AN13489

Position Sensorless Square-Wave Control Application of 3-phase BLDC Using KE17Z

Rev. 0 — 25 January 2022 Application Note

Contents

1 Introduction.....1
2 KE17Z features and advantages.....1
3 BLDC Sensorless Control theory.... 3
4 Hardware and software implementation.....5
5 Peripheral configurations..... 7
6 Software implementation.....11
7 References.....14
8 Revision history.....14

1 Introduction

Kinetis KE1xZ256 MCUs are the leading parts for the KE1xZ family based on Arm® Cortex®-M0+ core. Providing up to 256 KB flash, up to 48 KB RAM, and the complete set of analog/digital features, KE1xZ extends Kinetis E family to higher performance and broader scalability. Robust and enhanced TSIs provide high-level stability and accuracy to customer's HMI system. 1 Msps ADC and FlexTimer help build a perfect solution for BLDC motor control systems.

This application note describes the implementation of the sensorless Motor Control Reference Solution Package (MCRSP) software for a 3-phase Brushless DC motor (BLDC), running on 32-bit Kinetis KE17Z series MCUs. The sensorless control software itself and the BLDC control theory in general is described in 3-Phase BLDC sensorless Motor Control Application. The NXP Freedom board is used as hardware platforms for the BLDC control reference solution. The hardware-dependent part of the sensorless control software is addressed as well, including detailed peripheral setup and the Motor Control Peripheral Drivers (MCDRV).

The KE17Z MCUs integrate key features, such as 12-bit ADC, Analog Comparator (ACMP), and Flexible Timers (FTM) to simplify design and to help reduce system cost. The 3-phase BLDC motor is widely used in the field of industrial control for its high efficiency, high reliability and high-power density. Thanks to the optimized design of the 3-phase BLDC control in the chip, KE17Z is suitable for some low-end applications that have strict cost control. For example, cooling fan and water pump. This application note introduces the principle of BLDC six-step control with hall sensor, hardware and software implementation, including a detailed peripheral setup and driver description.

2 KE17Z features and advantages

The MEK17Z256 devices are highly integrated, low-power and low pin count 16-bit microcontrollers based on the Kinetis core platform.

The mainly features concerning motor control function are as follows:

Core processor and system:

- Arm® Cortex®-M0+ core, supports up to 72 MHz frequency
- Arm Core based on the Armv6 Architecture and Thumb®-2 ISA
- Configurable Nested Vectored Interrupt Controller (NVIC)
- 8-channel DMA controller extended up to 63 channels with DMAMUX

Memory and memory interfaces:

- Up to 256 KB program flash
- Up to 48 KB SRAM
- 128 bytes flash cache

Mixed-signal analog:

- 1× 12-bit analog-to-digital converter (ADC) with up to 16-channel analog inputs per module, up to 1 Msps

- 1× high-speed analog comparators (CMP) with internal 8-bit digital-to-analog converter (DAC)

Timing and control:

- 3× Flex Timers (FTM) for PWM generation, offering up to 8 standard channels
- 1× 16-bit Low-Power Timer (LPTMR) with flexible wake-up control
- 1× 32-bit Low-Power Periodic Interrupt Timer (LPIT) with 4 channels

Human-machine interface (HMI):

- Supports up to 32 interrupt request (IRQ) sources
- Up to 89 GPIO pins with interrupt functionality
- 2 x 25 channel Touch Sensing Input (TSI) module, each TSI has 12 mutual channels (up to 6 × 6 channel matrix) and 3 shield channels

Clock interfaces:

- OSC: high range 4-40 MHz (with low-power or high-gain mode) and low range 32-40 kHz (with high-gain mode only)
- 48-60 MHz high-accuracy (up to ±1%) Fast Internal Reference Clock (FIRC) for normal Run
- 8 MHz/2 MHz high-accuracy (up to ±3%) Slow Internal Reference Clock (SIRC) for low-speed Run
- 128 kHz Low-Power Oscillator (LPO)
- Low-Power FLL (LPFLL)
- Up to 60 MHz DC external square wave input clock
- System Clock Generator (SCG)

Connectivity and communications interfaces:

- 3×Low-Power Universal Asynchronous Receiver/Transmitter (LPUART) modules with DMA support and low power availability
- 1×Low-Power Serial Peripheral Interface (LPSPI) modules with DMA support and low power availability
- 1×Low-Power Inter-Integrated Circuit (LPI2C) modules with DMA support and low power availability
- FlexIO module for flexible and high performance serial interfaces

Operating characteristics:

- Voltage range: 2.7 V to 5.5 V
- Ambient temperature range: –40 °C to 105 °C

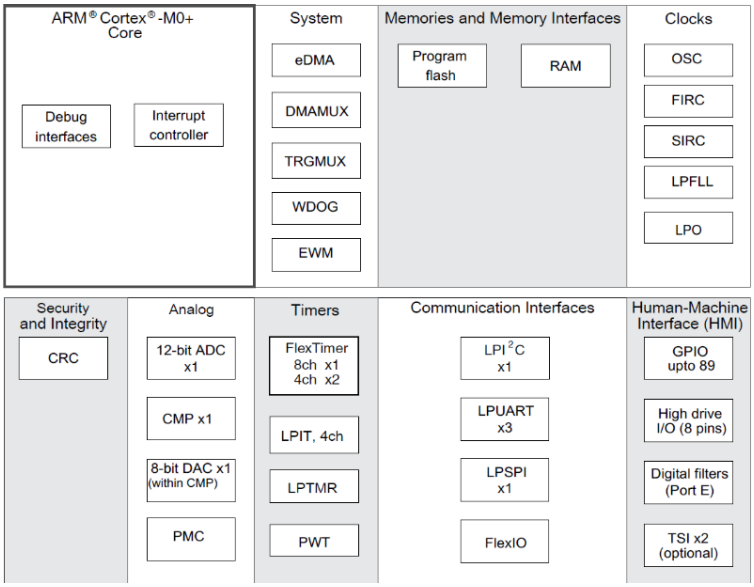

Figure 1. Functional block diagram

3 BLDC Sensorless Control theory

The BLDC motor is a rotating electric machine. The stator is similar with the 3-phase stator of a traditional induction motor; the rotor has surface-mounted permanent magnets. There are no brushes on the rotor and the commutation is performed electronically at certain rotor positions. The stator is made from silicon steel sheets. Figure 2 shows a typical cross section of a BLDC Motor. The stator-phase windings are inserted in the slots, distributed winding. Because the air-gap magnetic field is produced by permanent magnets, the rotor magnetic field is constant.

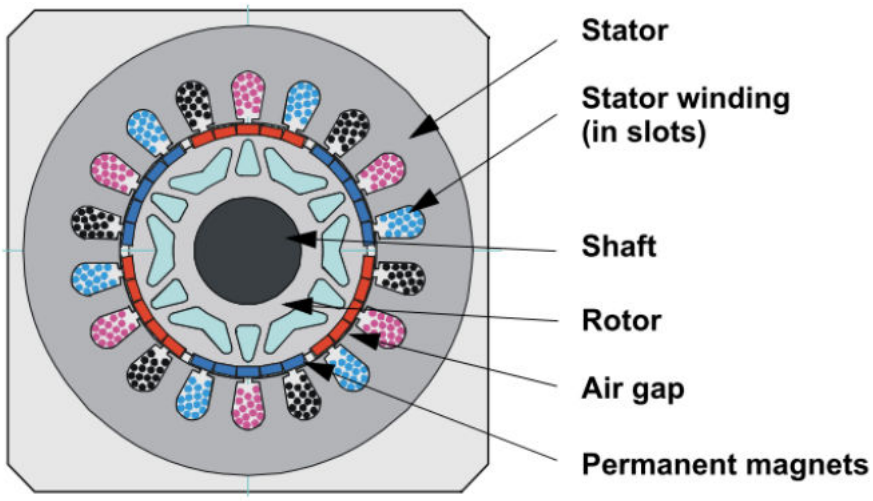

Figure 2. BLDC motor cross section

The magnetization of the permanent magnets and their displacement on the rotor is chosen so that the Back-EMF shape, the voltage induced on the stator winding due to rotor movement, is trapezoidal. The DC voltage with a rectangular shape can be used to create a rotational field with low-torque ripples, as shown in Figure 2.

Controlling BLDC motor requires a 3-phase inverter circuit. The 3-phase bridge is composed of six power switch components (Q1-Q6). Six-step commutation control is used to drive each switch component, as shown in Figure 3.

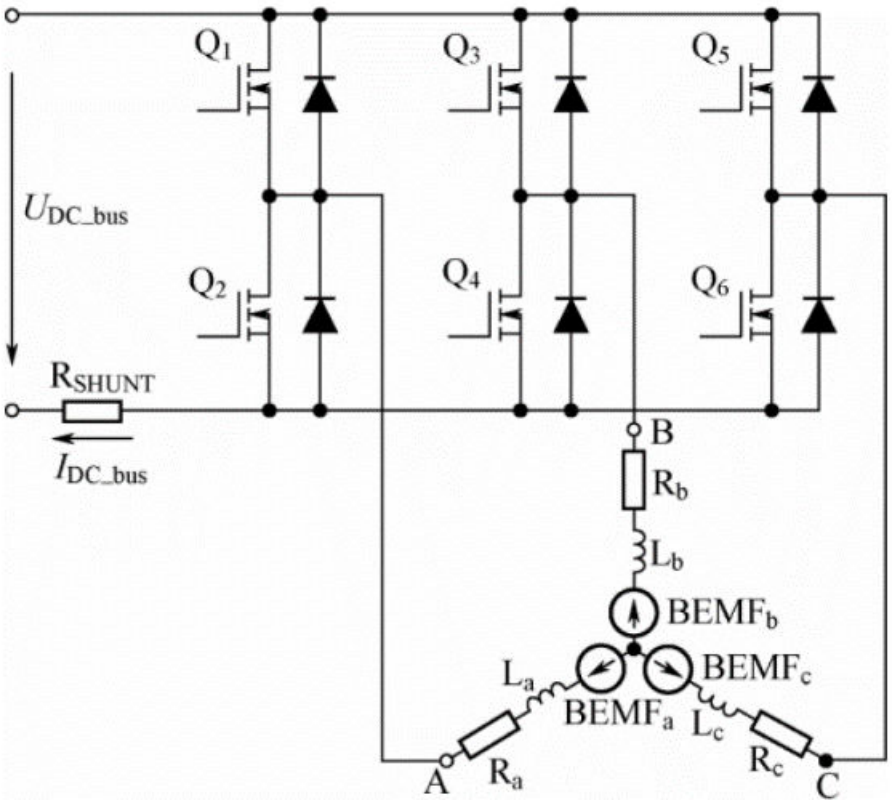

Figure 3. Phase inverter circuit

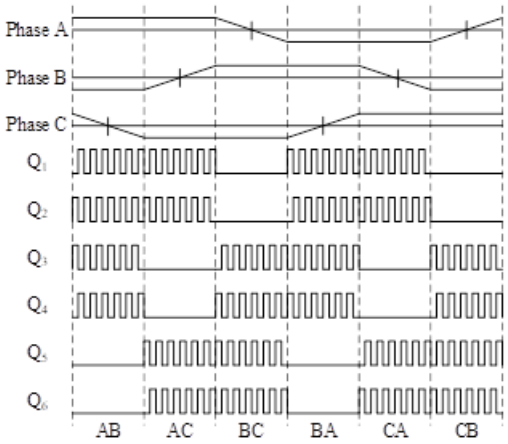

Figure 4. Phase voltage system of BLDC motor

There are multiple PWM modulation modes for six-step commutation control. The process of the rotor rotating one electrical cycle is divided into six sectors, each sector selects two phases to conduct, the current flows in from one phase to another phase, and the remaining phase is a non-conducting phase. The rotor position can be detected based on the back electromotive force(BEMF) generated in this phase. In order to obtain a better freewheeling effect when the MOS tube is turned off, a complementary output mode can be used, as shown in Figure 4, and the performance and peripherals of the Kinetis series chips also meet such requirement.

The position estimation of BLDC is relatively easy to obtain, because only the rotor position at the time of commutation needs to be estimated. For a three-phase winding motor, only six times are estimated in one electric cycle, and the rotor position differs

by 60° electrical angle between two adjacent times. Commonly used methods include back EMF method, stator third harmonic method, current path monitoring method and so on.

For a steady-state motor, the BEMF method is the simplest and most practical method. The principle is: BLDC has only two phases conduction at any time, and each phase winding conducts at 120° electrical angles in the forward and reverse directions respectively. By measuring the voltage V_a, V_b, V_c and V_N of the three-phase winding terminal and the neutral point relative to the negative end (or positive end) of the DC bus, when the potential of a certain end point is equal to the neutral point potential, such as $V_a = V_N = V_{dd}/2$, then the BEMF of the phase winding crosses zero at this moment, and the power device must be commutated after another 30° electrical angle. Based on this, a zero-crossing detection and phase-shifting (or timing) circuit can be designed to obtain the switching sequence of the full-bridge driving six power devices. This method is also called the direct BEMF method. As mentioned earlier, these two methods are only suitable for steady-speed motor operation.

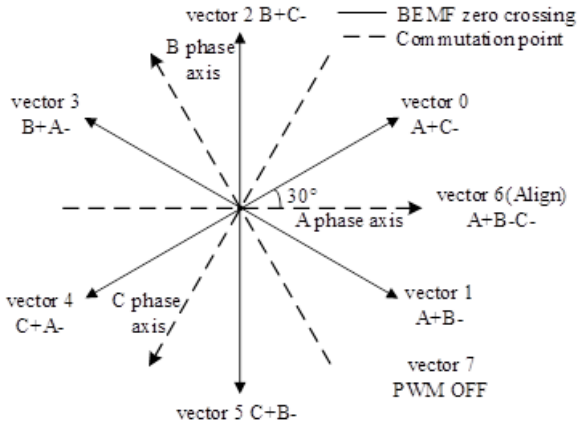

Figure 5. Commutation point and BEMF zero crossing in BLDC sensorless control

The BEMF integration method is not so sensitive to switching noise, and it can automatically adjust the switching time of the inverter to adapt to the change of rotor speed. The basic principle is to use a signal selection circuit, and the selected phase is the non-conducting winding. When the back EMF crosses zero, it begins to integrate its absolute value. When the integral value v_{int} reaches the preset threshold value v_{th}, a commutation signal is generated.

Among them, the amplitude of the BEMF is proportional to the rotation speed, and the period is inversely proportional to the rotation speed. Therefore, the back EMF is integrated at the zero crossing point and integral value at π/6 electrical angle is not related to speed but related to BEMF constant of motor.

4 Hardware and software implementation

4.1 System hardware design

The application hardware includes the following parts:

• FRDM-KE17Z

The FRDM-KE17Z Freedom Board is designed to work in standalone mode or as the main board of FRDM-TOUCH and FRDM-MC-LVBLDC. This Freedom board is compatible with DC 5v and 3.3v power supply and features a KE17Z, a device boasting up to 256 KB Flash and 32 KB SRAM and numerous analog and digital peripherals. The onboard interfaces include an RGB LED, a 6-axis digital sensor, a 3-axis digital angular rate gyroscope, an ambient temperature sensor, and two capacitive touch pads.

• FRDM-MC-LVBLDC

The FRDM-MC-LVBLDC low-voltage, 3-phase BLDC Freedom development board platform adds BLDC motor control capabilities, such as rotational or linear motion, to your design applications.

LINIX 45ZVN24-40 BLDC motor is selected. The motor control development platform block diagram and actual demo picture are as shown in Figure 6.

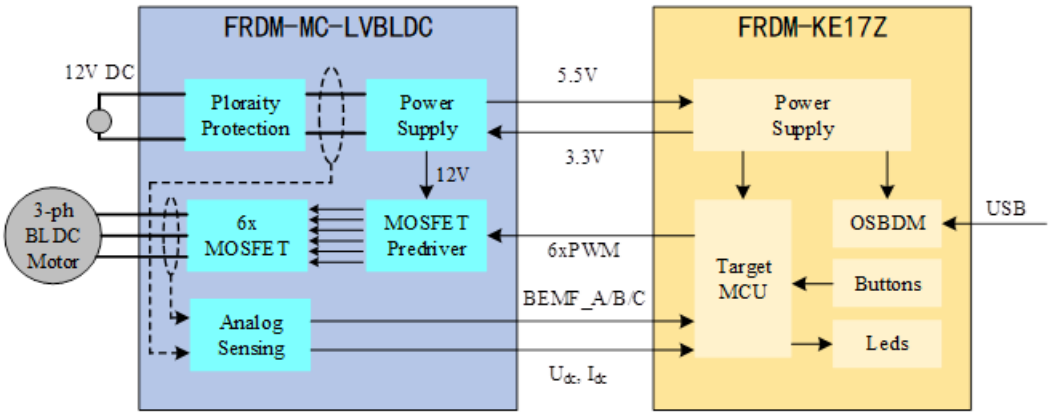

Figure 6. Motor control development platform block diagram

4.2 System software design

The software and hardware application meet the following design requirements:

Select KE17Z as controller.

- Low-voltage sensorless control with closed-loop speed and torque.
- Overvoltage, undervoltage, and overcurrent faults protection based on software.
- Minimal speed of 300 rpm, maximal speed of 2500 rpm (depending on motor used).
- Set limit current to 7 A by default.
- Support two directions of rotation.
- SW3 button control the demo mode.
- Real-time motor status monitoring based on FreeMASTER.

Figure 7 shows the system block diagram.

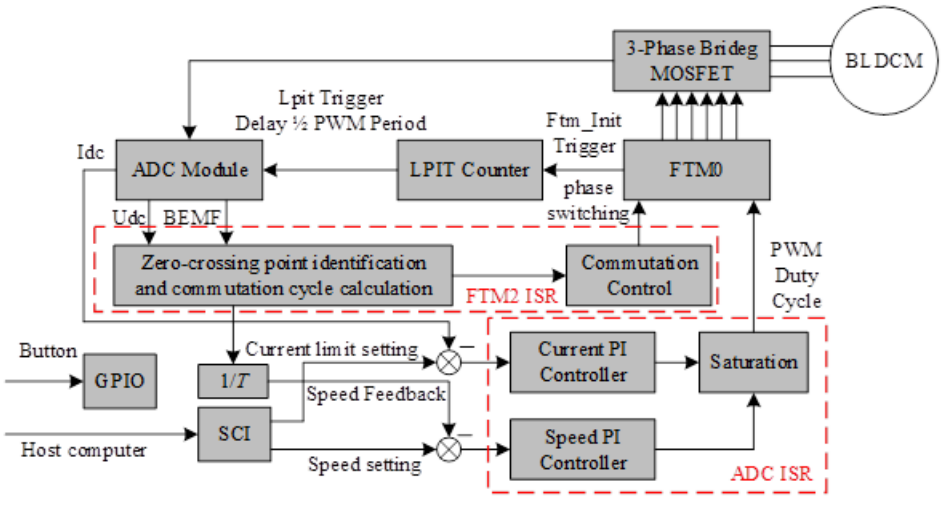

Figure 7. System block diagram

From this block diagram, we can see that the overall control process is completed in two interrupts.

- *ADC ISR* (Fast control loop, 20 kHz): ADC result acquisition, commutation control, PWM duty update and actual speed measurement are done in ADC ISR (10 KHz).
- *FTM2 ISR* (Slow control loop, 1 kHz): Speed and current PI controller calculation, application state machine update.

Since the application code is designed for 45ZWN24-40 BLDC motor, its rated voltage is 24 V and rated speed is 4000 rpm. But the output voltage of LVBLDC is 12 V, then the motor running speed will not exceed 2300 rpm under the square wave control because of the BEMF saturation. The specific phase potential waveform is shown in the figure below.

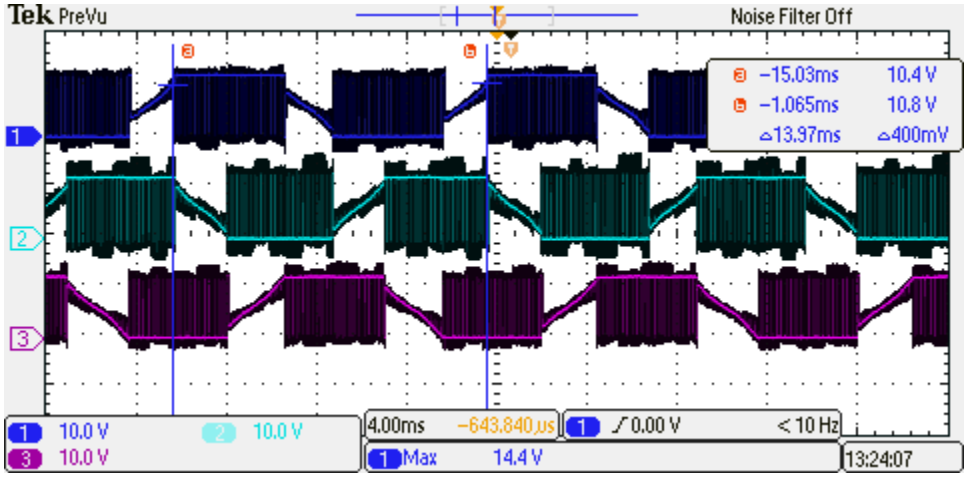

Figure 8. 3-phase BEMF waveform

5 Peripheral configurations

This section describes the configuration of peripherals used for the motor control on Kinetis series, including FTM 0/1/2, LPIT, ADC 0/1, KBI, IPC.

5.1 FTM0

Configuring FTM to generate 6 PWM outputs to drive the BLDC motor, enables the PWM overflow interrupt, read BEMF value through ADC result and detect the zero-crossing point of the BEMF and integrate to perform the corresponding commutation operation.

FTM0 configuration:

- System clock source
- Running frequency of 20 kHz with 50 µs period
- Output center-aligned and high-true pulses PWM
- Configure FTM0 channel 0 - 5 to drive BLDC motor
- Enable counter overflow interrupt

The most important thing in BLDC motor control is commutation and the commutation point detection needs to be achieved through BEMF of the non-conducting phase. Only when the MOSFET is turned off can the correct BEMF be detected in the non-conducting phase. Therefore, in the center-symmetrical PWM output waveform, it is necessary to delay the configuration of the sampling sequence for a period of time after the PWM trigger signal is generated to ensure that the sampling point of non-conducting phase BEMF is in the time when the PWM driving waveform of MOSFET on the corresponding conduction phase is low, as shown in Figure 9.

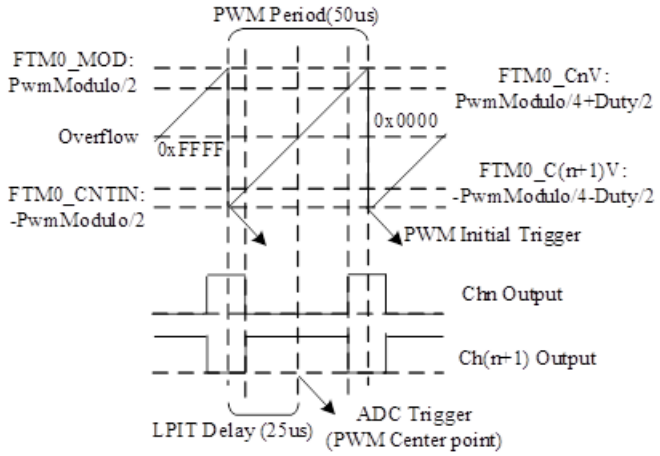

Figure 9. PWM output and sample point

5.2 LPIT0

The Low-Power Periodic Interrupt Timer (LPIT) is a multichannel timer module generating independent pre-trigger and trigger outputs. These timer channels can operate individually or can be chained together. The LPIT can operate in low-power modes if configured to do so. The pre-trigger and trigger outputs can be used to trigger other modules on the device.

Each timer channel can be configured to run independently and made to work in either compare or capture modes. In compare mode, the timers decrement when enabled and generate an output pre-trigger and timeout pulse. The trigger output is 1 clock cycle delayed of the pre-trigger pulse. Each timer channel start, reload, and restart can be controlled via control bits. The timer can be configured to always decrement, or decrement on selected trigger inputs or previous channel timeout (when channels are chained). By chaining timer channels, applications can achieve larger timeout durations. In capture mode, the timer can be used to perform measurements as the timer value is captured (in the timer value register) when a selected trigger input is asserted. In capture mode, the timer can support once-off or multiple measurements (for example, frequency measurements).

The timer channels operate on an asynchronous clock, which is independent from the be register read/write access clock. Clock synchronization between the clock domains ensures normal operations.

LPIT0 configuration:

- FIRC clock source
- Running in chain mode with COCO signal of ADC module
- Configure LPIT0 channel 0–2 to configure sampling sequence of BEMF, DC-BUS current and voltage

The trigger is configured by TRGMUX module on the chip, as shown in Figure 10. After the FTM trigger, LPIT0 starts to count and delay for 25 µs, then ADC samples sequentially controlled by chain action of LPIT trigger and COCO signal of ADC.

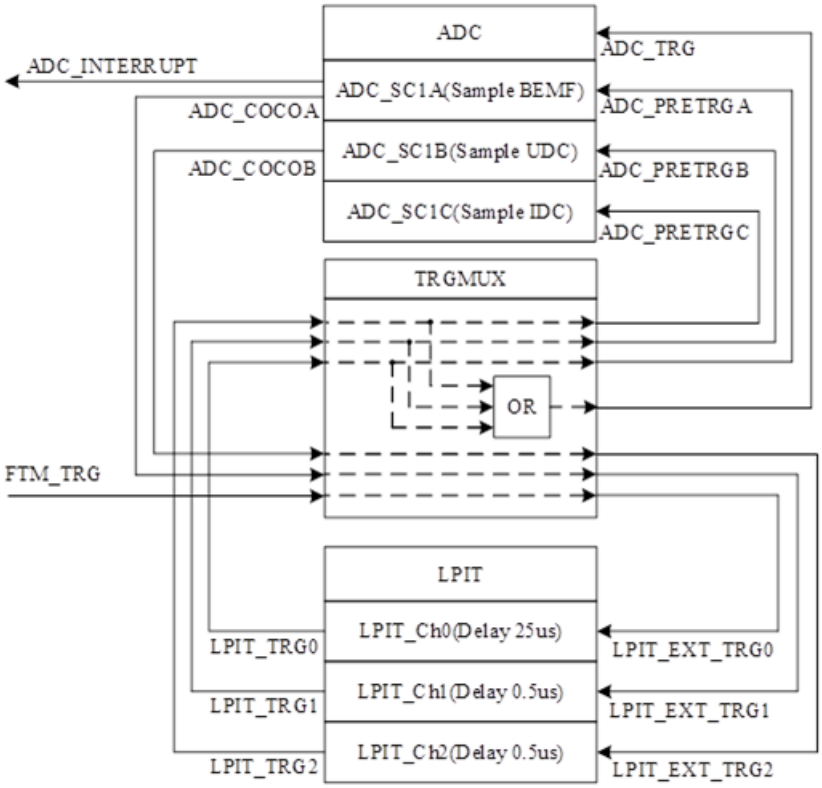

Figure 10. TRGMUX configuration

5.3 ADC

The ADC samples the DC-bus voltage, DC-bus current and BEMF of non-conducting phase. The sampled values are used to compare with the overcurrent value, overvoltage value, and undervoltage value given by the user, to realize the software protection of the motor control system. The back electro-motive force value is used to detect zero-crossing point and start to find commutation position through BEMF integral method.

ADC configurations:

- Bus clock source, clock divide value is 1.
- 12-bit sampling accuracy, long sample time
- Configure three sampling channels, PTA3/ADP3 channel is configured to sample DC-bus current, PTB3/ADP7 channel is configured to sample DC-bus voltage.

The total conversion time depends upon:

- The sample time as determined by CFG2[SMPLTS]
- The MCU bus frequency
- The conversion mode, as determined by CFG1[MODE]
- The frequency of the conversion clock, that is, f_{ADCK}.

After the module becomes active, sampling of the input begins.

1. CFG2[SMPLTS] selects between sample times based on the conversion mode that is selected.

2. When sampling is completed, the converter is isolated from the input channel and a successive approximation algorithm is applied to determine the digital value of the analog signal.

3. The result of the conversion is transferred to Rn upon completion of the conversion algorithm.

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by CFG1[ADICLK], and the divide ratio is specified by CFG1[ADIV]. To calculate total conversion time the following formula is applied:

ADC TOTAL CONVERSION TIME = Sample Phase Time (set by SMPLTS + 1) + Hold Phase (1 ADC Cycle) + Compare Phase Time (8-bit Mode = 20 ADC Cycles, 10-bit Mode = 24 ADC Cycles, 12-bit Mode = 28 ADC Cycles) + Single or First continuous time adder (5 ADC cycles + 5 bus clock cycles)

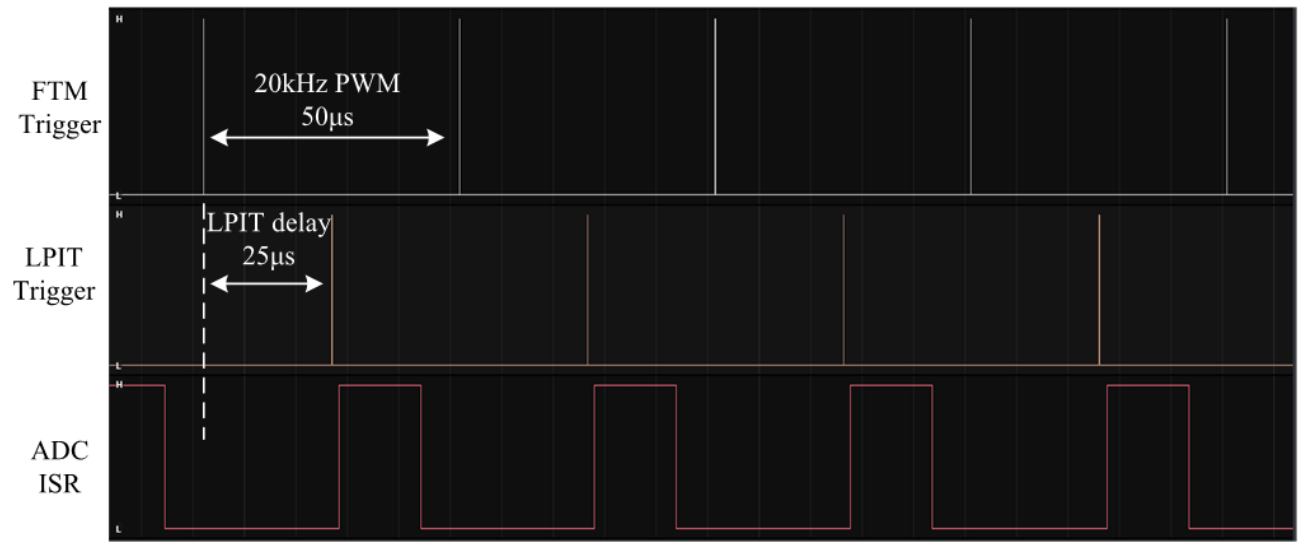

Figure 11. Trigger signal and interrupt execution program

5.4 FTM1

Since the BEMF is small and difficult to detect when the motor is running at low speed, it is necessary to perform open-loop forced commutation starting through the align and startup process to achieve the conditions that can stably detect the zero-crossing point of the BEMF and then perform closed-loop operation. Initialize the FTM1 for forced commutation control.

FTM1 configurations:

- Fixed frequency clock source
- Bus clock, divide by 128, 1.7777 µs at 72 MHz clock
- Enable the interrupt

5.5 FTM2

FTM2 is used to generate interrupt and perform slow control loop (1 kHz).

FTM2 configurations:

- Fixed frequency clock source
- Bus clock, divide by 16, 0.2222 µs at 72 MHz clock
- Enable the interrupt

6 Software implementation

This section describes the software design of the BLDC motor sensorless control application. The description of the software includes the following parts:

- Main function flow chart
- ADC interrupt
- FTM2 interrupt

6.1 Main function flow chart

After a reset, the application initializes all used peripherals and enters the endless loop. Figure 12 shows the flow chart of *main*().

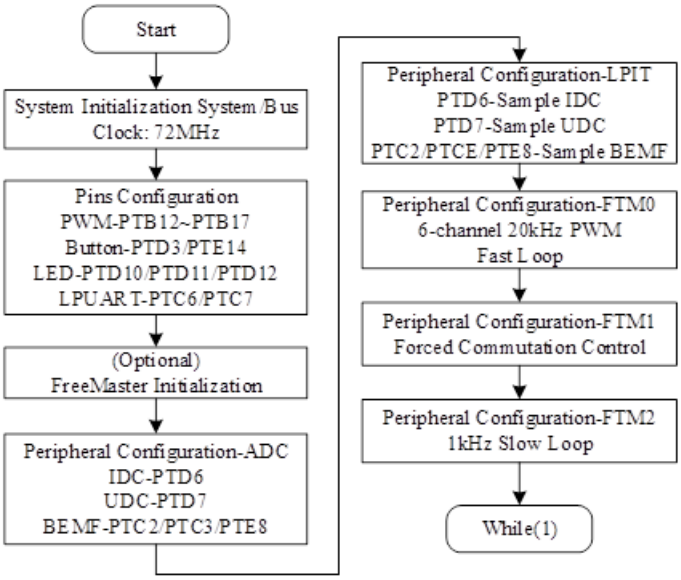

Figure 12. Main function flow chart

6.2 State machine switch

The main state machine consists of the following substates: INIT, STOP, RUN, FAULT. Figure 13 shows the main state machine switch.

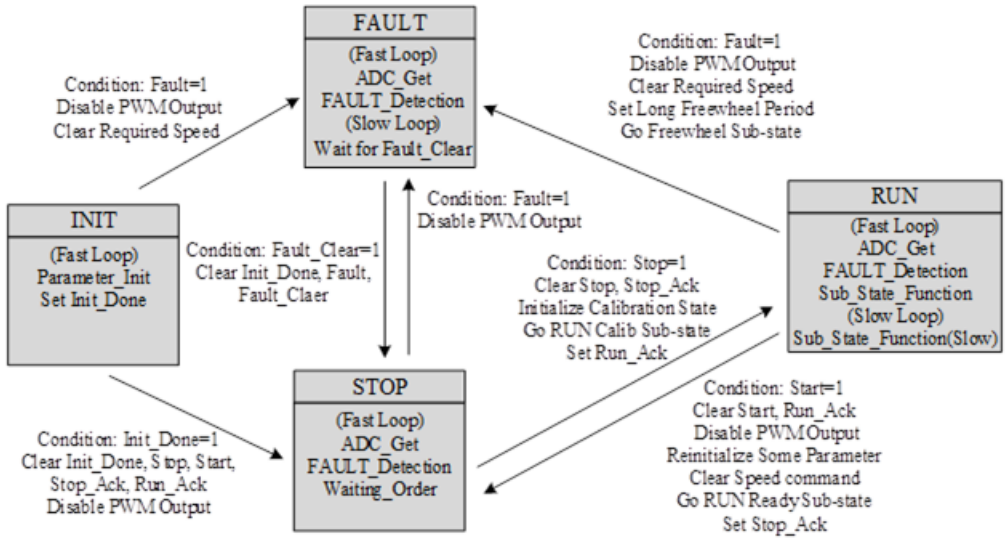

Figure 13. Main state machine flow chart

- M1_StateFaultFast (*void*): read ADC results (DC-bus voltage, current and phase voltage sample), DC-bus current and voltage filter, disable user application switch, clear fault state manually from FreeMASTER and detects faults.
- M1_StateInitFast (*void*): initialize freewheel period, alignment period, DC-bus current measurement parameters, current controller parameters, reset required speed, initialize speed ramp, speed controller parameters, sensorless algorithm parameters, fault thresholds, frequencies of control loop, commutation timer for MCAT constant calculation, Defined scaling for FreeMASTER and filter initial value.
- M1_StateStopFast (*void*): read ADC results, DC-Bus current and voltage filter, calculate BEMF voltage from phase voltage, detect user speed command and detects faults.
- M1_StateRunFast (*void*): detect fault, read ADC results, DC-bus current and voltage filter, calculate BEMF voltage from phase voltage, run RUN substate function, configure ADC for next measurement.
- M1_StateFaultSlow (*void*): after fault condition ends wait defined time to clear fault state.
- M1_StateRunSlow (*void*): run RUN substate function.
- M1_TransInitFault (*void*): disable PWM outputs and clear required speed.
- M1_TransInitStop (*void*): disable PWM outputs.
- M1_TransStopFault (*void*): log error messages.
- M1_TransStopRun (*void*): pass calibration routine duration to state counter, initialize calibration state, got to RUN Calib substate and acknowledge that the system can proceed into the RUN state.
- M1_TransRunFault (*void*): disable PWM outputs, turn off application, clear required speed, clear required speed, set long Freewheel period and go to RUN Freewheel substate.
- M1_TransRunStop (*void*): disable PWM outputs, re-initialize PI controllers and duty cycle, clear speed command, enter RUN Ready substate and acknowledge that the system can proceed into the STOP state.

As shown in Figure 13, speed PI control loop is realized in AppRun function. There is a detailed description about the speed control process, below figure shows the block diagram of speed PI control.

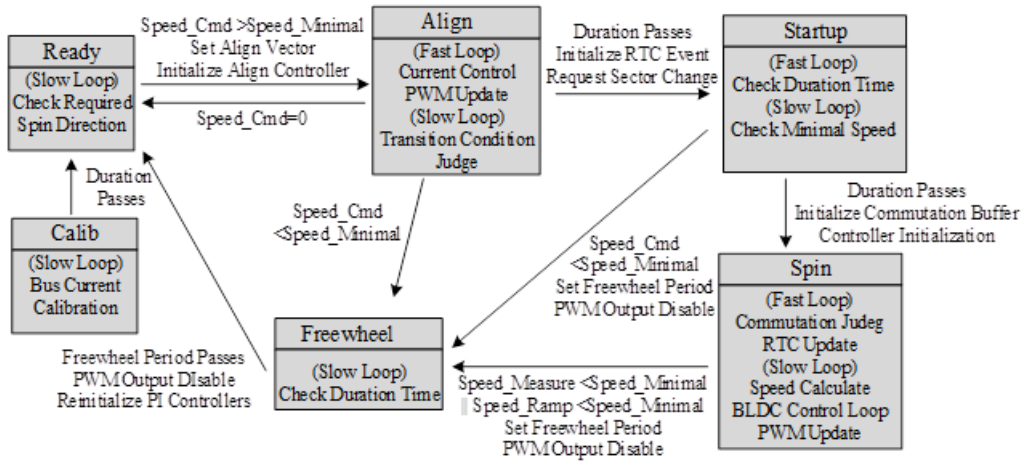

Figure 14. Run substate machine flow chart

- M1_StateRunAlignFast (*void*): current closed loop control.
- M1_StateRunStartupFast (*void*): calculate Startup count value and transport to RUN Spin substate.
- M1_StateRunSpinFast (*void*): check Toff period after commutation event, mirror BEMF voltage according to the sector (change falling BEMF to rising BEMF), integrate if positive BEMF voltage, check whether the integral of BEMF voltage threshold has been reached, save commutation time and periods, perform commutation, request sector change, calculate Toff time, calculate next time of safety commutation, request next time event, reset BEMF integral value and decrement commutation error counter.
- M1_StateRunCalibSlow (*void*): integrate DC-bus current offset, wait until the calibration duration passes then set the DC-bus measurement offset value and transition to the RUN Ready substate.
- M1_StateRunReadySlow (*void*): if the required speed is higher than minimal, switch to Ready state then check required spin direction and transition to the RUN Align substate.
- M1_StateRunAlignSlow (*void*): check if the required speed is lower than minimal then switch to RUN Freewheel substate, check Align count value then switch to RUN Startup substate, if zero speed command go back to RUN Ready substate.
- M1_StateRunStartupSlow (*void*): check if the required speed is lower than minimal the transition to the RUN Freewheel substate.
- M1_StateRunSpinSlow (*void*): calculate absolute value of average speed, call BLDC control loop - speed and current PI controller, if actual speed was less than minimal speed boundary then transition to the RUN Freewheel substate.
- M1_StateRunFreewheelSlow (*void*): wait until free-wheel time passes then switch to RUN Ready substate.
- M1_TransRunCalibReady (*void*): switch to RUN Ready substate.
- M1_TransRunReadyAlign (*void*): request alignment vector and zero duty cycle, init alignment period and initialize align current controller.
- M1_TransRunAlignStartup (*void*): initialize startup commutation period counter and next commutation period time, request next time event, select next sector based on required spin direction, request sector change and go to RUN Startup substate.
- M1_TransRunStartupSpin (*void*): calculate Toff period, initialize commutation period buffer, clear BEMF integrator, DC-bus current and speed PI controller initialization, initialize speed ramp based on spin direction and request next time event.
- M1_TransRunSpinFreewheel (*void*): set long Freewheel period - expected motor spinning and PWM output disable request.
- M1_TransRunStartupFreewheel (*void*): set short Freewheel period and PWM output disable request.
- M1_TransRunFreewheelReady (*void*): PWM output disable request and re-initialize PI controllers and duty cycle.

6.3 Speed PI closed loop

As shown in Figure 15, speed PI control loop is realized in RUN state. There is a detailed description about the speed control process, below figure shows the block diagram of speed PI control.

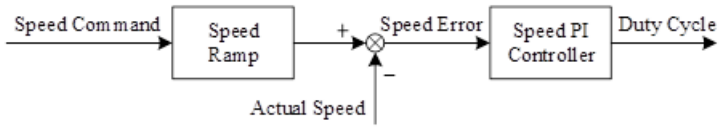

Figure 15. Speed PI control bock diagram

Since the overall application is a system with large inertia, the speed command must be refined during the application, otherwise, the system may be overloaded. One method is to generate a ramp, to make the speed ramp approach the speed command (required speed value) by step increments.

6.4 Actual speed measure

All speed constants are scaled in this application code and the speed constants divided by the pre-defined maximum value (N_MAX = 4400 rpm), then convert the signed fractional number in the range of [-1,1) into a fixed point 16-bit number (* 32768) in the format Q1.15.

The actual speed value is calculated by using the commutation period of per 60° electrical degrees and a scale constant M1_SPEED_SCALE_CONST which enables time value conversion to speed value. The actual speed f16SpeedMeasuredAbs is the filtered value of the average of the count values ui32PeriodSixCmtSum of the first six sectors of the current sector.

The actual speed f16SpeedMeasuredAbs is calculated using the following equation:

$$f16SpeedMeasuredAbs = \frac{M1\_SPEED\_SCALE\_CONST}{ui32PeriodSixCmtSum}$$

where M1_SPEED_SCALE_CONST is pre-defined as following:

NUMERATOR_FOR_SPEED=((PWM_FREQUENCY_KHZ*1000.0*60.0/6.0/PP)/N_MAX)

where:

• PWM_FREQUENCY_KHZ: 56.25 (kHz)

• N_MAX: 4400(rpm)

• PP: 2 (pole pairs)

7 References

These references are available on www.nxp.com:

- KE1xZ256/KE1xZ128 Sub-Family Reference Manual (KE1xZP100M72SF0RM)
- Tuning Three-Phase BLDC Motor Sensorless Control Application Using the MKV10x (AN4870)
- Sensorless BLDC Control on Kinetis KV and KE (AN5263)
- BLDC Motor Control with Hall Effect Sensors Using MQX on Kinetis (AN4376)

8 Revision history

The following table lists the substantive changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	25 January 2022	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP 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. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 January 2022
Document identifier: AN13489

arm