Software implementation on the MC9S12ZVML128

4.1 MC9S12ZVM configuration

The BLDC sensorless application framework is designed to meet the following technical specification:

- Targeted at the MC9S12ZVML128 evaluation board (for further information, see MC9S12ZVM128MCBUG, *MC9S12ZVML128 Evaluation Board User's Manual* and MTRCKTSBNZVM128QSG, *Quick Start Guide for 3-Phase Sensorless BLDC Kit with MagniV MC9S12ZVML128 MCUs*)
- PWM output frequency = 20 kHz
- Current loop sampling period 1 ms
- Speed loop sampling period 1 ms
- Linear approximation of the ADC samples for accurate BEMF zero-crossing detection. Internal dividers and multiplexer of the MC9S12ZVML128 are used for phase voltage measurement.
- DC bus voltage measurement at the HD pin by the ADC. The internal voltage dividers of the MC9S12ZVML128 are employed.
- Internal operational amplifier AMP0 is used for DC bus current measurement.

The MC9S12ZVML128 device includes modules such as the Pulse Width Modulator with Fault Protection (PMF), a Programmable Trigger Unit (PTU), an Analog-to-Digital Converter (ADC), a Timer module (TIM), and a Gate Drive Unit (GDU) suitable for control applications, in particular, motor control applications. These modules are directly interconnected and can be configured to meet various motor control application requirements. Figure 11 shows module interconnection for a typical BLDC sensorless application. The modules are described below and a detailed description can be found in MC9S12ZVMRMV1, *MC9S12ZVM-Family Reference Manual*.

4.1.1 Module interconnection

The modules involved in output actuation, data acquisition, and the synchronization of actuation and acquisition form the so-called *control loop*. This control loop consists of the PMF, GDU, ADC, and PTU modules. The control loop is very flexible in operation and can support static, dynamic, or asynchronous timing.

The PTU and ADC operate using lists stored in memory. These lists define the trigger points for the PTU, commands for the ADC, and results from the ADC.

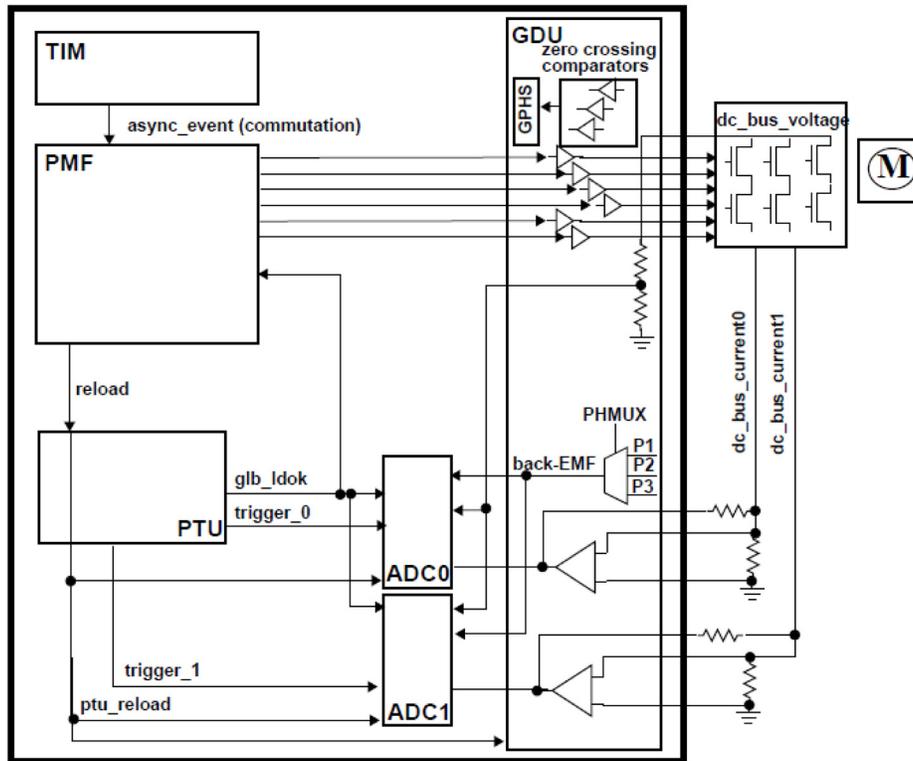


Figure 11. Block diagram of sensorless BLDC configuration

Each control loop cycle is started by a PMF reload event. The PMF reload event restarts the PTU timebase. If the PTU is enabled, the reload is immediately passed on as a `ptu_reload` event to the ADC and GDU modules.

The PMF generates the reload event at the required PWM reload frequency. The PMF reload event causes the PTU timebase to restart, to acquire the first trigger time from the list, and to generate the `ptu_reload` signal for the ADC_x to start loading the ADC conversion command from the Command Sequence List (CSL).

When the trigger time is encountered, the corresponding PTU trigger generates the `trigger_x` signal for the associated ADC. For simultaneous sampling, the PTU generates two simultaneous `trigger_x` signals, one for each ADC. At the `trigger_x` signal assertion, the ADC starts the first conversion of the next conversion sequence in the CSL (the first A/D command is already downloaded). For further information, see MC9S12ZVMRMV1, *MC9S12ZVM-Family Reference Manual*.

In this way, the PTU module serves as a delay block that can schedule several acquisitions of state variables relative to the start of the PWM period and within one PWM period.

4.1.2 Module involvement in BLDC sensorless software control loop

This section discusses measurements on the MC9S12ZVM128L and the internal hardware features to support the measurement of BEMF voltages and phase current.

Each commutation event first gets triggered by a timer interrupt. This interrupt can be defined as a PWM reload signal. The PWM reload signal itself also triggers the reload of the trigger list in the PTU module and restarts the PTU counter. When the PTU counter reaches the predefined value in the trigger list (T1 and T2), the PTU triggers an ADC measurement. At the time T1, measurement of the DC bus current is triggered. Two other simultaneous measurements are triggered at time T2, one for BEMF voltage and one for the DC bus voltage. The ADC conversion results are automatically stored into a predefined queue in memory. The principles of phase voltage, DC bus voltage, and DC bus current measurement are described in Section 3.3.3, “BEMF voltage measurement” and Section 3.4, “DC bus current measurement.”

The CPU is triggered by the ADC conversion complete interrupt service routine. Based on the stored ADC values, it calculates the zero-crossing event. Based on the zero-crossing period, the time of the next commutation event is calculated.

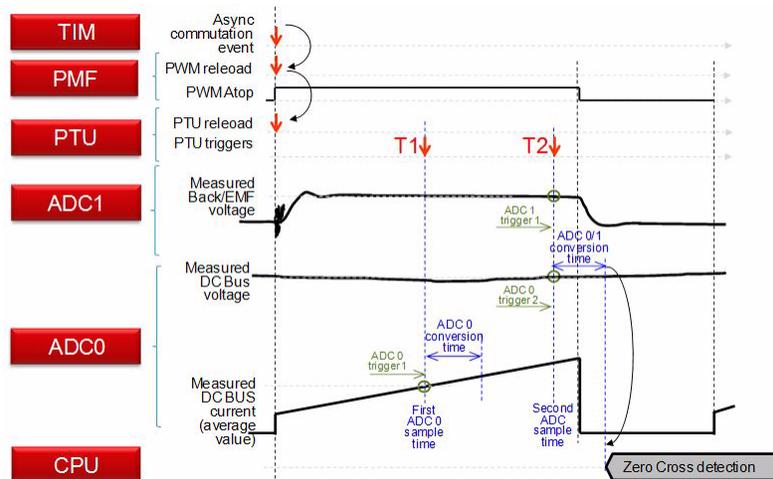


Figure 12. Module involvement in BLDC sensorless software control loop

The TIM, PTU, GDU, and ADC peripherals are based on the bus clock. To achieve a higher resolution of PWM, the PMF module is supplied by a core clock. The core clock is double that of the bus clock. The clock module uses a 4 MHz external crystal and is configured to generate a 12.5 MHz bus clock and 25 MHz core clock.

4.1.3 PMF

The Pulse Width Modulator with Fault Protection (PMF) module is configured to generate an edge-aligned (PMFCFG0_EDGE_{Ex} = 1) PWM with a frequency of 20 kHz (PMFMODA = 1250). In order to protect the MOSFET devices in the same leg of the inverter, dead time is set to approximately 0.5 μs

(PMFDTMA = 13). PWM generator A runs as the master and generates the Reload signal as a synchronization signal for the other submodules (PMFCFG2_REV[0:1] = 1). The Reload signal is generated at every PWM opportunity (PMFFQCA = 0). Pair A, Pair B, and Pair C PWMs are synchronized to PWM generator A (PMFCFG0_MTG = 0). All three pairs use value register zero (PMFCFG3_VLMODE = 1 – the value written into value register zero is also written into value registers 1 to 5) to set the duty-cycle. The PWM generator is restarted with every commutation event (PMFENCA_RSTRTA = 1).

To achieve a unipolar PWM pattern, one phase is in complementary PWM mode while the second phase is grounded by the software control mode (PMFOUTC, PMFOUTB) and the third phase stays unpowered by masking the output (PMFCFG2[MSK5:MSK0]) as shown in [Figure 3](#). The registers (PMFOUTC, PMFOUTB, PMFCFG2[MSK5:MSK0]) used to control the unipolar PWM pattern are in double-buffered mode (PMFCFG1_ENCE = 1). The double-buffered signals are swapped at a commutation event. A PWM pulse width PMFVAL registers are also double-buffered and are swapped when GLDOK is set and the PWM Reload signal occurs. The GLDOK is an external signal generated by the PTU module. The GLDOK is enabled at the PWM module (PMFENCA_GLDOKA = 1)

For example, in the first cycle, Phase A is powered by the complementary PWM signal, while the bottom transistor of Phase B is grounded (PMFOUTC = 0x0C, PMFOUTB = 0x2A) and Phase C is unpowered (PMFCFG2[MSK5:MSK0] = 0x34). After the commutation event at 90° electrical degrees, Phase A is still powered by the complementary PWM signal, Phase B is now unpowered (PMFCFG2[MSK5:MSK0] = 0x1C) and Phase C becomes grounded instead (PMFOUTC = 0x30, PMFOUTB = 0x2A).

4.1.4 PTU

The Programmable Trigger Unit (PTU) is intended to completely avoid CPU involvement in the time acquisitions of state variables during the control cycle.

The PTU module consists of two trigger generators (TG). For each TG, a separate enable bit is available so that both TGs can be enabled independently. TG0 is connected to ADC0, and TG1 is connected to ADC1. The trigger generation of the PTU module is synchronized to the incoming reload event. This reload event resets and restarts the internal time base counter and makes sure that the first trigger value from the actual trigger list is loaded. Furthermore, the corresponding ADC is informed that a new control cycle has started.

If the counter value matches the current trigger value, then a trigger event is generated. In this way, the reload event is delayed by the number of bus clock cycles defined by the current trigger value. All acquisition time values are stored inside the global memory map; that is, inside the system memory as a three dimensional array of integers (PTUTriggerEventList[][][]). The exact location of the acquisition time values (PTUTriggerEventList[][][]) in the system memory is given by the linker command file and linked to the PTU module during the initialization phase.

```
PTUPTRL = (uint8_t)((long)PTUTriggerEventList);
PTUPTRM = (uint8_t)((long)PTUTriggerEventList >> 8);
PTUPTRH = (uint8_t)((long)PTUTriggerEventList >> 16);
```

Each TG uses only one list to load the trigger values from the memory. The pointers for the primary (TG0L0IDX/ TG1L0IDX) and alternate (TG0L1IDX/ TG1L1IDX) lists are equal.

```
TG0L1IDX = (uint8_t) (((long)&PTUTriggerEventList[0][0][0] - (long)PTUTriggerEventList) >> 1);
TG1L0IDX = (uint8_t) (((long)&PTUTriggerEventList[1][0][0] - (long)PTUTriggerEventList) >> 1);
TG1L1IDX = (uint8_t) (((long)&PTUTriggerEventList[1][0][0] - (long)PTUTriggerEventList) >> 1);
```

The TG uses only one physical list of trigger events, even if the TG logic is switching between both pointers. At the end of the trigger list, the interrupt for TG1 is enabled (PTUIEL_TG1DIE = 1). This interrupt is used to store the time of the ADC measurement of the respective phase voltage. The PTU module generates the LDOK signal used to inform other modules that the double buffered registers were updated by software.

4.1.5 TIM

The Timer Module (TIM) is a basic scalable timer that consists of a 16-bit, software-programmable counter driven by a flexible programmable prescaler. The BLDC sensorless algorithm employs two timer channels in output compare mode (TIM0TIOS_IOS0 = 1; TIM0TIOS_IOS3 = 1).

Timer Channel 0 serves to identify the commutation event. The output compare signal is internally routed to the PMF module as an `async_event` in order to perform commutation of the PWM pairs. When the timer counter reaches the value in the channel registers of an output compare channel, the timer toggles (TIM0CTL2_OL0 = 1; TIM0CTL2_OM0 = 0) the channel output. This channel also schedules the interrupt (TIM0TIE_C0I = 1) where some of the application flags are controlled.

Timer Channel 3 is employed to control the torque of the motor by a software task. Timer Channel 3 schedules periodic interrupts (TIM0TIE_C3I = 1) with a period of 1 ms (TIM0TC3 = 781).

The timer's prescaler is equal to 4 (TIM0TSCR2_PR = 4) and all of the timers output compare pins are disconnected (TIM0OCPD = 0xff).

4.1.6 GDU

The Gate Drive Unit (GDU) is a Field Effect Transistor (FET) pre-driver designed for 3-phase motor control applications. The following GDU features are used in BLDC sensorless control:

- *Charge pump*: used to maintain the high-side driver gate source voltage VGS when PWM is running at a 100% duty cycle. The clock for the charge pump is set to be $\frac{f_{bus}}{32}$ (GDUCLK2_GCPCD = 2)
- *Desaturation error*: integrates three desaturation comparators for the low-side FET pre-drivers and three desaturation comparators for the high-side FET pre-drivers. The desaturation level is set to be 1.35 V (GDUDSLVL = 0x77) for both low-side and high-side FETs. A blanking time during the FET transients needs to be employed. The blanking time is set to be approximately 8 μ s (GDUCTR = 0x13).
- *Phase muxing*: used to select the phase voltage (GDUPHMUX) to be routed internally to ADC1 Channel 2.
- *Current sense amplifier*: Internal current sense amplifier 0 (GDUE_GCSE0 = 1) is used to measure motor phase current. The output of the current sense amplifier 0 is routed internally to ADC0 Channel 0.

4.1.7 ADC

The MC9S12ZVML128 uses two independent Analog-to-Digital Converters (ADC). Both ADCs are *n*-channel multiplexed input successive approximation analog-to-digital converters. The List Based Architecture (LBA) provides a flexible conversion sequence definition, as well as flexible oversampling. Both ADC conversion command lists are stored inside the global memory map; that is, inside the system memory as two dimensional arrays of bytes (ADC0CommandList[[]], ADC1CommandList[[]]). The exact location of the ADC conversion commands in the system memory is given by the linker command file and linked to the respective ADC module during the initialization phase. The same strategy is used for the ADC results. The conversion results are stored in an array of shorts (ADC0ResultList[], ADC1ResultList[]) located in system memory.

```

// ADC0 Command Base Pointer
ADC0CBP_0 = (uint8_t)((long)ADC0CommandList >> 16);
ADC0CBP_1 = (uint8_t)((long)ADC0CommandList >> 8);
ADC0CBP_2 = (uint8_t)(long)ADC0CommandList;

// ADC0 Result Base Pointer
ADC0RBP_0 = (uint8_t)((long)ADC0ResultList >> 16);
ADC0RBP_1 = (uint8_t)((long)ADC0ResultList >> 8);
ADC0RBP_2 = (uint8_t)(long)ADC0ResultList;

// ADC1 Command Base Pointer
ADC1CBP_0 = (uint8_t)((long)ADC1CommandList >> 16);
ADC1CBP_1 = (uint8_t)((long)ADC1CommandList >> 8);
ADC1CBP_2 = (uint8_t)(long)ADC1CommandList;

// ADC1 Result Base Pointer
ADC1RBP_0 = (uint8_t)((long)ADC1ResultList >> 16);
ADC1RBP_1 = (uint8_t)((long)ADC1ResultList >> 8);
ADC1RBP_2 = (uint8_t)(long)ADC1ResultList;

```

The ADC conversion clocks are set to be 6.25 MHz (ADC0TIM = 0; ADC1TIM = 0). The results are stored in memory as 12-bit (ADC0FMT_SRES = 4; ADC1FMT_SRES = 4) left-justified data (ADC0FMT_DJM = 0; ADC1FMT_DJM = 0).

Conversion flow of both ADCs is controlled by internal signals (generated by the PTU) and by the data bus (ADC0CTL_0_ACC_CFG = 3; ADC1CTL_0_ACC_CFG = 3). The results are stored in system memory even if commutation occurs when conversion is ongoing (ADC0CTL_0_STR_SEQA = 1; ADC1CTL_0_STR_SEQA = 1).

ADC0 schedules the end-of-list interrupt (ADC0CONIE_1_EOL_IE = 1) to calculate the BEMF zero-crossing algorithm.

The BLDC sensorless algorithm uses ADC0 to measure the motor phase current and DC bus voltage. The ADC1 is used to measure the respective motor phase voltage.

4.2 Software architecture

This section describes the software design of the BLDC sensorless algorithm based on zero-crossing. [Figure 13](#) shows the conceptual system block diagram.

The application is optimized for MC9S12ZVM motor control peripherals to achieve the least possible core involvement in state variable acquisition and output action application. The motor control peripherals (PMF, PTU, ADC, TIM, and GDU modules) are internally linked/set up together to work independently

from the core and to achieve deterministic sampling of analog quantities and precise commutation of the stator field. The software part of the application consists of different blocks which are described below. The entire application behavior is controlled from a PC with the FreeMASTER tool. For further information, see MTRCKTSBNZVM128QSG, *Quick Start Guide for 3-Phase Sensorless BLDC Kit with MagniV MC9S12ZVML128 MCUs*.

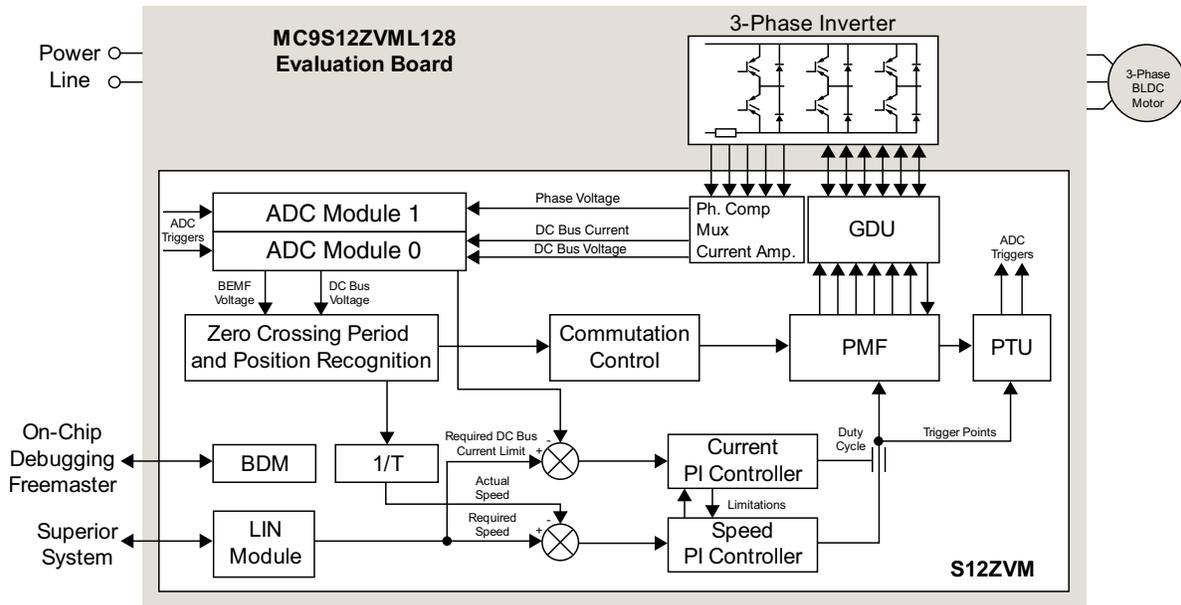


Figure 13. System block diagram

The block diagram shown in Figure 13 is the overview of the control blocks. The top box shows the power stage; the lower parts in the figure are the functions implemented in the S12ZVM. The power stage uses a one-shunt based system to measure the phase current.

The light green blocks are implemented in hardware on the S12ZVM device; the light blue blocks are implemented in software and benefit from the Motor Control Library (MCLib) (for further information, see MC9S12ZVMMCLUG, *Math and Motor Control Library for MC9S12ZVML128 User's Guide*). The inputs of the control loop are the measured voltages and current on the power stage, in particular the phase voltages, the DC bus current, and DC bus voltage.

The DC bus current is amplified by the current sense amplifier, which is part of the Gate Drive Unit (GDU), and then routed together with the DC bus voltage to one of the two ADCs for measurement acquisition. The phase voltages are multiplexed by the GDU and then used by the second ADC.

From a control perspective, the block diagram is divided into two logical parts:

- *Commutation control*, where the phase voltages and DC bus voltage are used to calculate the actual position of the shaft. According to the identified position, the next commutation event can be prepared.
- *Speed/torque control*, where the required shaft velocity is compared to the actual measured speed and regulated by the PI controller. The output of the speed PI controller is the duty-cycle. The duty-cycle is limited by the current PI controller and assigned to the PWM.

The application is controlled by FreeMASTER, a real-time debugging tool. FreeMASTER communicates with the S12ZVM device by means of BDM or an SCI peripheral.

4.2.1 Application flow

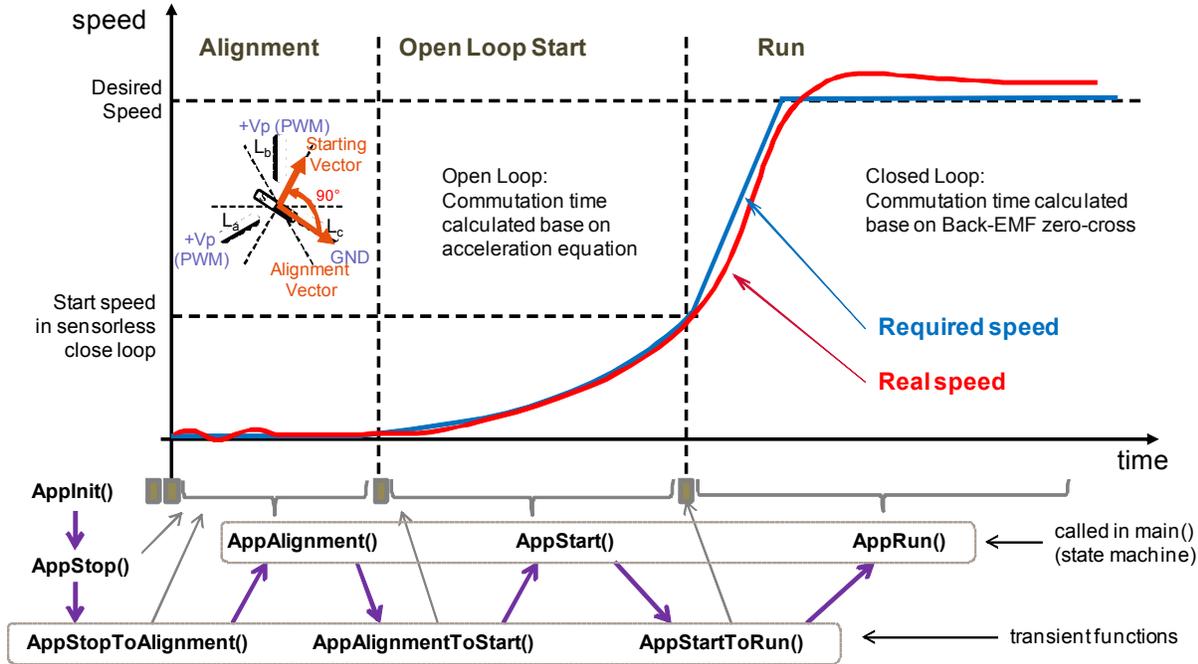


Figure 14. Application state flow

Figure 14 explains the different application states. The figure consists of two interconnected parts:

- The speed over time characteristic
- The blocks in the lower part of the picture, which show the states of the application and the transitions between respective states

The application software has three main states: the *alignment* state, the *open-loop start* state, and the *run* state. In the run state, the BLDC motor is fully controlled in closed-loop sensorless mode.

After the initialization of the peripheral modules has completed, the software enters the alignment state. In alignment, the rotor position is stabilized into a known position in order to create the same startup torque in both directions of rotation. This is achieved by applying a PWM signal to Phase A and Phase B. The duty cycle is calculated by the alignment PI controller. Phase C is assigned with a duty cycle equal to zero; that is, Phase C is connected to the negative pole of the DC bus. The value of the duty cycle on Phase A and Phase B depends on the motor inertia and load applied on the shaft. Such a technique aligns the shaft with Phase C, which is perpendicular to the two flux vectors generated by the stator windings, and therefore ensures the same startup torque in both directions of rotation. The duration of the alignment state depends on the motor’s electrical and mechanical constants, the applied current (meaning duty cycle), and the mechanical load.

When the alignment timeout expires, the application software moves to the open-loop start state. At a very low shaft velocity, the BEMF voltage is too low to reliably detect the zero-crossing. Therefore, the motor

has to be controlled in an open-loop mode for a certain time period. The very first vector generated by the stator windings needs to be set to a position 90° relative to the position of the flux vector generated by magnets mounted on the rotor. The alignment and first start-up vector are shown in Figure 14. The duration of the open-loop start state is defined by the number of open-loop commutations. The number of open-loop commutations depends on the mechanical time constant of the motor, including load, and also on the applied motor current. The shaft velocity after an open-loop startup is approximately 5% of nominal velocity. At a velocity approximately 5% of nominal velocity, the BEMF voltage is high enough to reliably detect the zero-crossing.

After a defined number of commutation cycles, the state changes from the open-loop start state to the run state. From here on, the commutation process based on the BEMF zero-crossing measurement takes place, and the control enters the closed-loop mode.

4.2.2 Application timing and interrupts

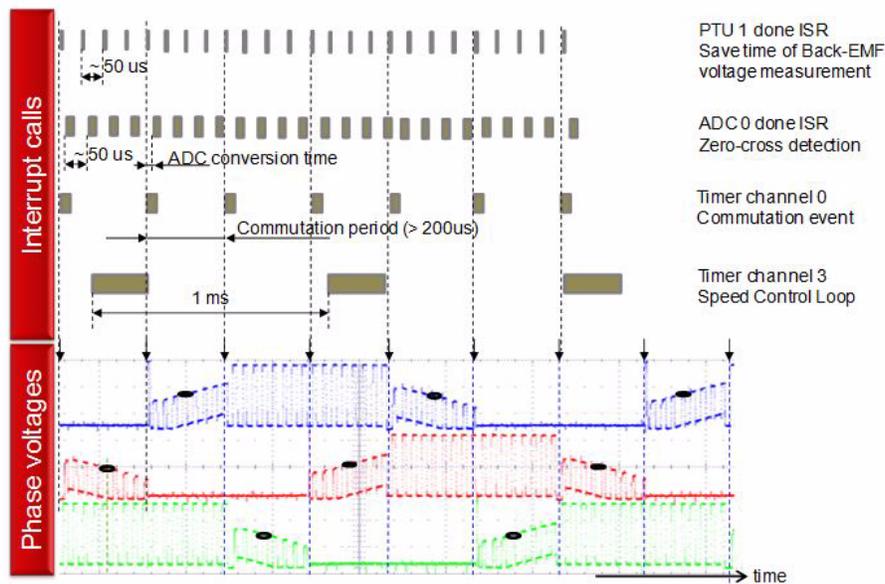


Figure 15. Application timing and interrupts

Figure 15 shows the application timing and the associated interrupts used for the commutation, zero-crossing and speed control. The grey boxes show the executed interrupt routines versus the phase voltage measurement.

The top row shows the interrupt that is activated when all the triggers in the PTU have been executed, which happens the moment the BEMF measurement is triggered by the Trigger Unit at the end of each active PWM cycle. In this interrupt, the timer value as a BEMF measurement reference point is saved.

The second row in the figure shows the interrupt that is activated when all the ATD measurements within a PWM cycle have been executed. At that time, all the necessary values for the zero-crossing detection have been stored automatically into memory by the ATD converter and the detection of the zero-crossing event and calculation of the zero-crossing period can then take place. After the zero-crossing detection, the

multiplexer for the phase voltage measurement is switched to the phase on which the BEMF will be sensed in the following commutation period.

The third row shows the interrupt that is the actual trigger for the commutation event, triggered by a timer channel interrupt. The time between each call is dependent on the actual speed of the motor.

The last row shows the interrupt routine used for the speed control loop. This is handled in a timer channel interrupt which is set every millisecond. The actual speed information, together with the variable of the required speed, is an input value to the speed controller.

4.2.3 Zero-crossing detection processing

4.2.3.1 Getting the time of the BEMF measurement

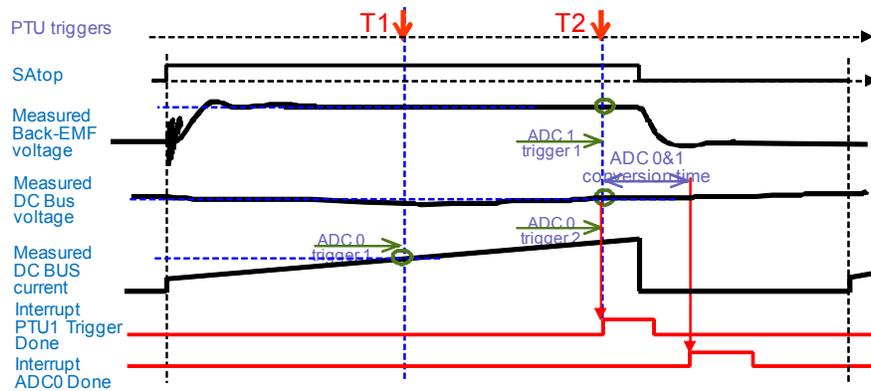


Figure 16. Measurement of DC bus current, voltage, and BEMF voltage

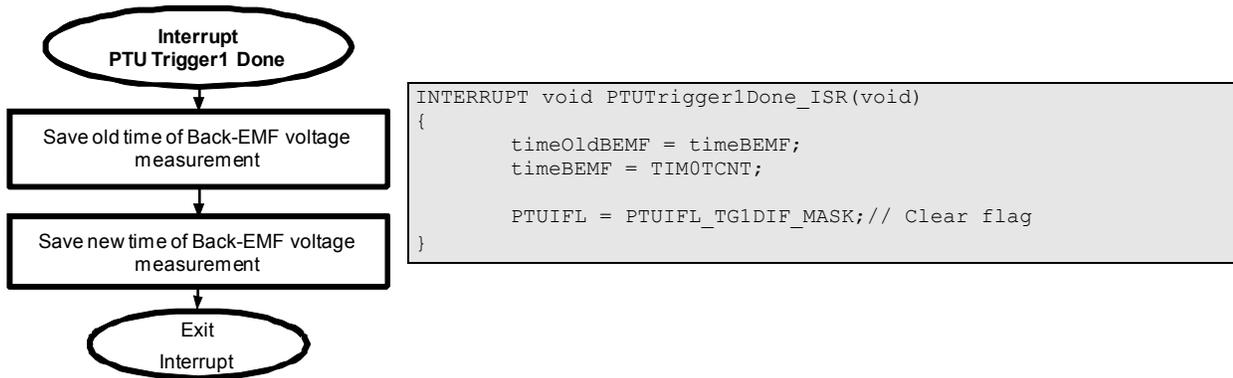


Figure 17. Obtaining the time of the BEMF measurement in the PTU Trigger Done ISR

The PTU Trigger Done interrupt service routine (ISR) is activated when all triggers in the PTU trigger list have been scheduled. The PTU, PTU trigger list, and ADC command lists are configured such that the PTU Trigger Done interrupt service routine is executed when the BEMF measurement takes place (Figure 16). In the ISR, the time of the BEMF measurement taken during the previous cycle is backed up and the counter value of the free running timer is stored, as shown in Figure 17. Both timestamps are needed for the calculation of the zero-crossing point.

4.2.3.2 State variable acquisition and zero-crossing detection processing

For state variable acquisition and zero-crossing detection processing, the ADC0 Done interrupt is used. The interrupt service routine is executed when the ADC0 and ADC1 reach the end of the ADC command list; measurements of the phase voltage and the DC bus voltage are the last commands in the ADC1 and ADC0 command lists, respectively. The phase voltage and the DC bus voltage are measured simultaneously on the ADC1 and ADC0. The flowchart and interrupt routine in Figure 18 describe the steps of processing the measurements.

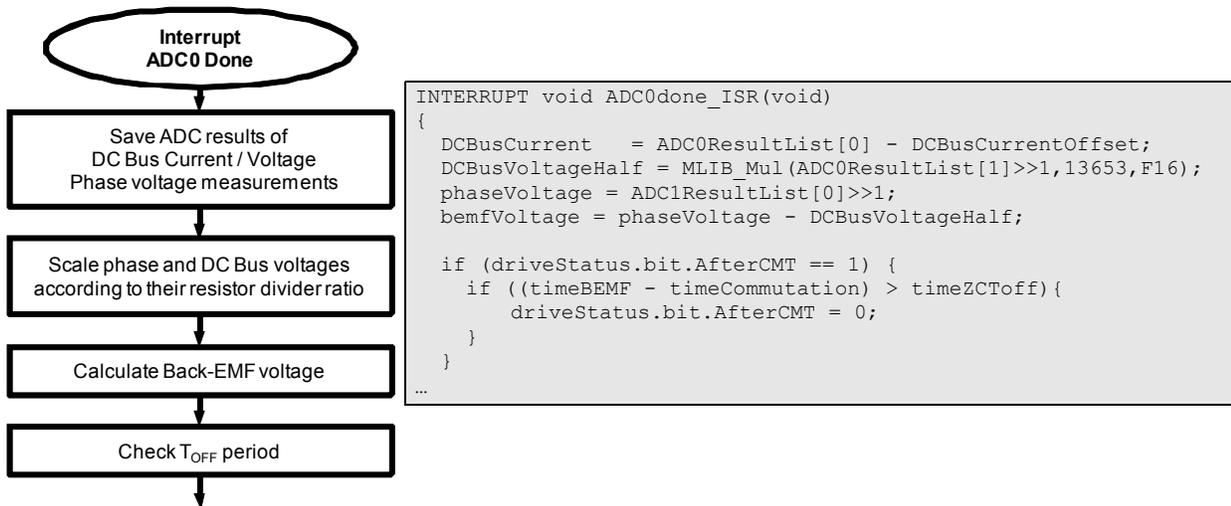


Figure 18. Processing measurements in the ADC Done ISR

Before the ADC0 Done ISR routine is executed, the ADC stores the results into predefined memory space (also known as the ADCx result list) by use of the DMA. The DC bus current and the DC bus voltage measurements are stored in the ADC0 result list and the phase voltage is stored in the ADC1 result list.

In the ADC0 Done ISR routine, the ADC results are scaled according to the hardware setup and saved into the appropriate variables. The BEMF voltage is calculated as the difference between the phase voltage and $\frac{U_{DCB}}{2}$. The BEMF voltage value is the signed number.

The software checks whether the commutation transient time has already passed (see Section 3.3.2, “BEMF zero-crossing event detection, phase current measurement, and complementary/independent unipolar PWM switching”). Commutation Transient time is called T_{off} (timeZCToff) in the software code implementation (Figure 19).

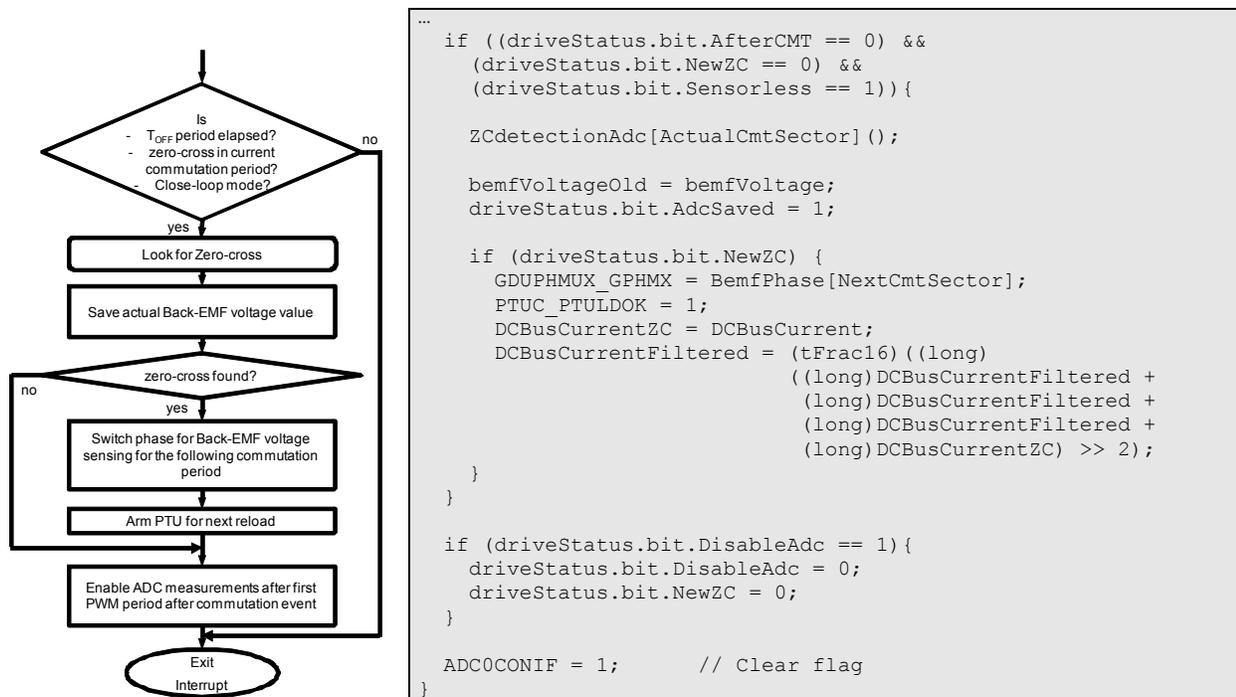


Figure 19. Commutation transients processing

Where the commutation transient time T_{Off} has not yet expired, the zero-crossing calculation will not be performed. The calculation will also not be performed if the zero-crossing point has already been identified in the current commutation period, or if the application is running in open-loop mode.

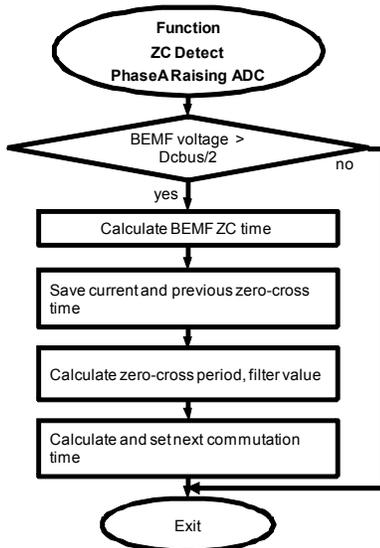
If the above mentioned conditions are not met, the zero-crossing detection routine will be executed. Based on the current commutation sector, a slightly different code execution will be performed, so therefore this sector number of 0 to 5 is passed to the algorithm.

When the zero-crossing position calculation is finished, the BEMF voltage value is stored as the old value as it will be referenced again in the next PWM cycle.

When a zero-crossing point is identified, the multiplexer that reads one of the three phase voltages is switched to the phase that is relevant for phase voltage reading in the next commutation cycle.

Afterwards, the `glb_ldok` of the PTU is set to be prepared for the next reload signal to change the phase multiplexer located in the GDU module.

The flow chart and the code listing in Figure 20 describe the zero-crossing detection routine that was called in the interrupt shown before. Depending on the actual detected phase and the active sector, a routine is called for either a rising or a falling BEMF voltage detection function. The example here shows the routine called in the case of a rising BEMF on Phase A.



```

void ZCdetectPhAraisingAdc(void)
{
    tFrac16 delta;

    if (bemfVoltage >= 0){
        // Raising approximation
        delta = bemfVoltage - bemfVoltageOld;
        if ((driveStatus.bit.AdcSaved == 1) &&
            (delta > bemfVoltage)) {
            timeBEMF -= MLIB_Mul(MLIB_Div(bemfVoltage, delta, F16),
                                timeBEMF - timeOldBEMF, F16);
        }
        else{
            // middle of previous ADC sensing events
            timeBEMF -= ((timeBEMF - timeOldBEMF) >> 1);
        }

        lastTimeZC = timeZC;
        timeZC = timeBEMF;

        periodZC_R_PhA = timeZC - lastTimeZC;
        actualPeriodZC = (actualPeriodZC + periodZC_R_PhA) >> 1;

        NextCmtPeriod = MLIB_Mul(actualPeriodZC,
                                advanceAngle, F16);
        TIM0TC0 = timeZC + NextCmtPeriod;

        driveStatus.bit.NewZC = 1;
    }
}
    
```

Figure 20. Zero-crossing detection routine

In the case of a negative BEMF voltage, the zero-crossing point has not been passed and the zero-crossing point is not detectable. The software exits the zero-crossing detecting routine and leaves the zero-crossing status bit as still unmarked. In the case of a positive BEMF voltage, the zero-crossing point was passed and Equation 8 is calculated, meaning that the BEMF voltage is divided by the delta of the two measured points and multiplied by the measured PWM period (BEMF measurement period). After this calculation, the old zero-crossing time and the new one are saved into the appropriate variables. The zero-crossing period is then calculated based on the calculated time of zero-crossing and the time of the zero-crossing in the previous commutation cycle. The zero-crossing period is also filtered to improve reliability.

At the end of the routine, the new commutation time is calculated. Here, some motor characteristics have to be taken into account. Instead of just adding half of a zero-crossing period to the actual zero-crossing time, a so-called advance angle factor is taken into account, which actually activates the commutation a bit earlier than calculated. This is usually a constant and depends on the motor characteristics.

Finally, the zero-crossing point is marked as found, and in that way the calculation does not take place anymore in current commutation cycle.

4.2.3.3 Speed evaluation and control

The speed controller in Figure 21 is executed in a timer interrupt every 1 ms. First of all, the actual speed is calculated from all of the last six zero-crossing periods, and this is stored in a scaled format as the actual speed.

The quality of the zero-crossing periods is evaluated in a function (StallCheck()) and compared against the min and max limits, with error stored if those limits are exceeded.

In the closed-loop mode, the speed error between the actual speed and the required speed is calculated. This calculated speed error is fed into the PI controller function. The PI controller is part of the Freescale Motor Control Library (for further information, see MC9S12ZVMMCLUG, *Math and Motor Control Library for MC9S12ZVML128 User's Guide*). Inputs to the PI controller function include the speed error and the PI controller's parameters such as the proportional and integral constants. The output of the PI controller is the duty-cycle, which is scaled to the PWM resolution.

At the end of the speed control function, the duty-cycle is assigned to the PMF module, and based on the duty-cycle, new trigger points are also calculated and updated in the PTU module.

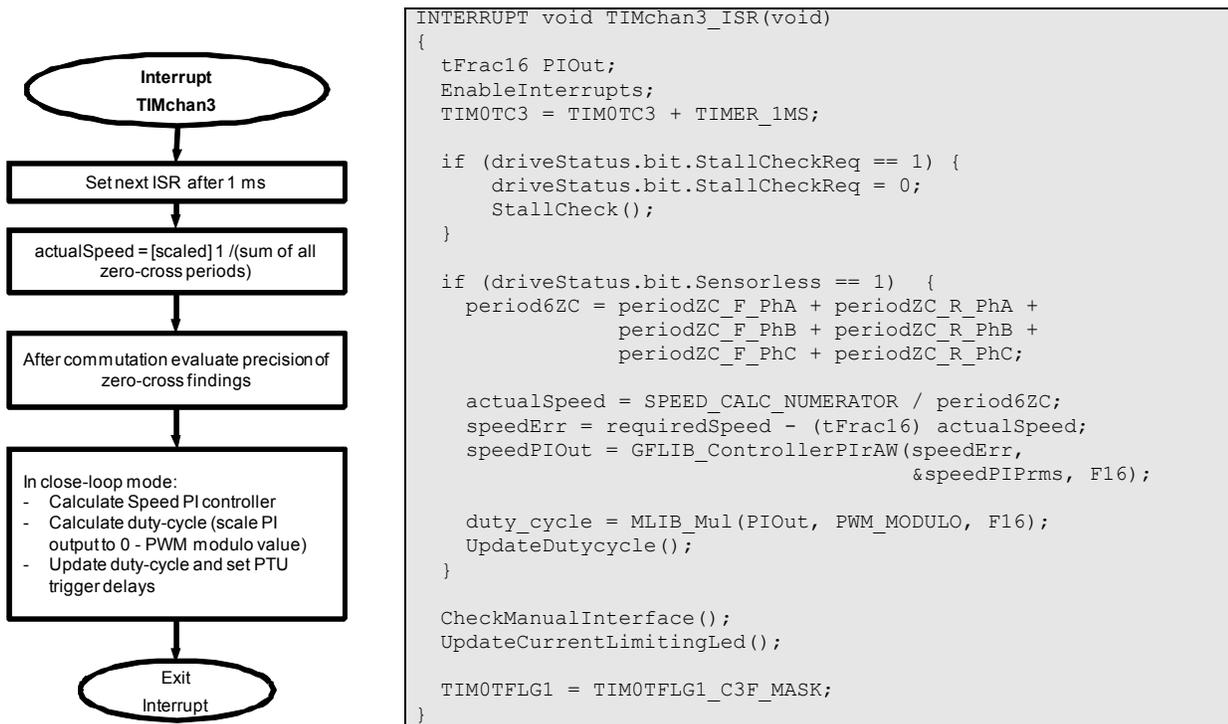


Figure 21. Speed evaluation ISR software flow

4.2.4 Current limitation controller

The current limit controller is located in the same 1 ms timer interrupt as the speed controller because the inputs and outputs of both controllers are linked together.

When the actual speed has been calculated, the current limit PI controller can be called by feeding it with the difference between actual current and the maximum allowed current of the motor. The output of the PI controller is scaled to the number proportional to the PWM period. After the current PI controller has calculated its duty cycle, both the duty cycle output values are compared to each other.

If the speed PI controller duty cycle output is higher than the current limit PI controller output, then the speed PI Controller duty cycle output value is limited to the value of the current limit PI controller; otherwise, the speed PI duty cycle output will be taken as the duty cycle update value. The value of the duty cycle will be used to update the PMF module and the trigger delay values in the PTU unit.

At the end, the integral portion values of both the PI controllers need to be synchronized to avoid one of the controllers increasing its internal value as far as the upper limit. If the duty cycle was limited to the current PI duty cycle output, then the integral portion of the current PI controller will be copied into the integral portion of the speed controller, and vice versa. The above described procedure is also described in [Figure 22](#).

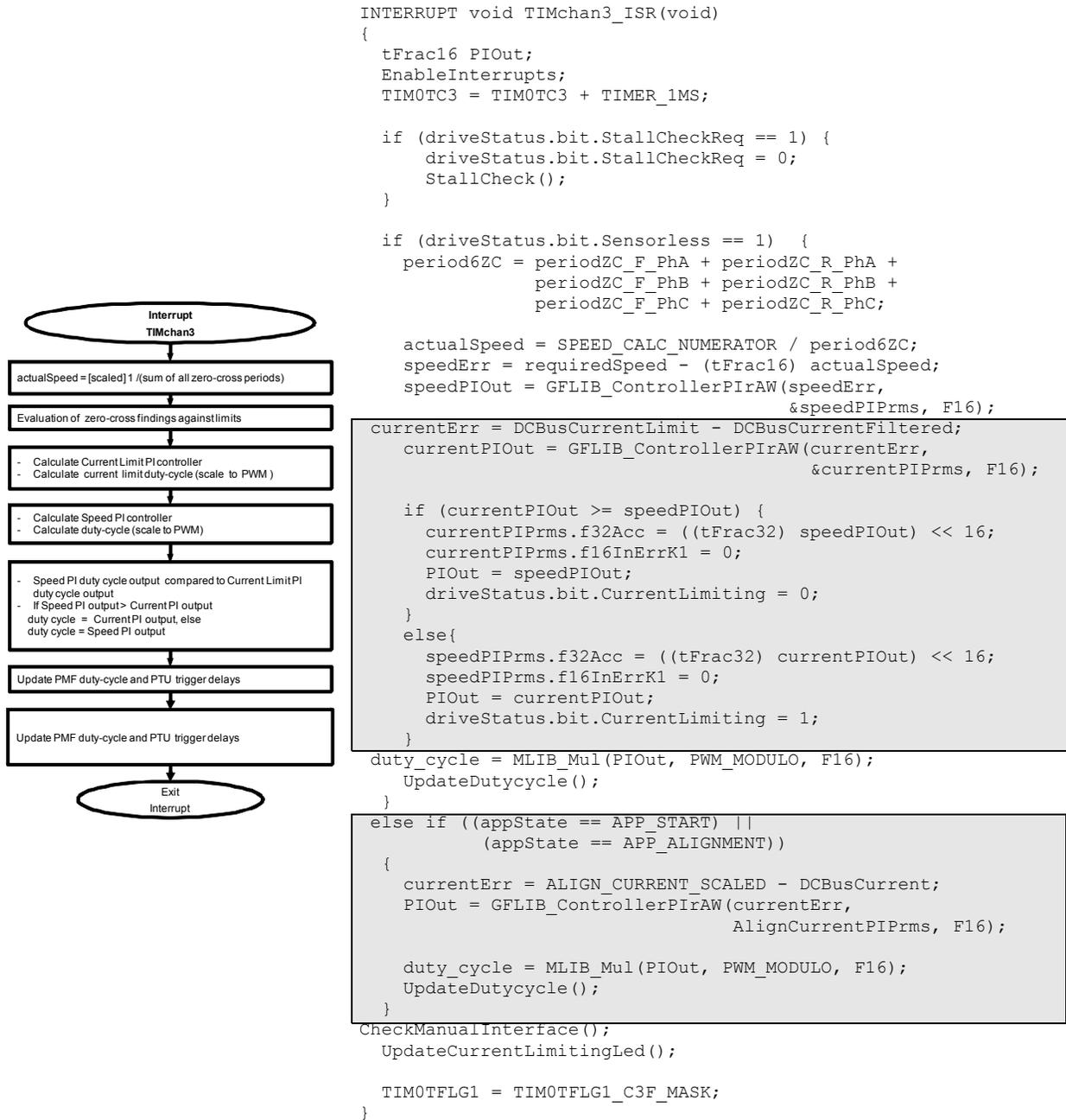


Figure 22. Speed evaluation and phase current limitation

4.2.5 MC Library

The application source code uses the Freescale Motor Control Library for the MC9S12ZVML128 microcontrollers (described in MC9S12ZVMMCLUG, *Math and Motor Control Library for MC9S12ZVML128 User's Guide*). The library contains three independent library blocks: GFLIB, GDFLIB, and GMCLIB. GFLIB includes basic mathematical functions (such as sine, cosine, ramp, and

so on). Advanced filter functions are part of the General Digital Filters Library, and standard motor control algorithms are part of the General Motor Control Library.

5 FreeMASTER user interface

The FreeMASTER debugging tool is used to control the application and monitor variables during runtime. (For further information, see freescale.com/FREEMASTER and document number MTRCKTSBNZVM128QSG, *Quick Start Guide for 3-Phase Sensorless BLDC Kit with MagniV MC9S12ZVML128 MCUs*). Communication with the host PC passes via USB. However, because FreeMASTER supports RS232 communication, there must be a driver for the physical USB interface, CP2102, installed on the host PC that creates a virtual COM port from the USB. The driver can be installed from silabs.com. The application configures the SCI module of the MC9S12ZVML128 for a communication speed of 9600 bit/s. Therefore, the FreeMASTER user interface also needs to be configured respectively.

6 Conclusion

The design described in this document shows the simplicity and efficiency of using the MC9S12ZVML128 microcontroller for sensorless BLDC motor control, and introduces it as an appropriate candidate for various low-cost applications in the automotive area.

7 References

Document number	Title	Availability
MC9S12ZVMRMV1	<i>MC9S12ZVM-Family Reference Manual</i>	freescale.com
MC9S12ZVM128MCBUG	<i>MC9S12ZVML128 Evaluation Board User's Manual</i>	
MTRCKTSBNZVM128	<i>3-phase Sensorless BLDC Motor Control Kit with the MagniV MC9S12ZVM</i>	freescale.com/AutoMCDevKits
MTRCKTSBNZVM128QSG	<i>Quick Start Guide for 3-Phase Sensorless BLDC Kit with MagniV MC9S12ZVML128 MCU</i>	
MC9S12ZVMMCLUG	<i>Math and Motor Control Library for MC9S12ZVML128 User's Guide</i>	freescale.com/AutoMCLib
—	<i>FREEMASTER Runtime Debugging Tool</i>	freescale.com/FREEMASTER

How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. MagniV is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number: AN4704

Rev. 1

08/2013

