AN12957

3-phase BLDC Control Based on S08PB16

Rev. 0 — 09/2020 Application Note

Contents

1 Introduction

This application note describes the implementation of the 3-phase Brushless DC motor (BLDC) control with Hall sensor based on the NXP S08PB16 processor.

The S08PB MCUs integrate key features like 12-bit ADC, Analog Comparator (ACMP), Amplifier (OPAMP), Fault Detection and Shutdown (FDS), 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, S08PB16 is particularly 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.

The software of this application is based on CodeWarrior 11.1 IDE. Download and install the service pack first.

2 S08PB16 features and advantages

The MC9S08PB16 devices are highly-integrated, low-power and low pin count 8-bit microcontrollers based on the S08P core platform.

The features are as follows:

- Maximum CPU frequency of 20 MHz which is used as bus frequency.
- Scalable memory options, up to 16 KB Flash and 1 KB RAM.
- Operating voltage ranges from 2.7 V to 5.5 V with full functional Flash program/erase/read operations.
- Multiple package options of 20-pin and 16-pin.
- Ambient operating temperature ranges from -40 °C to 105 °C for V part and -40 °C to 125 °C for M part.

Figure 1 shows the system block diagram.

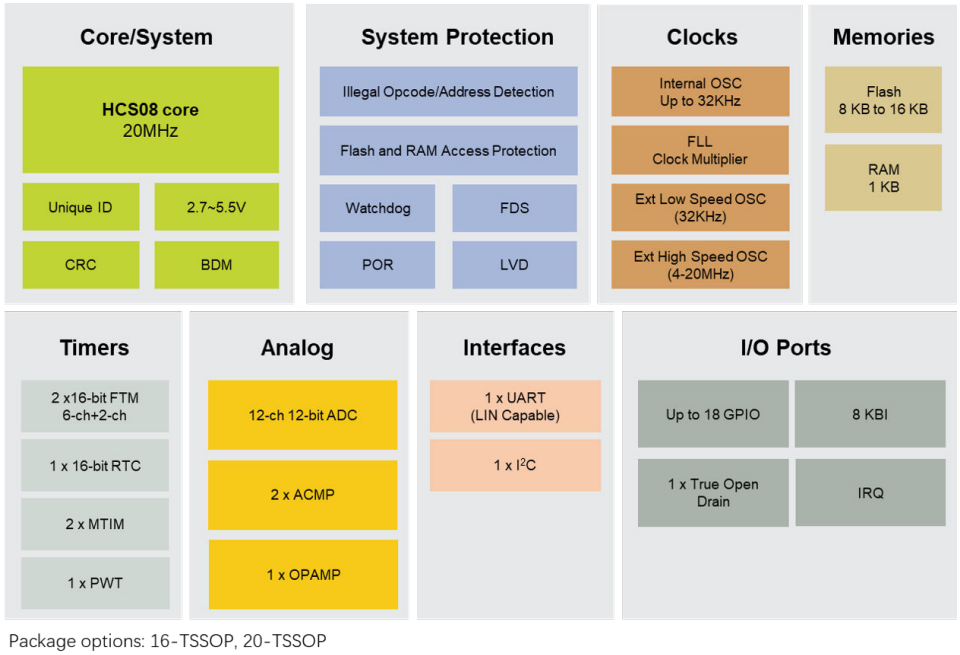

Figure 1. S08PB16 system block diagram

For BLDC motor control, the advantages of S08PB16 are:

- Multi-channel PWM signal output.
- Abundant timer and communication interfaces.
- Rich analog IP internal integrated, 12-channel 12-bit ADC, two ACMP and internal OPAMP. The interconnection of these modules is convenient for overcurrent protection. OPAMP can save the external op amp circuit, saving BOM costs.
- Enhanced FDS fault protection module that optimized for motor control can ensure high system reliability.

3 BLDC motor 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 usually 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.

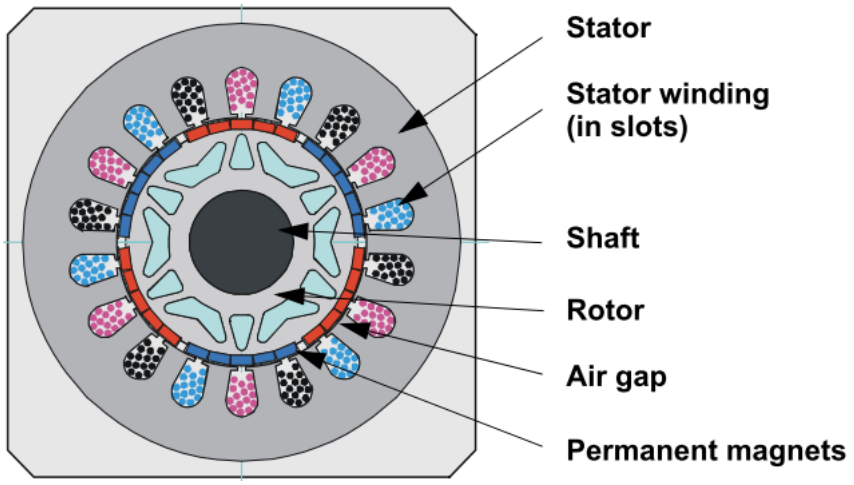

Figure 2. BLDC motor cross section

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 4.

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 usually used to drive each switch component, as shown in Figure 3.

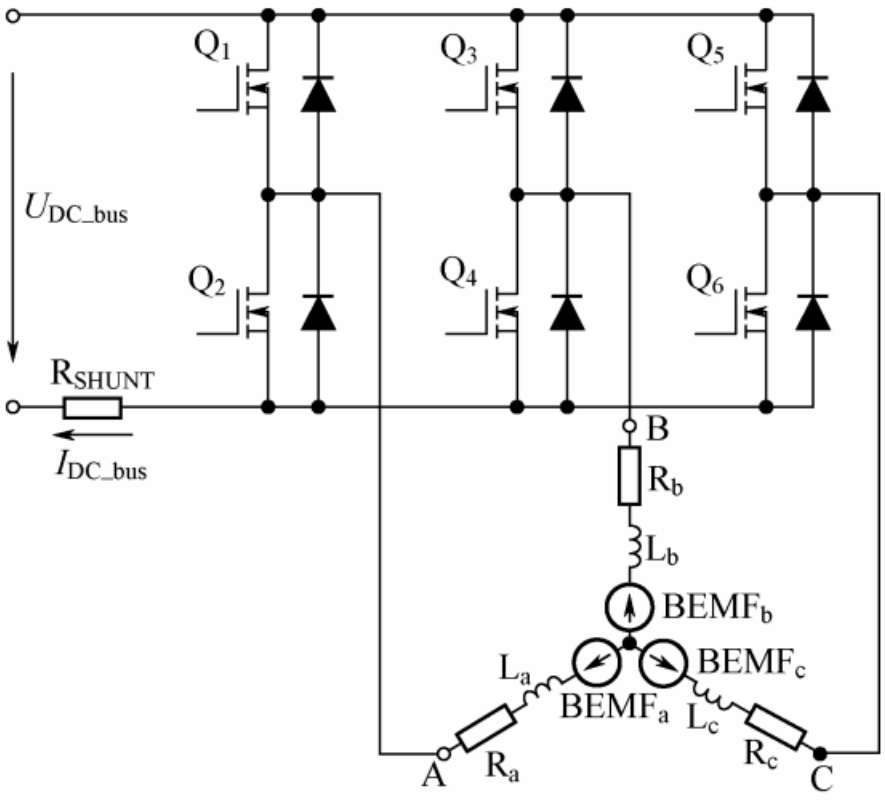

Figure 3. 3-phase inverter circuit

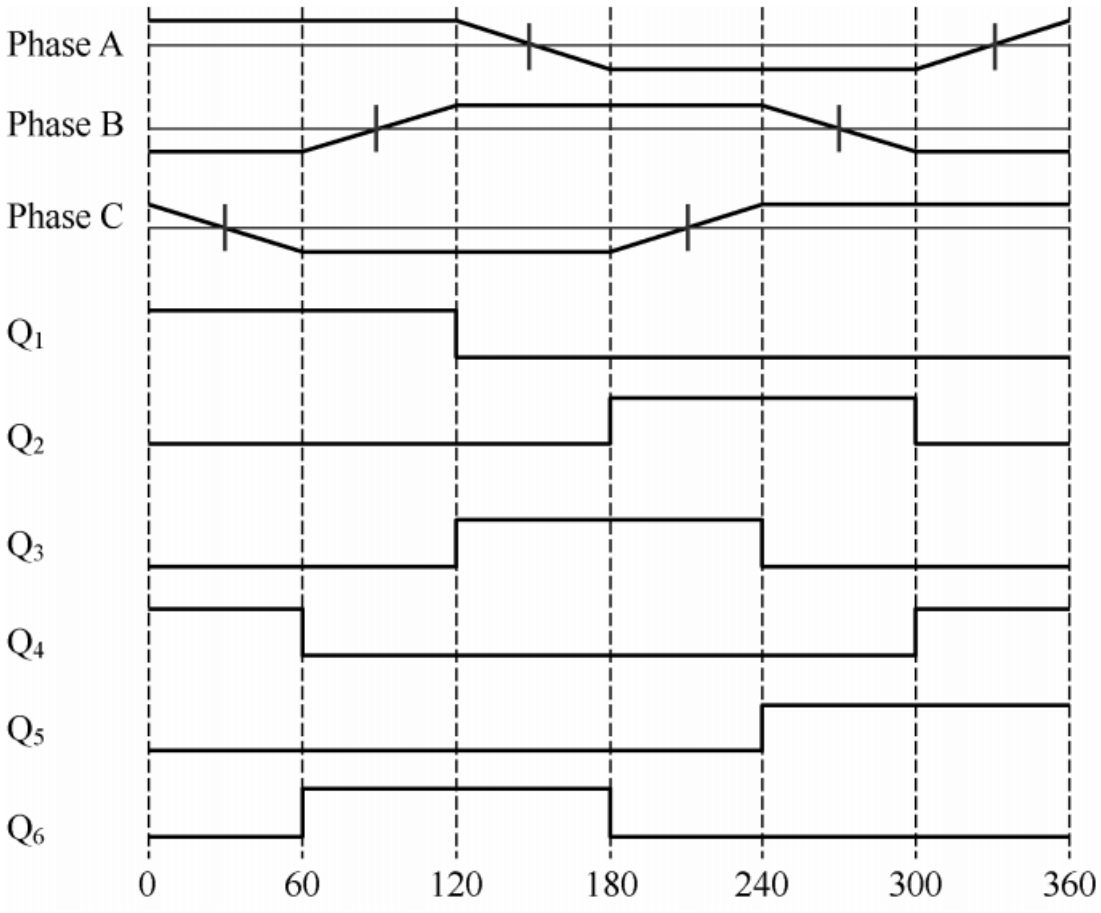

Figure 4. 3-phase voltage system of BLDC motor

There are multiple PWM modulation modes for six-step commutate control. Considering that there is no automatic dead-time insertion function in the S08PB16 FTM module, it is decided to use `HPWM_LON` mode modulation, as shown in Figure 5.

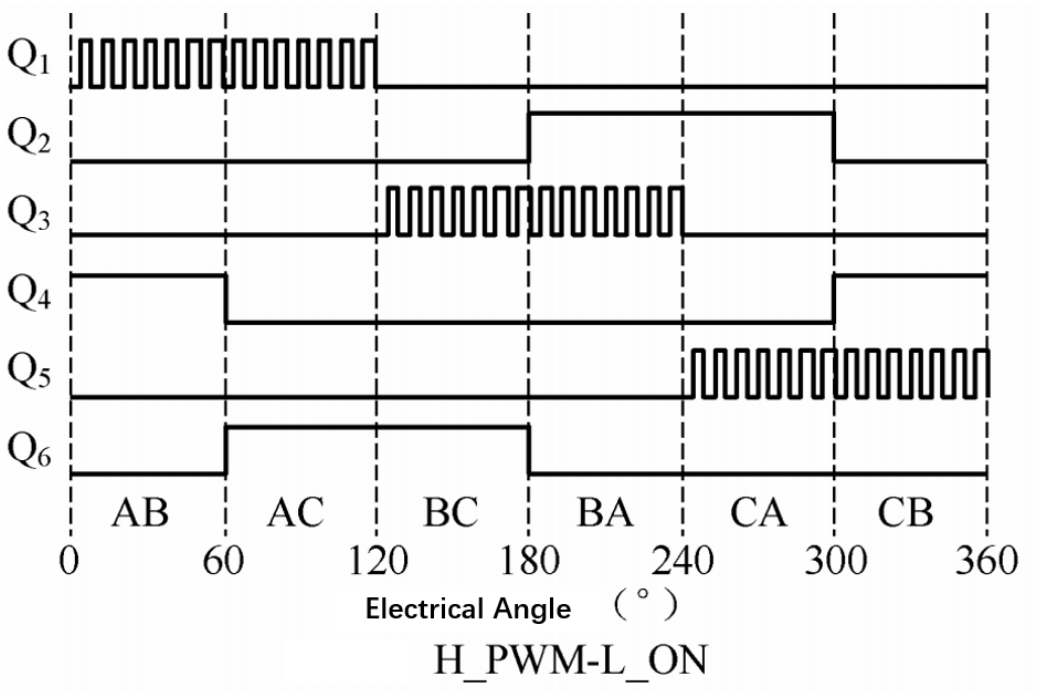

Figure 5. HPWM-LON modulation modes

4 Hardware and software implementation

4.1 System hardware design

The application hardware includes the following parts:

- **S08PB16-EVK**

  The S08PB16-EVK is cost-effective development hardware for the NXP S08PB and S08PLS 5 V 8-bit MCUs. It is small, yet powerful, with rich integrated peripherals to evaluate all features of S08PB and S08PLS MCUs. S08PB16-EVK supports OSBDM to flash program and run-control debug without an external debug tool.

- **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 and Figure 7.

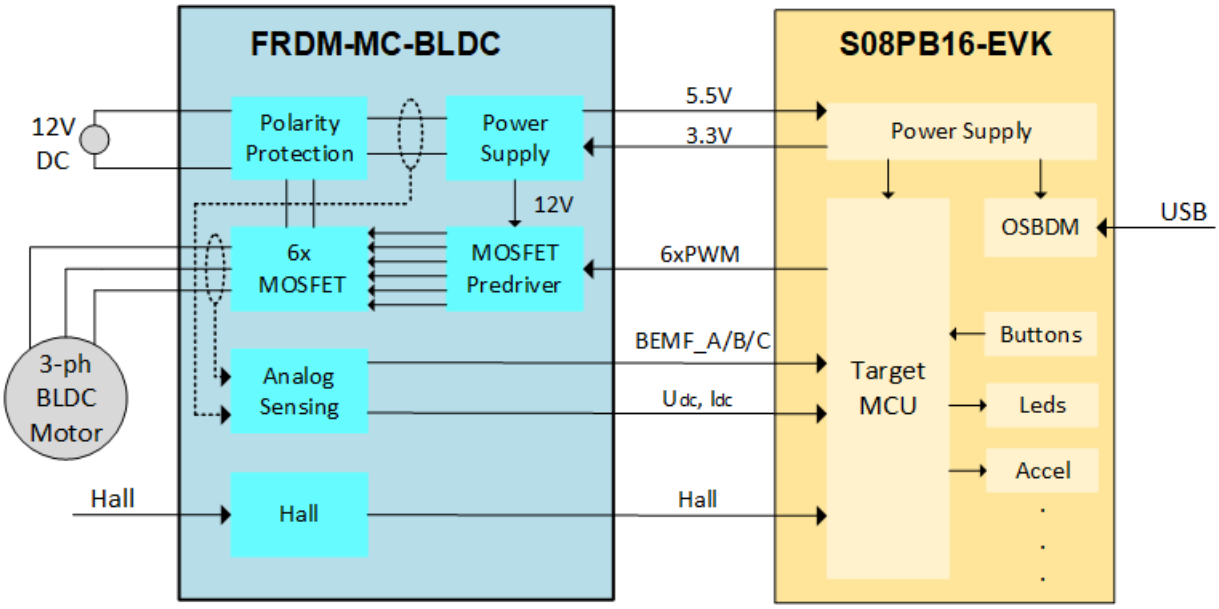

Figure 6. Motor control development platform block diagram

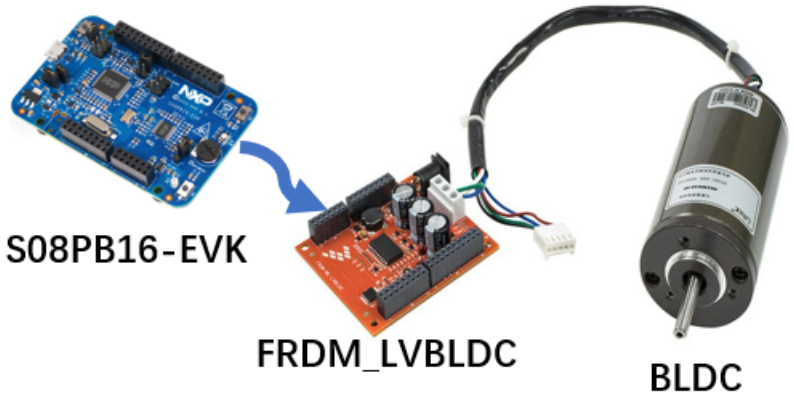

Figure 7. Actual demo picture

4.2 System software design

The software and hardware application meet the following design requirements:

- Select S08PB16 as controller.
- Full-speed closed-loop control based on Hall sensor.
- Overvoltage, undervoltage and overcurrent faults protection based on hardware and software.
- Minimal speed of 300 rpm, maximal speed of 2500 rpm (depending on motor used).
- Set limit current to 4 A by default.
- Support two directions of rotation.
- Start from any motor position without rotor alignment.
- SW3 button control the demo mode.

- Real-time motor status monitoring based on FreeMASTER.

Figure 8 shows the system block diagram.

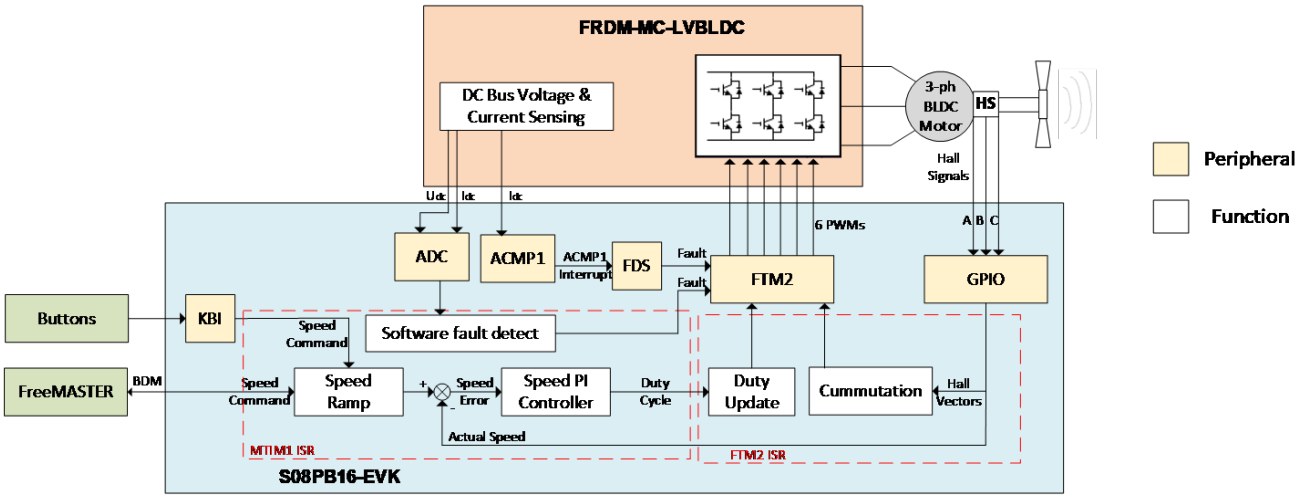

Figure 8. System block diagram

As shown in Figure 8, the overall control process is completed in three interrupts.

- **FTM2 ISR**

  Hall signal acquisition, commutation control, PWM duty update and actual speed measurement are done in FTM2 ISR (10 KHz). The sector where the rotor is located is determined by polling the GPIO values connected to hall signals in this interrupt.

- **MTIM1 ISR**

  Speed PI controller calculation, application state machine update and software fault protection are completed in the MTIM1 ISR (1 KHz). The output PWM duty cycle of speed PI controller is updated in FTM2 ISR. ADC samples DC-bus current and DC-bus voltage to detect the overcurrent, overvoltage and undervoltage faults, to implement software protect the system.

- **ACMP1 interrupt**

  System hardware overcurrent fault is caused by ACMP1 interrupt. In ACMP1 ISR, the corresponding flag of the overcurrent fault, ACMP1 interrupt flag, is clear.

5 Peripheral configurations

This section describes the configuration of peripherals used for the motor control on S08PB16, including FTM 2 , ACMP1 & FDS, ADC, KBI, MTIM1, IPC, PORTA.

5.1 FTM2

Configuring FTM to generate 6 PWM outputs to drive the BLDC motor, enables the PWM overflow interrupt, reads the HALL signal in real time during the interrupt to obtain the rotor position signal, and performs the corresponding commutation operation.

FTM2 configuration:

- System clock source
- Running frequency of 10 kHz with 100 µs period
- Output edge-aligned and high-true pulses PWM
- Configure FTM2 channel 0 - 5 to drive BLDC motor.
- Enable counter overflow interrupt

The most important thing in BLDC motor control is commutation. For HPWM_LON modulation mode, at any moment, only one MOSFET is in PWM modulation, For S08PB16, we can use SYS_SOPT8 and SYS_SOPT7 to conveniently control the output of each bridge without affecting the actual FTM timing. Take sector 1 as an example (A+B- conduction):

```
SYS_SOPT8 = SYS_SOPT8_FTM2CH3OCV_MASK;//Q4 active
SYS_SOPT7 = SYS_SOPT7_FTM2CH1OC_MASK | SYS_SOPT7_FTM2CH2OC_MASK | SYS_SOPT7_FTM2CH3OC_MASK |
SYS_SOPT7_FTM2CH4OC_MASK | SYS_SOPT7_FTM2CH5OC_MASK;//Q1 PWM output
```

The 6 PWM outputs are also controlled by FDS module. Once the overcurrent fault signal is generated to trigger FDS, FDS will shut down the 6 PWM signals and control the outputs of the 6 pins. Figure 9 shows the generate and control process for PWM signals.

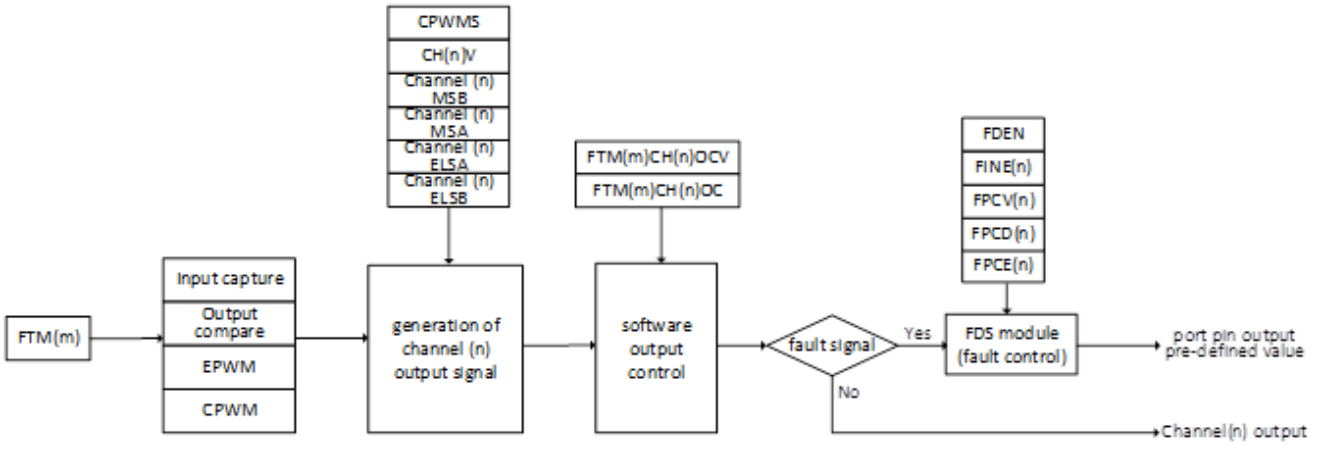

Figure 9. Generate and control process for PWM signals

5.2 ACMP and FDS

The system hardware overcurrent fault signal is generated by internal ACMP1 interrupt.

External OPAMP output is connected to ACMP1+ input, ACMP1- is connected to the internal DAC output of ACMP1, ACMP1 can compare the OPAMP output with the voltage of the internal DAC. At the same time, we can configure **ACMP1_C1_DACVAL** to change the DAC output value to adjust the limit current value. When OPAMP output is larger than DAC output value, ACMP1 interrupt flag is set, the overcurrent fault signal is generated.

FDS fault input2 source (FIN2) is ACMP1 interrupt output. Once the overcurrent fault signal occurs, FDS will shut down the output of 6 PWM channels and set the corresponding port pins to output pre-defined value 0. For specific information about internal OPAMP, ACMP and FDS, see *How to Use the Interconnection of OPAMP, ACMP1 and FDS for S08PB16* (document AN12836).

ACMP1 configurations:

- ACMP1+ input source: PTA3/ACMP1IN0/OPAMP+, the PTA3 pin connects to external OPAMP output.
- ACMP1- input source: DAC output value.
- ACMP1 output enable, used to determine whether the hardware overcurrent fault flag, FDS interrupt flag, should be removed.
- ACMP1 interrupt enable and ACMP1 interrupt on output rising edge.

FDS configurations:

- FDS input 2 enable (FIN2), ACMP1 interrupt output as input.
- Configure FDS FDSOUT0 - FDSOUT5 channels: Enable the corresponding six pins configuration, the direction of the 6 pins set as output and the output values of the 6 pins set to 0.

  Figure 10 shows the interconnection of ACMP1 and FDS.

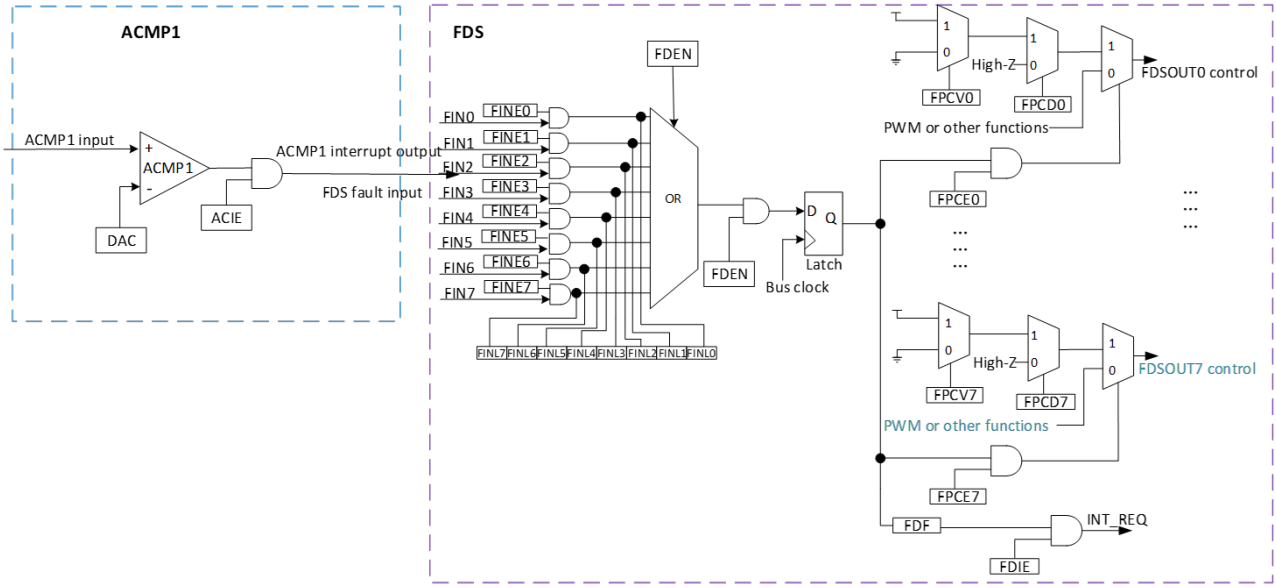

Figure 10. Interconnection of ACMP1 and FDS

- There is already an external OPAMP on FRDM-MC-LVBLDC board, the bus current is not sampled by S08PB16 internal OPAMP in this application. Customers can choose to use external OPAMP or S08PB16 internal OPAMP to sample according to their needs in practical applications.

5.3 ADC

The ADC samples the DC-bus voltage and DC-bus current, and the sampled values will be 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.

ADC configurations：

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

5.4 MTIM1

Use the MTIM1 module to generate a 1ms interrupt to implement the state machine detect, speed loop control and so on.

MTIM1 configurations:

- Fixed-frequency clock source
- No prescaler, the modulo value is 16, MTIM1 interval frequency is set to 1 kHz.
- Enable the interrupt

5.5 IPC

The Interrupt Priority Controller (IPC) module is used to configure FTM2 and MTIM1 interrupt priority level to implement interrupt nested. It needs to configure FTM2 interrupt priority level higher than MTIM1 interrupt priority level in this BLDC motor control application code.

MTIM1 interrupt priority level is set to **0** and FTM2 interrupt priority level is set to **3**. FTM2 interrupt request can preempt MTIM1 interrupt being serviced, and the MTIM1 interrupt request is blocked when executing FTM2 ISR.

IPC configurations:

- Interrupt priority controller enable.
- FTM2 interrupt priority level set to **3** and MTIM1 interrupt priority level set to **0**.

5.6 KBI

The manual interface button (onboard SW3) is used to control the demo mode on or off. As known as the S08PB16-EVK schematic, PTB2/KBI0P6 pin is configured as the SW3 button input channel.

KBI configurations:

- Only detect the falling edge.
- KBI interrupt is disabled.
- PTB2 is enabled as KBI pin.

5.7 PORTA

3-phase hall sensor signals connect to PTA0, PTA1, PTA2. The sector where the rotor is located is determined by polling the GPIO values connected to hall signals in this interrupt.

When the motor is working, the position of the rotor is determined by continuously reading the values of the three pins.

PORTA configurations:

- Configure PTA0, PTA1, PTA2 data direction as input to read hall values

6 Software implementation

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

- Main function flow chart
- MTIM1 interrupt
- FTM2 interrupt

6.1 Main function flow chart

After a reset, the application initializes all used peripherals and enters the endless loop. Figure 11 shows the flowchart of `main()`.

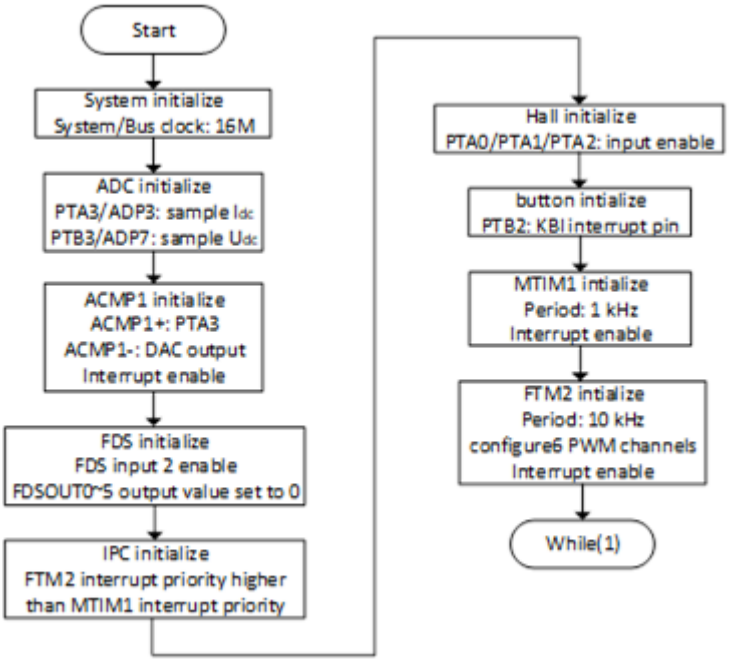

Figure 11. Main function flow chat

6.2 MTIM1 interrupt

Figure 12 shows the detailed process of the MTIM1 interrupt service routine.

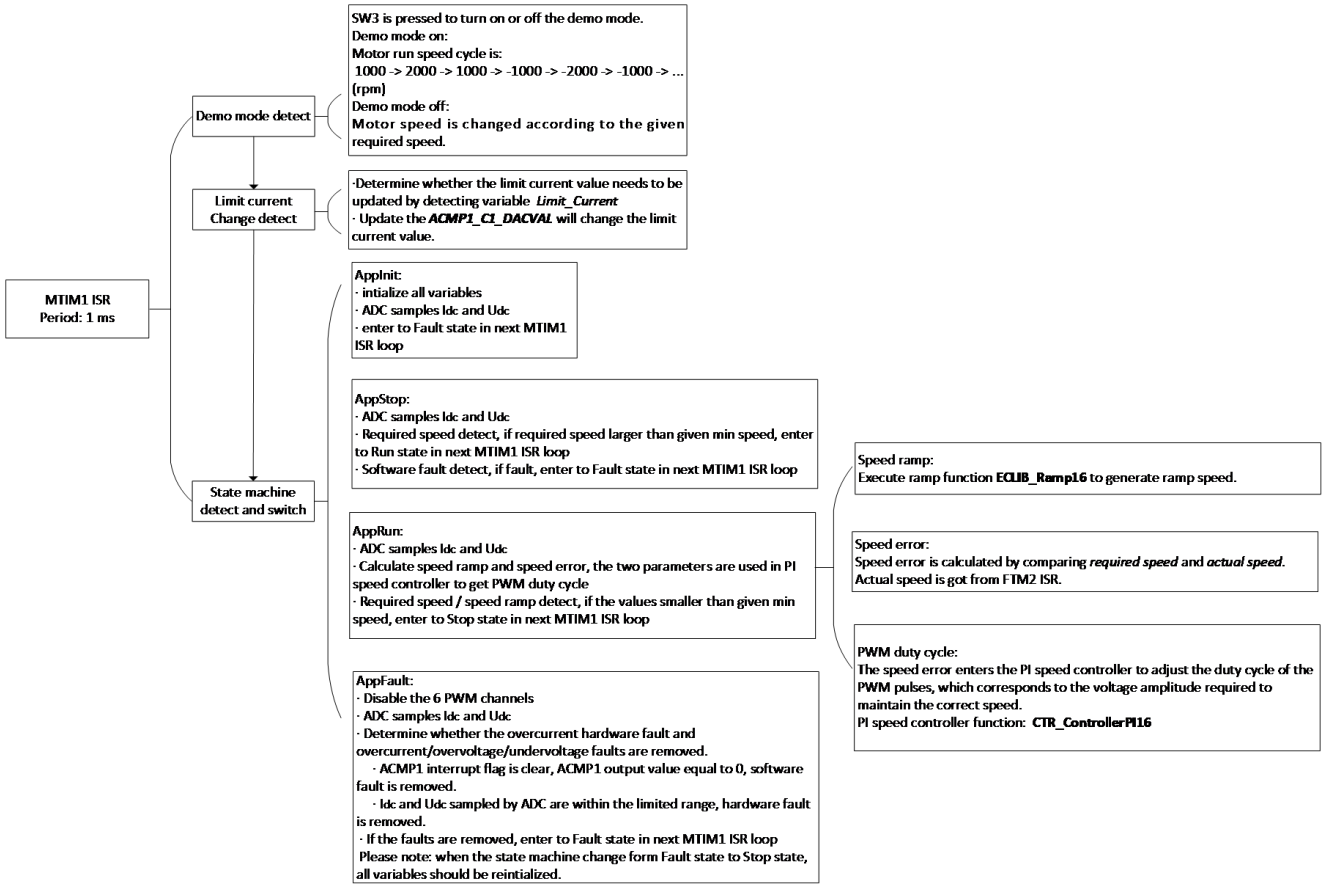

Figure 12. MTIM1 interrupt service routine

6.2.1 State machine switch

The main state machine consists of the following sub-states: init, stop, run, fault. Figure 13 shows the main state machine switch.

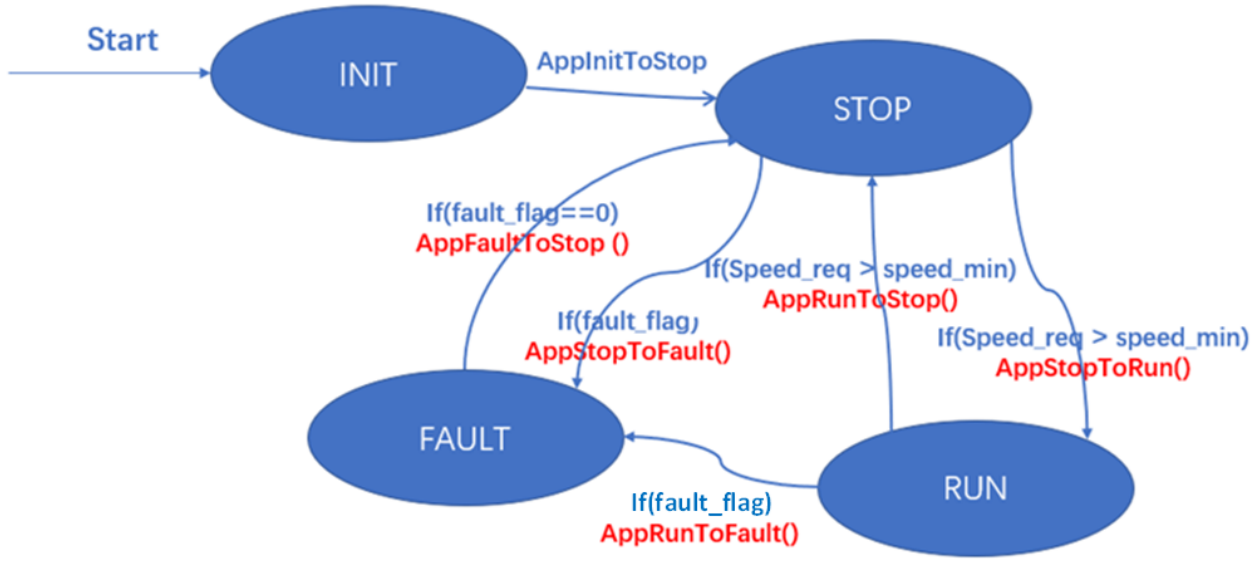

Figure 13. Main state machine flowchart

6.2.2 Speed PI closed loop

As shown in Figure 12, speed PI control loop is realized in AppRun function. There is a detailed description about the speed control process. Figure 14 shows the block diagram of speed PI control.

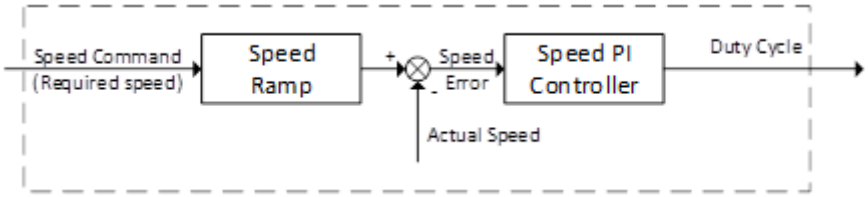

Figure 14. Speed PI control bock diagram

1. **Speed ramp**

   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 defined in the code.

   In this application, it needs three parameters in **ECLIB_Ramp16** function to get actual speed ramp.

   - Predefined ramp value (`RAMP_SPEED`)
   - Required speed (`w16Speed_req`)
   - Speed ramp (`w16Speed_ramp`)

   `RAMP_SPEED` is defined as follows.

```
/* Speed ramp value settings for close loop */
#define SLOW_PERIOD_US    1000.0//us
#define RAMP_SPEED_S    10000.0//(rpm/s)
#define RAMP_SPEED      (RAMP_SPEED_S*SLOW_PERIOD_US*32768.0/1000000.0/N_MAX)
```

NOTE

The `RAMP_SPEED` parameter is scaled. For the description of speed scale, see Actual speed measure.

Figure 15 shows the process of **ECLIB_Ramp16** function to increment or decrement the ramp speed value by a defined step to bring the ramp value closer to the required value.

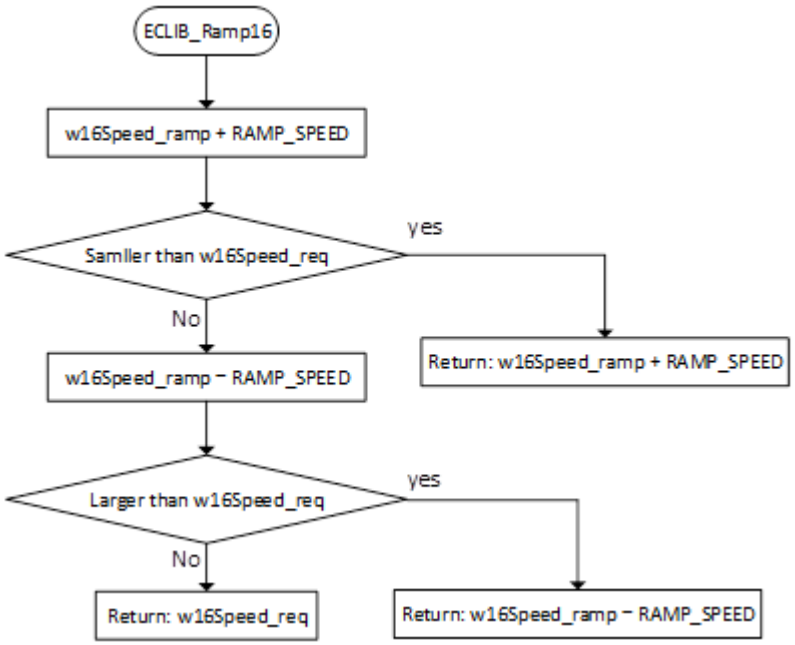

Figure 15. Flow chart for getting speed ramp

2. **PI controller**

The speed PI control algorithm attempts to correct the speed error between required speed and actual speed, *w16Speed_error = w16Speed_ramp - w16Speed_Act_flt*. The PI controller output is passed to the PWM generator (FTM2) as a newly corrected value of the applied motor voltage.

The Proportional-Integral (PI) algorithm in the continuous time domain is as shown in Equation 1.

$$u(t) = K_p \cdot e(t) + K_i' \int_0^t e(t)\,dt$$

Equation 1. PI algorithm

Where:

- *e(t)*: Input error in the continuous time domain.
- *u(t)*: Controller output in the continuous time domain.
- K_p: Proportional gain in the continuous time domain.
- K_p: Integral gain in the continuous time domain.

Equation 2 describes the discrete equation.

$$u(k) = K_p \cdot (k) + K_i' \cdot T_s \sum_{j=1}^{k} e(j)$$

Equation 2. Doscrete equation

Where:

$$K_i = K_i' \cdot T_s$$

- T_s: Sample time.
- *e(k)*: Input error in the discrete time domain.
- *u(k)*: Controller output in the discrete time domain.
- K_p: Proportional gain in the discrete time domain.
- K_i: Integral gain in the discrete time domain.

The speed PI controller routine is executed in the `AppRun()` function which is called every 1 ms, the sample time (T_s) of PI controller is 1 ms.

One input of the PI controller is speed error (`w16Speed_error`), and the other input is pointer to the structure of the PI controller parameters `sSpeedPiParams` (K_p, K_i, ...). All these parameters are used in the PI controller function, **CTR_ControllerPI16**.

The output of the **CTR_ControllerPI16** function is `uw16Duty`, which is used to update PWM duty in FTM2 ISR.

K_p is configured to **10** and K_i is configured to **1** in this application. These parameters must be reconfigured if the speed scale or the motor is changed.

Figure 16 shows the PI control process by using the **CTR_ControllerPI16** function to get the duty cycle.

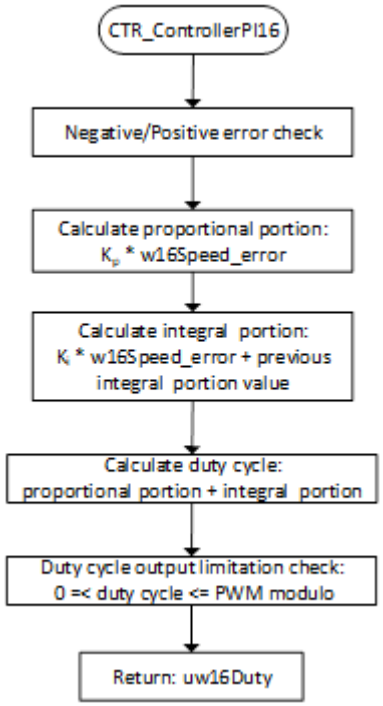

Figure 16. Flow chart of getting duty cycle

6.2.3 Limit current calculation

The DC-Bus current sensing circuit is in FRDM-DC-LVBLDC board, as shown in Figure 17.

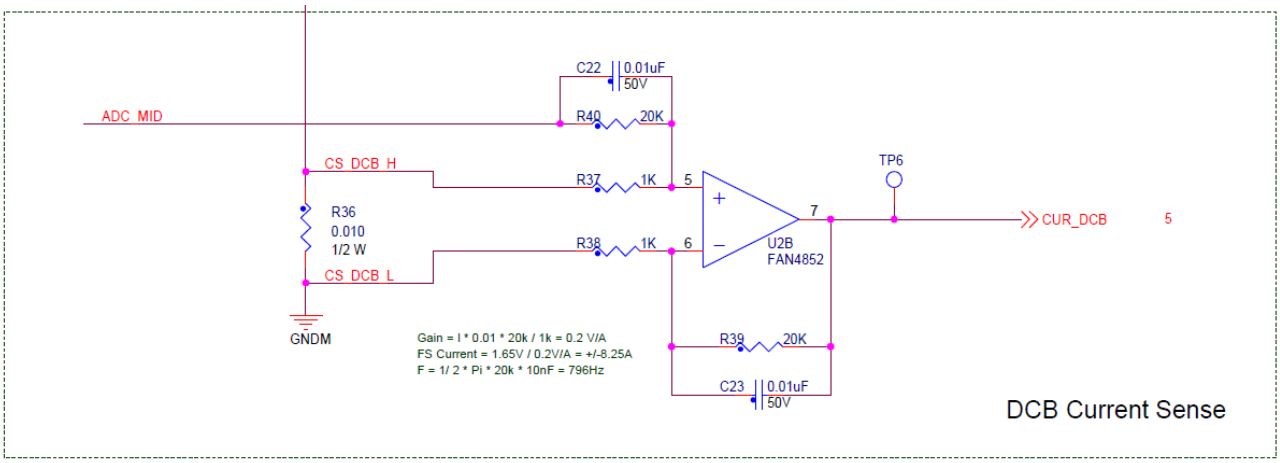

Figure 17. DC-Bus current sensing circuit

As shown in Figure 17, the reference voltage for the above OPAMP is 1.65 V, the gain is 0.2 V/A, and the output will connect to ACMP1+ input of S08PB16. If we the set limit current value to **i**, it follows the equation:

ACMP1 positive input voltage = 1.65 + 0.2 × *i*

According to the ACMP1 configuration in ACMP and FDS, we can know that DAC output, (***ACMP1_C1_DACVAL*** + 1) / 64 × Vin, is configured as ACMP1- input. ACMP + input should be smaller than ACMP- input. We can get the equation:

(***ACMP1_C1_DACVAL*** + 1) / 64 × Vin > 1.65 + 0.2 × *i*

Where:

- Vin: The reference voltage of DAC. Select VDDA (5 V) as the reference voltage in this application note.
- *i*: The desired limit current.

Therefore, the current limit value can be changed by changing the `ACMP1_C1_DACVAL` value according to the above equation.

6.3 FTM2 interrupt

Figure 18 shows the detailed process of the FTM2 interrupt service routine.

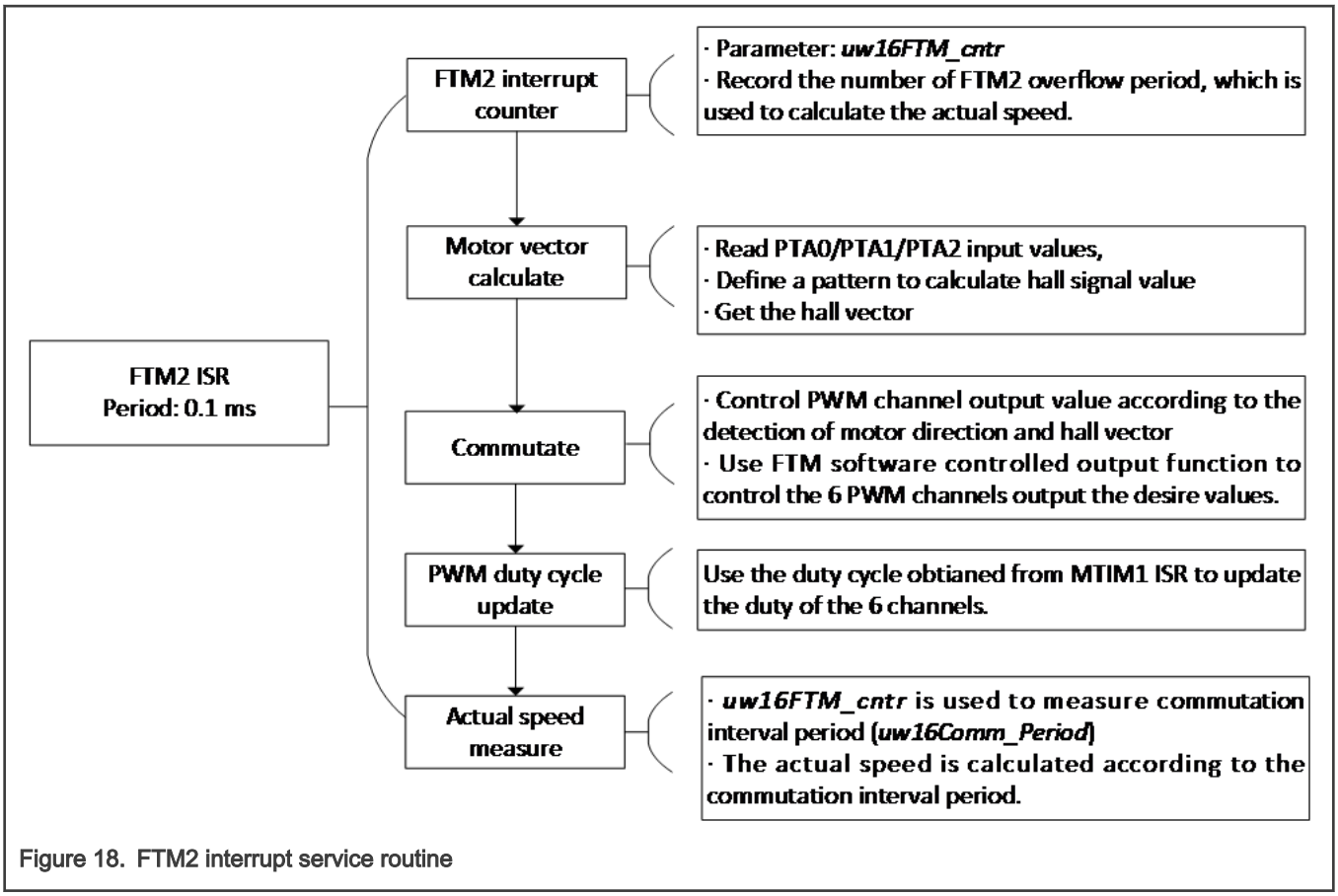

Figure 18. FTM2 interrupt service routine

As shown in Figure 18, the commutation and actual speed measurement are realized in FTM2 ISR, described in details as below.

6.3.1 Commutation

The definition of the correct commutation table is the key point of the application porting for the customer-specific BLDC motor.

In this application note, when the motor rotates clockwise and counterclockwise, the corresponding relationship between hall values and the power sequence of three phases is determined by manual test. Customers can obtain hall signals by voltage align or refer to motor manufacturer-provided commutation table.

This chapter gives a detailed description of how to implement commutation. Figure 19 shows the definition of the commutation table in this application note.

NOTE

The hall signals of different motors will change according to the different installation positions and 3-phase phase sequence. In actual use, we can use the open-loop align method to determine the hall signals corresponding to each rotor sector.

Hall pattern definition				Vector definition	Commutation definition **Clockwise direction**				Commutation definition **Counterclockwise direction**			
PTA2_ Hall3	PTA0_ Hall2	PTA0_ Hall1	HALL_ signal	Vector	Phase A	Phase B	Phase C		Phase A	Phase B	Phase C	
1	1	0	**6**	**1**	+	-	NC	**A+B-**	-	+	NC	**B+A-**
0	1	0	**2**	**2**	+	NC	-	**A+C-**	-	NC	+	**C+A-**
0	1	1	**3**	**3**	NC	+	-	**B+C-**	NC	-	+	**C+B-**
0	0	1	**1**	**4**	-	+	NC	**B+A-**	-	+	NC	**A+B-**
1	0	1	**5**	**5**	-	NC	+	**C+A-**	+	NC	-	**A+C-**
1	0	0	**4**	**6**	NC	-	+	**C+B-**	NC	+	-	**B+C-**

Figure 19. Commutation definition

The hall sensors are installed at a 120° electrical degree apart in our used BLDC motor, they detect the rotor flux. Combining the outputs of all the three sensors that will give 6 status, except 111 and 000. The commutation is repeated per 60° electrical degrees. We defined a hall signal calculation pattern to get Vector, then execute the commutate process according to Vector value. The steps of commutation are:

- Hall signal calculation:

  Hall_signal = PTA2_Hall3 × 0x04 + PTA2_Hall3 × 0x02 + PTA2_Hall1 × 0x01

- Vector definition:

  The Vector value can be got by the defined array: vector_Table[HALL_signal] = {0,4,2,3,6,5,1,0}

- Commutation:

  Commutation is achieved by changing the energization direction of phase A, B, and C voltage according to the vector value, as shown in Figure 19.

Commutation provides the creation of a rotating magnetic field. The angle between the stator flux and the rotor flux should be kept as close to 90° as possible, to get the maximum torque. For the proper operation of a BLDC motor, it is necessary to keep the angle between the stator flux and rotor flux close to 90°, the real angle varies from 60° to 120°.

With six-step control, we get a total of six possible stator flux vectors. The stator flux vector must be changed at a certain rotor position. The rotor position is usually sensed by the six states of hall effect sensors. Each of the six states corresponds to a certain stator flux vector. As shown in Figure 19, all the six hall sensor states with corresponding stator flux vectors are illustrated in Figure 20.

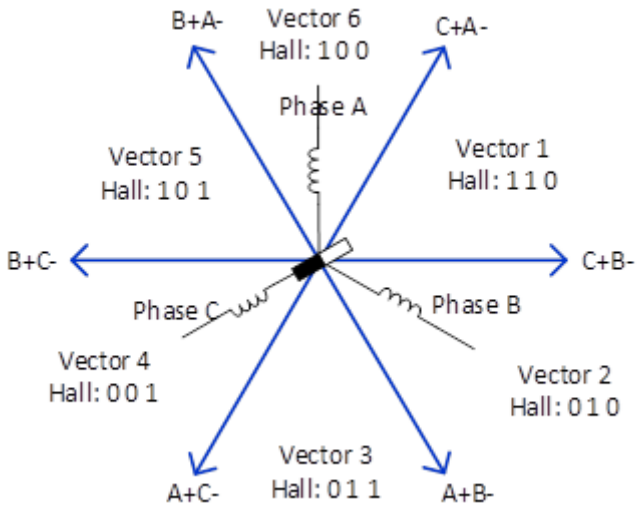

Figure 20. Stator flux vectors at six-step control

The rotor rotates in a direction that reduces the angle between the stator flux and rotor flux.

As shown in (a) in Figure 21, when the rotor is in Vector 1, A+B- commutation is performed. The rotor will run from Vector 1 to Vector 2 to reduce the angle between the stator and rotor flux to achieve clockwise rotation.

As shown in (b) in Figure 21, when the rotor is in Vector 1, B+A- commutation is performed. The rotor will run from Vector 1 to Vector 6 to reduce the angle between the stator and rotor flux to achieve counterclockwise rotation.

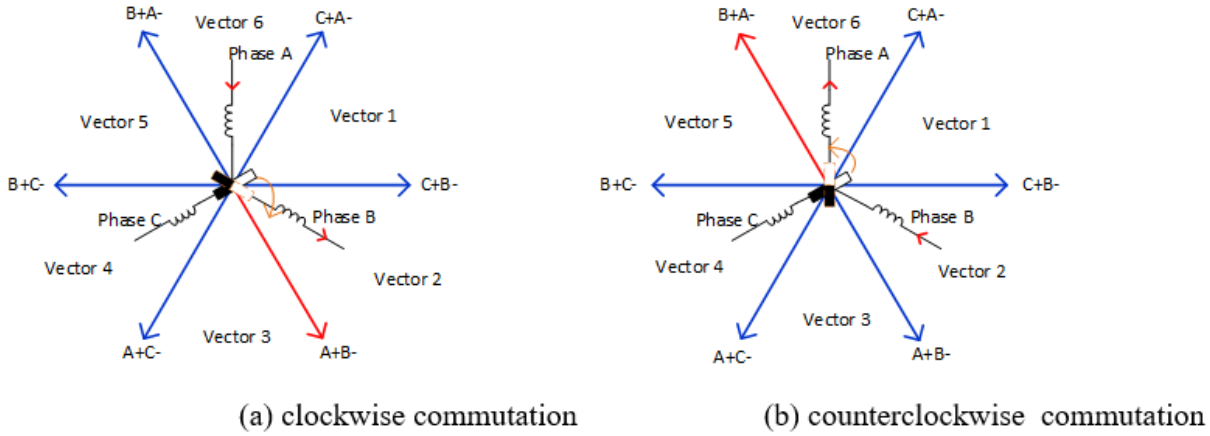

Figure 21. A commutation process when the rotor position is in vector 1

The 3-phase voltage is modulated by six PWMs. The modulation method is the `HPWM_LON` mode. If the rotating direction of the motor is clockwise, with the power sequence of (A+B-) -> (A+C-) -> (B+C-) -> (B+A-) -> (C+A-) -> (C+B-)), in one electrical angle cycle, the commutation process and the timing of the six PWMs are as shown in Figure 22.

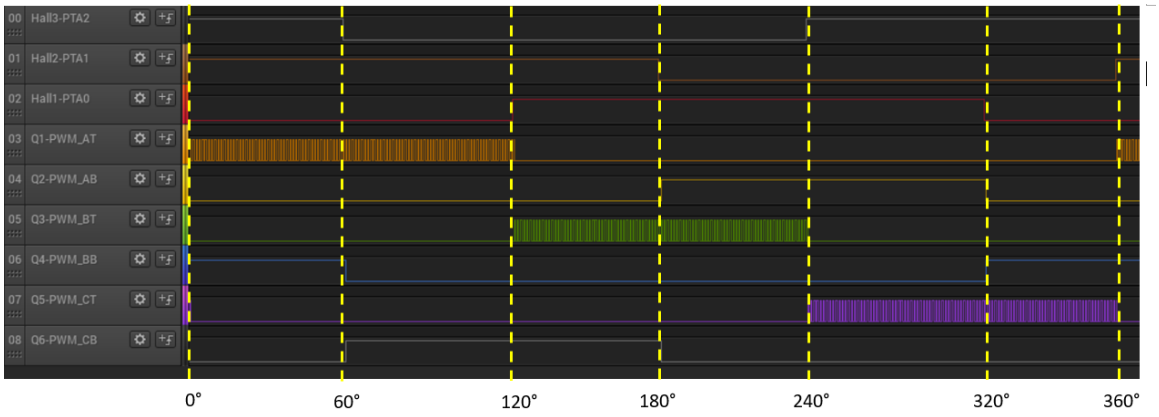

Figure 22. Commutation process and the timing of the 6 PWMs

6.3.2 Actual speed measure

All speed constants are scaled in this application code. All speed constants divided by the pre-defined maximum value, N_MAX = 100000 rpm, 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, *uw16Comm_Period*, and a scale constant *NUMERATOR_FOR_SPEED* which enables time value conversion to speed value. The actual speed *w16Speed_Act_flt* is the filtered value of *w16Speed_Act*. *w16Speed_Act* can be calculated with Equation 3.

$$w16Speed\_Act = \frac{NUMERATOR\_FOR\_SPEED}{uw16Comm\_Period}$$

Equation 3. Actual speed

NUMERATOR_FOR_SPEED is pre-defined as follows:

$$NUMERATOR\_FOR\_SPEED = ((PWM\_FREQUENCY\_KHZ*1000.0*60.0/6.0/PP)/N\_MAX)*32768.0$$

Where:

- `PWM_FREQUENCY_KHZ`: 10 kHz
- `N_MAX`: 100000 rpm
- `PP`: 2 (pole pairs)

7 Application guide

The BLDC motor control application can be controlled by the FreeMASTER with PC or use the button SW3 on the EVK board to control the speed.

Please strictly follow the steps below to set up hardware and software and then start the BLDC demo:

1. Set the jumper according to the position of the red mark in Figure 23.

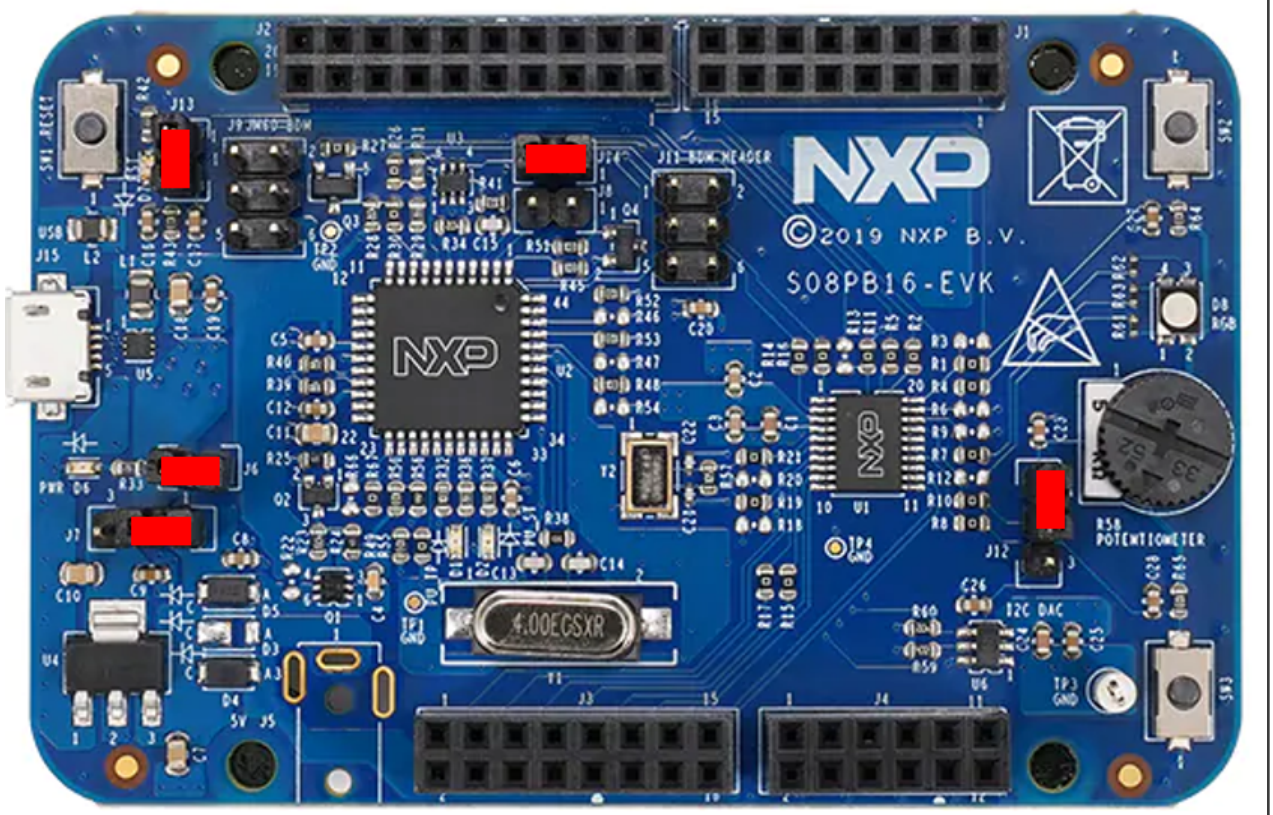

Figure 23. Jumper settings of S08PB16-EVK

2. Rework the S08PB16-EVK board. By default, PTA0/PTA1/PTA2/PTA3/PTB3 are connected to peripherals on the S08PB16-EVK board. But now, these pins need to connect to motor control signals from the FRDM-LVBLDC board. Remove 0 Ω resistances from R1/R4/R7/R10/R11and weld 0 Ω to R3/R6/R9/R12/R13. See Figure 24 for the circuit rework.

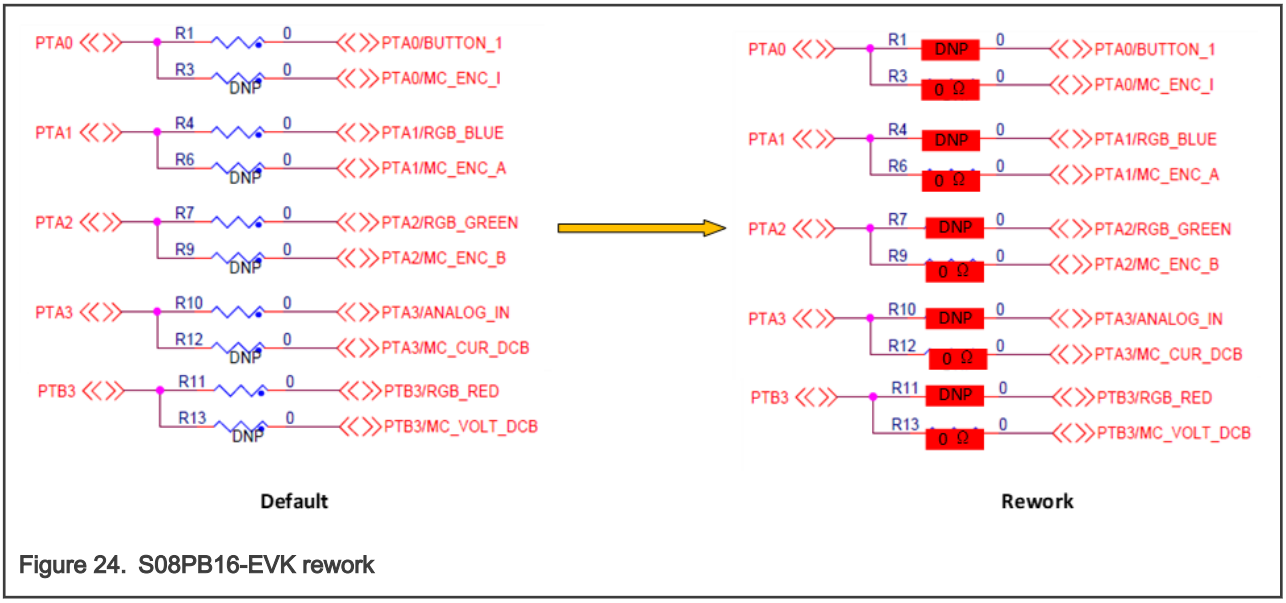

Figure 24. S08PB16-EVK rework

3. Combine the S08PB16-EVK board with FRDM-MC-LVBLDC board through the I/O headers.
4. Connect the motor three phase output and HALL signal to the FRDM-MC-LVBLDC board in order.
5. Connect the USB cable to the OSBDM port on the S08PB16-EVK board.
6. Open the project with CodeWarrior (11.1 version or above) then make and download the firmware to chip.
7. Open the **S08PB16_BLDC.pmpx** with FreeMASTER (3.0 version or above). Select **Project** -> **Options** -> **Comm** –> **Communications** -> **Plug-in Module** to select **FreeMASTER BDM** as communication. Click **Project** -> **Options** -> **MAP Files**. Select the *.abs* file as **Default symbol file** and **File format** as **Binary ELF with DWARF2 or DWARF4 dbg format**. The configurations are as shown in Figure 25.

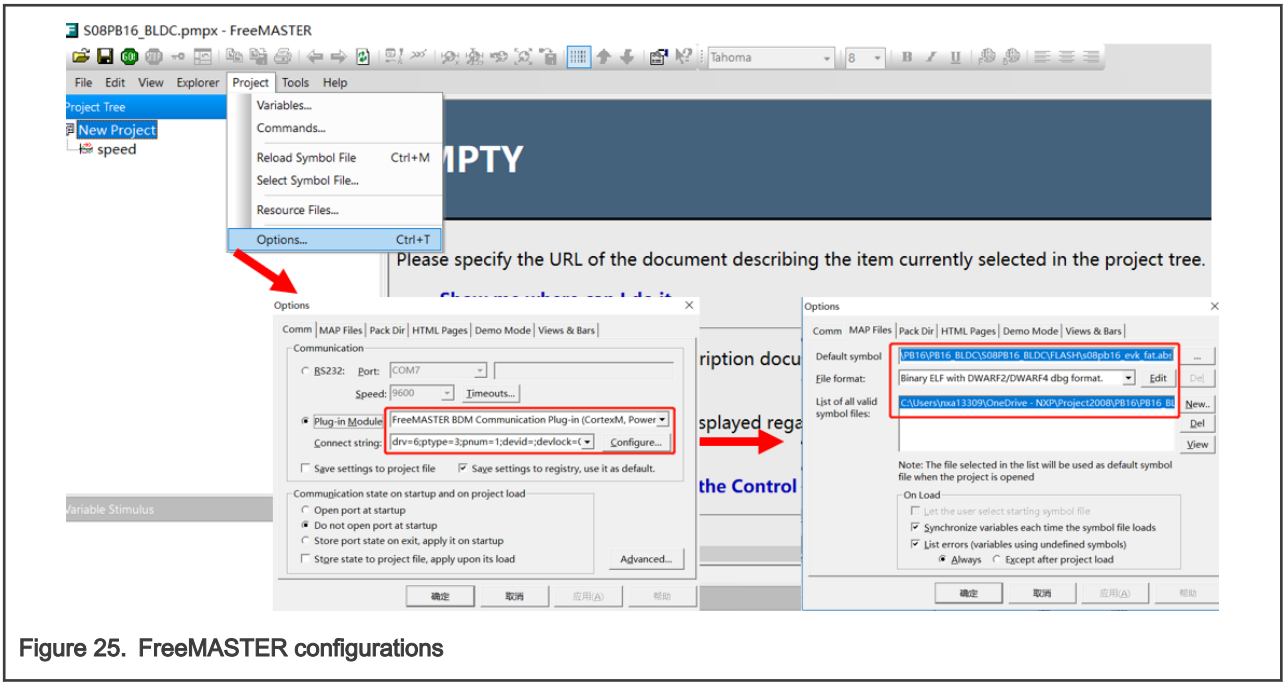

Figure 25. FreeMASTER configurations

8. Supply the 12 V DC voltage to the FRDM-MC-LVBLDC board and click the **GO** button on FreeMASTER. Set the speed command value(rpm) to **Speed_req** or drag the **Speed Required** slider to start the motor.

9. Figure 26 shows the FreeMASTER control page, which can be used for speed control, demo mode control, adjustment of limit current, and real-time monitoring of speed/DC-bus current/DC-bus voltage/state machine/variables.

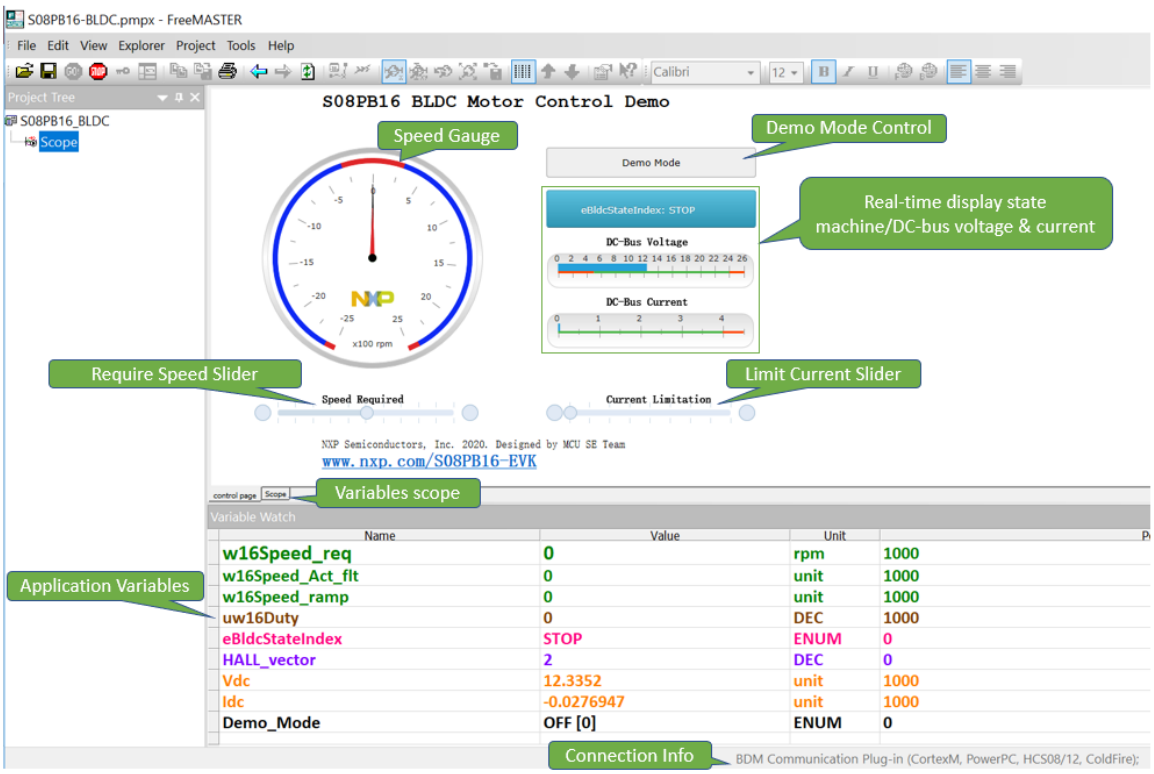

Figure 26. FreeMASTER control page

8 References

These references are available on https://www.nxp.com.

1. *MC9S08PB16 Reference Manual* (document MC9S08PB16RM)
2. *How to use the interconnection of OPAMP, ACMP1 and FDS for S08PB16* (document AN12836)
3. *BLDC Motor Control with Hall Sensors Driven by DSC* (document AN4413)
4. *BLDC Motor Control with Hall Effect Sensors Using MQX on Kinetis* (document AN4376)

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 09/2020

Document identifier: AN12957

arm