AN13040

How to use SCT to generate PWM and control BLDC motor on LPC51U68

Rev. 0 — 11/2020 Application Note

Contents

1 Introduction

The LPC51U68 based on Arm® Cortex®-M0+ is a low cost, low-power consumption, 32-bit Micro Controller Unit (MCU) family that operates at frequencies of up to 100 MHz and supports up to 256 KB on-chip flash memory and up to 96 KB total SRAM composed of up to 64 KB main SRAM, plus an additional 32 KB SRAM. The on-chip peripherals in LPC51U68 includes:

- One DMA controller
- 48 General-Purpose I/O (GPIO) pins
- One CRC engine
- One 12-bit ADC
- One 32-bit Real-Time Clock (RTC)
- One multiple-channel Multi-Rate 24-bit Timer (MRT)
- One Windowed Watchdog Timer (WWDT)
- Eight Flexcomm interfaces which can be selected by software to be a USART, SPI, or I²C interface

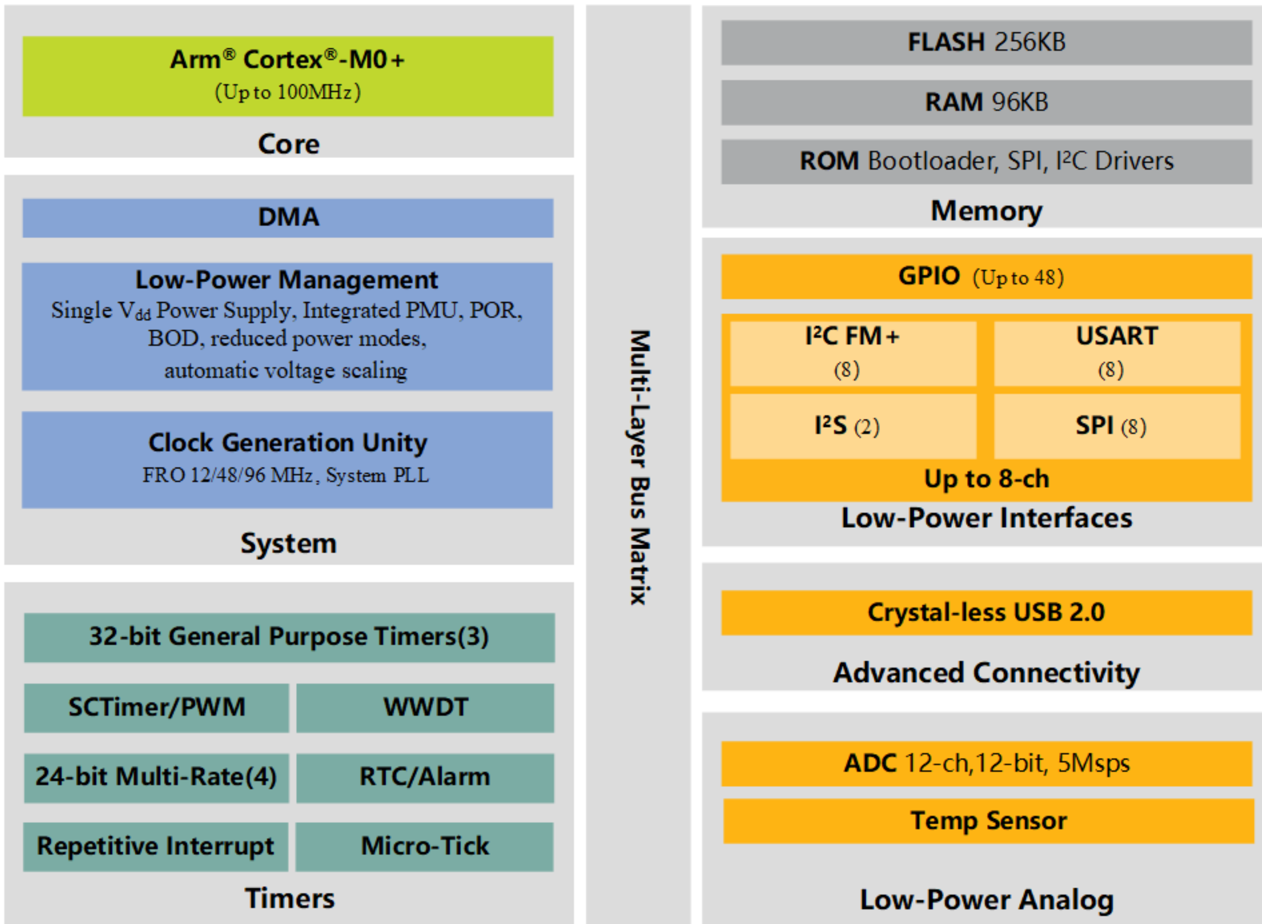

Figure 1. LPC51U68 block diagram

As the peripheral only available in NXP MCUs, the state configurable timer (SCTimer/PWM) is available on all LPC51U68 devices. It can work like traditional timer as a timer/counter. However, unlike traditional timer, a state machine is introduced to SCTimer/PWM. This is the most outstanding feature of SCTimer/PWM and it greatly enhances the configuration and control flexibility of LPC devices. Thanks to this feature, SCTimer/PWM is allowed to support multi-channel PWM with dead-time.

This application note demonstrates how to generate PWM with dead-time and implements 3-phase BrushLess Direct Current (BLDC) motor control including motor position detection based on Hall sensor and six-step commutation based on LPC51U68.

2 Brushless direct current motor composition

BLDC consists of a permanent-magnet rotor with a three-phase stator. Therefore, the winding as an electromagnet is wound on the stator to remain stationary while the rotor rotates as a permanent magnet.

BLDC, as the name suggests, is electronic commutation instead of mechanical brush commutation used by brushed direct current motor. In BLDC control, power should be supplied to the appropriate stator windings according to the position of the rotor to generate a proper magnetic field to ensure continuous motor rotation. Therefore, accurate rotor position is a key factor for the smooth and continuous rotation of the BLDC motor. In many applications, Hall-effect sensors are used to detect the rotor position and the microcontroller implements electronic commutation based on the position output of the Hall-effect sensor.

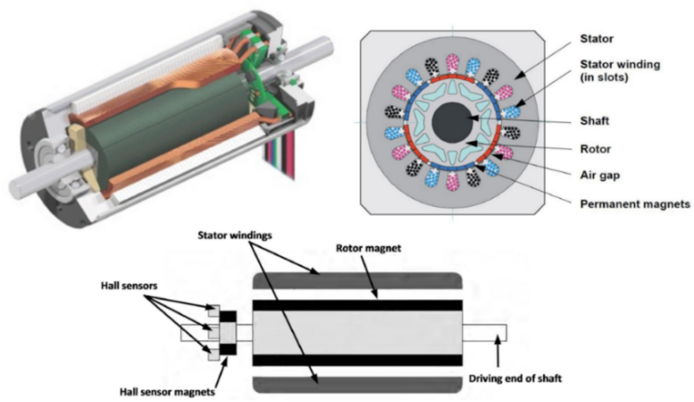

Figure 2. BLDC motor composition

3 Brushless direct current motor control principle

3.1 BLDC power output stage

Motor control system usually needs to provide power outputs to drive the motor. However, the standard TTL/CMOS level signal generated by the microcontroller is not enough to drive the motor directly. Therefore, in most motor drive schemes, the microcontroller just generates control signals to drive the power output stage, and the power output stage drives the motor to rotate.

Figure 3 shows a basic three-phase motor control topology. The motor control topology is a three-phase inverter which is composed of six N-channel enhanced FETs(Field Effect Transistor) including Q1T, Q1B, Q2T, Q2B, Q3T, Q3B.

- `PWM_1P` and `PWM_1N` are for phase A and are applied on Q1T and Q1B.
- `PWM_2P` and `PWM_2N` are for phase B and are applied on Q2T and Q2B.
- `PWM_3P` and `PWM_3N` are for phase C and are applied on Q3T and Q3B.

The three pairs of complementary PWMs are all from MCU. If voltage applied on N-channel enhanced FET is high level, the FET is turned on and if voltage applied on N-channel enhanced FET is low level, the FET is turned off.

PWM modulation modes applied on BLDC power output stage can be divided into three types: restricted unipolar, unipolar, bipolar. There are different control waveforms and characteristics among these three modes. Topology structure and forward/reverse current flow for restricted unipolar, unipolar, bipolar are shown in Figure 4, Figure 5, and Figure 6. Table 1 makes a comparison of the advantages and disadvantages of these three modes.

As shown in Figure 4, Figure 5, and Figure 6, topology of driving circuit is like English alphabet H. Therefore, it is also called as H-bridge driving circuit. Restricted unipolar mode applies PWM on single side top or bottom transistor and other side bottom or top transistor is on constantly. Unipolar mode applies complementary PWMs on single side two transistors and other side bottom transistor is on constantly. Bipolar mode applies one PWM on Q1 and Q4, other PWM on Q2 and Q3. These two PWMs are complementary.

NOTE

This document uses unipolar mode to drive motor.

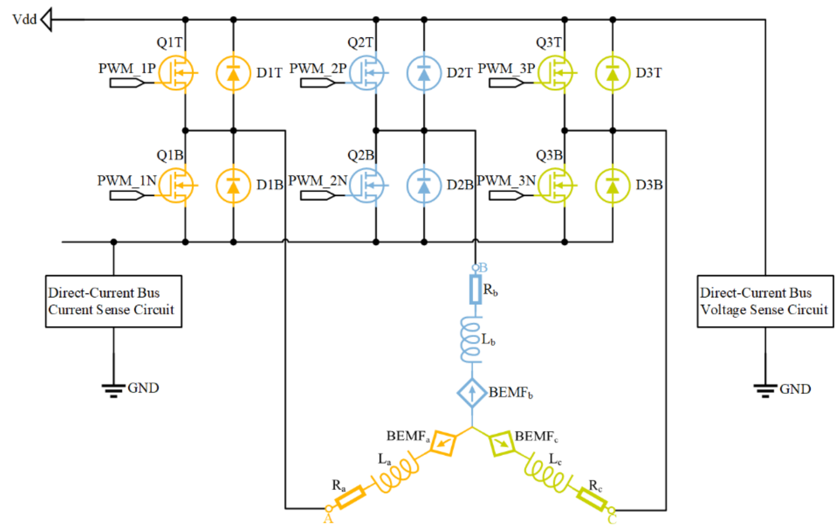

Figure 3. BLDC motor power output stage

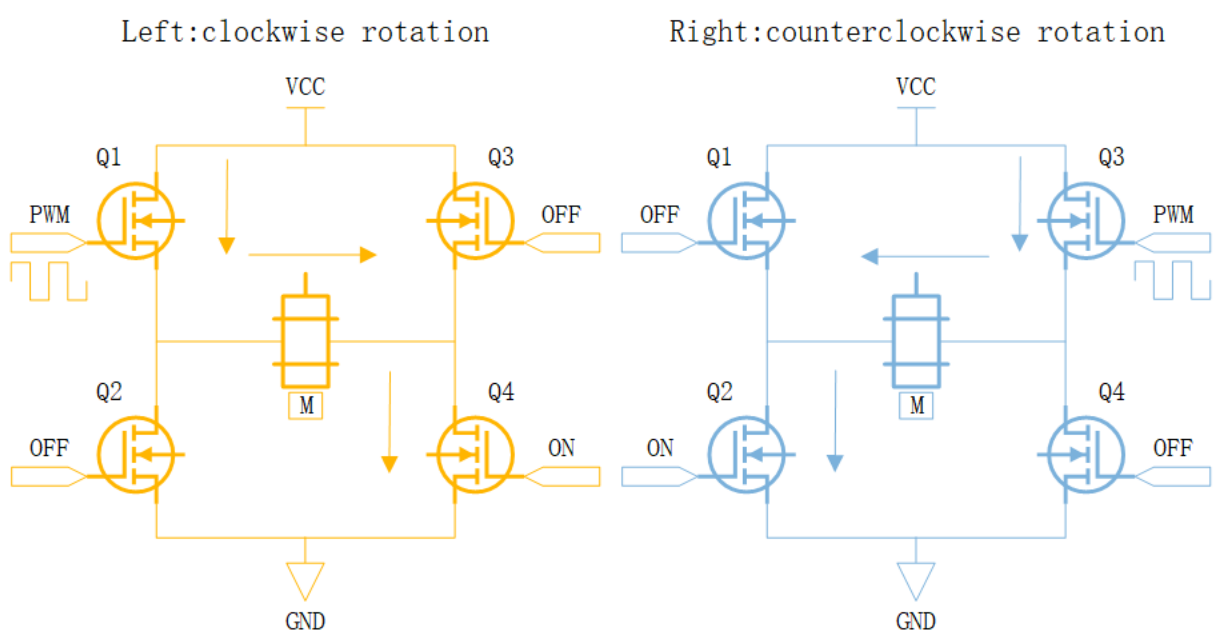

Figure 4. Restricted unipolar mode

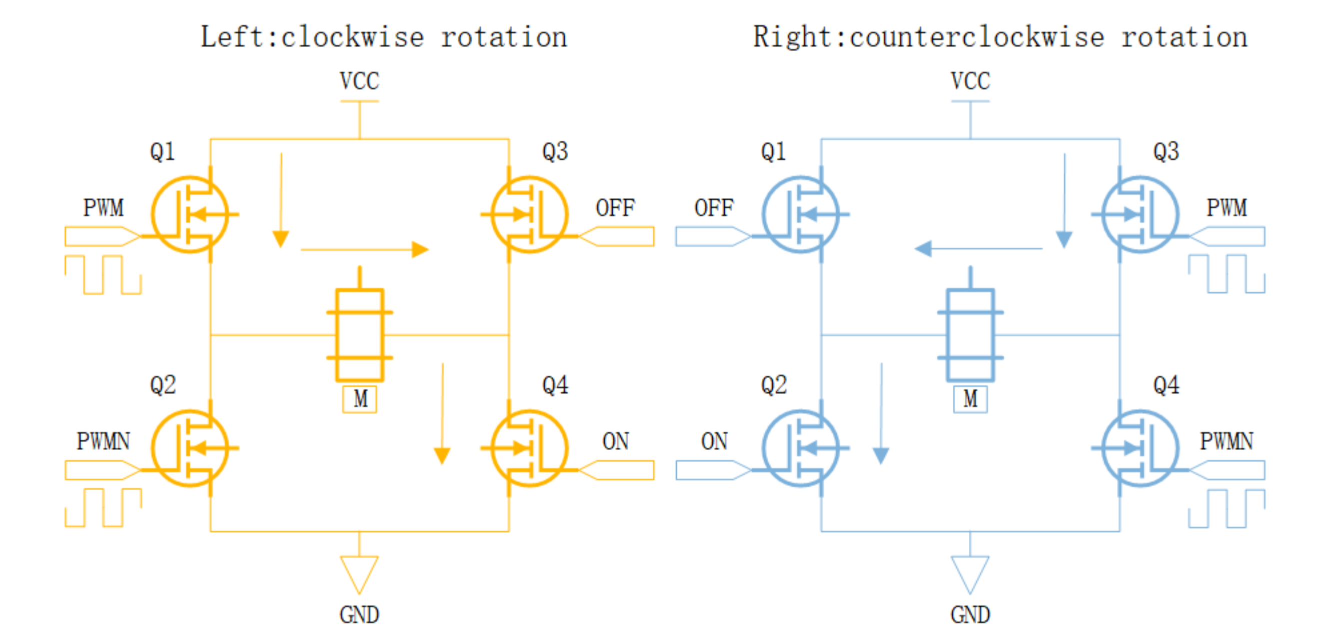

Figure 5. Unipolar mode

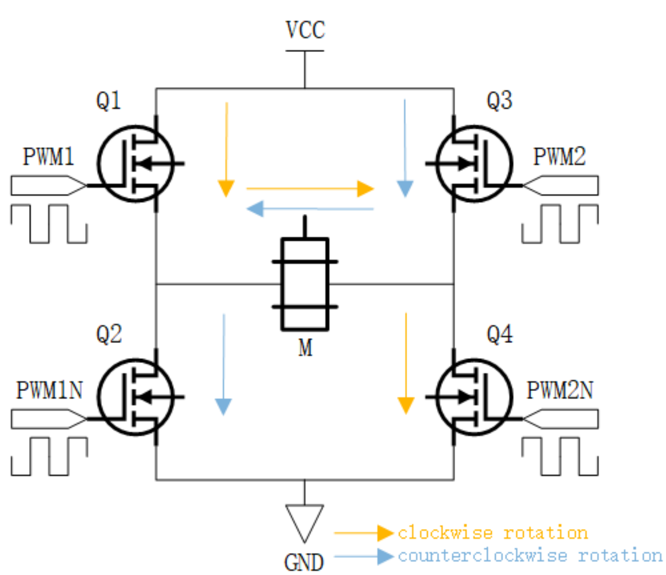

Figure 6. Bipolar mode

Table 1. Power output stage PWM modulation mode characteristics

Mode	Advantage	Disadvantage
Restricted unipolar	Simplify control	1. No brake 2. No energy consumption brake 3. No reverse torque when the load exceeds the set speed

Table continues on the next page...

Table 1. Power output stage PWM modulation mode characteristics (continued)

Mode	Advantage	Disadvantage
		4. Large static difference 5. Poor speed control performance 6. Poor stability
Unipolar	1. Fast start-up 2. Acceleration 3. Brake 4. Energy consumption brake 5. Energy feedback	1. No brake when speed is close to 0 2. Poor dynamic performance
Bipolar	1. Clockwise or counterclockwise rotation 2. Fast startup 3. High precision of speed regulation 4. Good dynamic performance 5. Small static difference of speed regulation 6. Wide range of speed regulation 7. Acceleration, slowing down 8. Brake 9. Reverse torque when the load exceeds preset speed 10. The static friction overcoming of motor bearing 11. Very low rotation rate	1. Complex control 2. Large power consumption

In order to generate the specified phase current, the MOS FETs of the BLDC power output stage need to be turned on and off properly. Table 2 describes the relationship between phase current and power output stage MOS FET.

Table 2. Relationship between phase current and MOS FETs of power output stage

Phase current[1]	Q1T[2]	Q1B	Q2T	Q2B	Q3T	Q3B
AB	PWM_1P[3]	PWM_1N	Low	High	Low	Low
AC	PWM_1P	PWM_1N	Low	Low	Low	High
BA	Low	High	PWM_2P	PWM_2N	Low	Low
BC	Low	Low	PWM_2P	PWM_2N	Low	High
CA	Low	High	Low	Low	PWM_3P	PWM_3N
CB	Low	Low	Low	High	PWM_3P	PWM_3N

1. Phase current direction rule is that the first phase is input and the second phase is output. For example, AB means that phase A is input and phase B is output.
2. Low means that apply low level on QxT or QxB and High means that apply high level on QxT or QxB.
3. `PWM_xP` and `PWM_xN` are complementary PWM pair and applied on QxT and QxB respectively.

3.2 Six-step commutation

Correct commutation is a key point for smooth and continuous rotation of the BLDC motor. In order to achieve correct commutation, two conditions should be met.

1. Precise commutation point is a must and this can be achieved by reading Hall sensor outputs.
2. It is important to make sure the commutation table is correct.

The commutation table describes the relationship between Hall sensor outputs and power sequence of three phases. Usually the commutation table is manually tested by the developer.

Three Hall sensors are installed at the position that is close to the rotor poles at 120 degrees of electrical angle in pairs, as shown in Figure 7. They determine the rotor position by detecting the rotor magnetic flux. The outputs of these three Hall sensors can be combined in six states except 000 and 111. The six states divide the 360-degree electrical angle into six sectors, as shown in Figure 8. Where, the Hb sensor is under the **N** pole of the permanent magnet and it outputs 1 signal. However, H_a and H_c are under the **P** pole of the permanent magnet and outputs 0 signal.

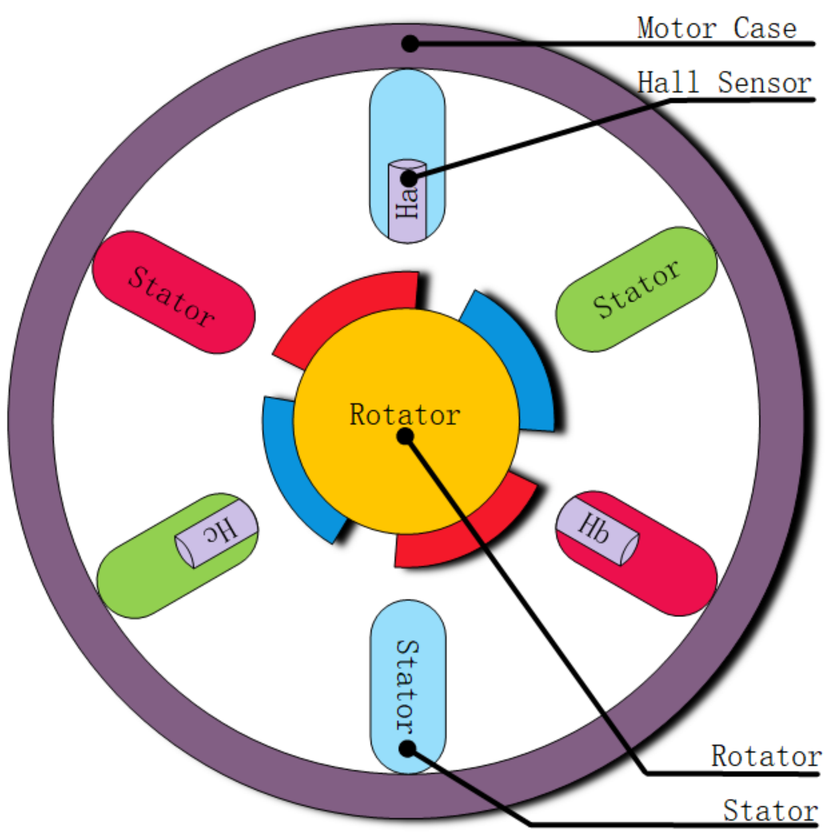

Figure 7. Hall sensor installation location

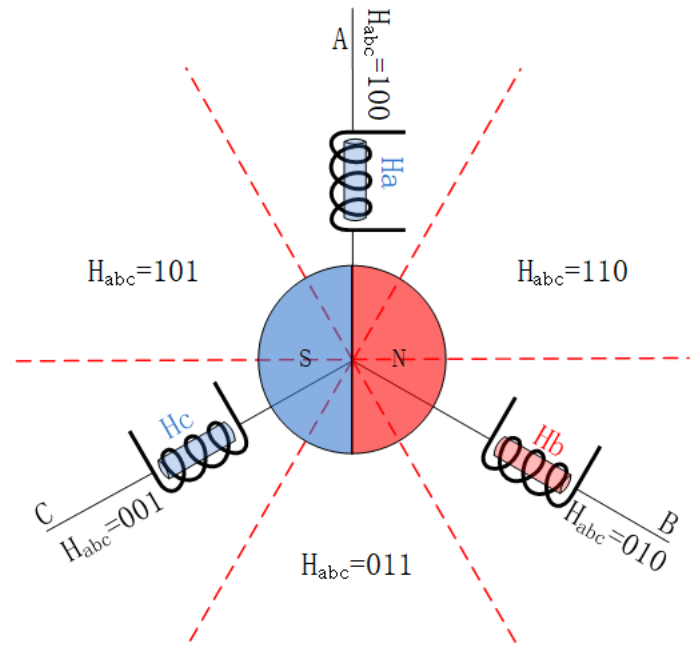

Figure 8. Hall sensor outputs

Figure 9 shows an example for BLDC commutation. Now, the combined state of the Hall sensors is `010`. For example, if the rotor rotates clockwise, it is crucial to generate magnetic field to move rotator from sector where the combined state of the Hall sensors is `010` to sector where the combined state of the Hall sensors is `011`. The magnetic field in this direction can be generated when the current flows into phase A and flows out of phase C. According to this principle, if the rotor rotates counterclockwise, it is necessary to generate the current which flows into phase C and flows out of phase A.

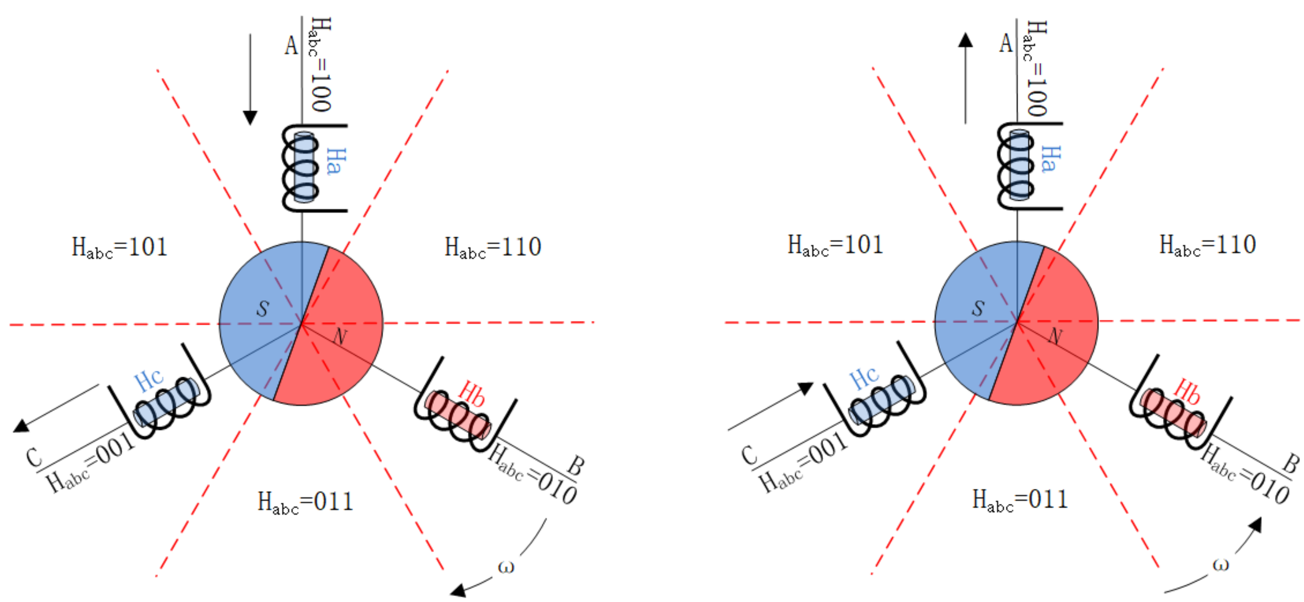

Figure 9. Commutation example

After understanding the BLDC commutation principle, if the BLDC rotates clockwise, power-on sequence of the phase A, B and C windings is AC -> BC -> BA -> CA -> CB -> AB -> AC and if the BLDC rotates counterclockwise, power-on sequence of the phase A, B and C windings is AC <- BC <- BA <- CA <- CB <- AB <- AC.

Finally, through the analysis of BLDC commutation, we can draw the following conclusions:

- The rotor rotates one circle and passes through a 360 degree of electrical angle.
- The winding excitation current needs to be changed every 60 degree of the rotor. This operation is also called as commutation. Therefore, 360-degree electrical angle undergoes a total of 6 commutations.
- The commutation point corresponds to the change of the Hall sensor output state.
- At a certain moment, only two-phase windings are powered, and the other phase winding is not powered.

3.3 Commutation table

The commutation table is a must so that the microcontroller can provide control on the BLDC. As mentioned above, the commutation table describes the relationship between the combined output value of the Hall sensors and the winding excitation.

Table 3. Commutation table for clockwise rotation

Phase ID	Hall sensors[1]			Phase[2]		
	a	b	c	A	B	C
1	0	1	1	NC[3]	+[4]	-[5]
2	0	0	1	-	+	NC
3	1	0	1	-	NC	+
4	1	0	0	NC	-	+
5	1	1	0	+	-	NC
6	0	1	0	+	NC	-

1. It is from Hall sensors output.
2. Phase value can be +, -, NC.
3. Symbol **NC** means that not connected, in other words, no voltage is applied on this phase.
4. Symbol **+** means that the current flows into this phase.
5. Symbol **-** means that the current flows out of this phase.

3.4 PWM waveform

As shown in Figure 3, the 3-phase currents are controlled by six-channel PWMs, including `PWM_1P`, `PWM_1N`, `PWM_2P`, `PWM_2N`, `PWM_3P`, `PWM_3N`. The PWM modulation mode is unipolar. If the rotating direction of the motor is clockwise, the power sequence for phase A,B,C is **AB** -> **AC** -> **BC** -> **BA** -> **CA** -> **CB**. Figure 10 shows the commutation process and the timing of the six-channel PWMs are as shown in . If the rotating direction of the motor is counterclockwise, the power sequence for phase A,B,C is **CB** -> **CA** -> **BA** -> **BC** -> **AC** -> **AB**. Figure 11 shows the commutation process and the timing of the six-channel PWMs. Figure 12 shows the dead time of six-channel PWMs.

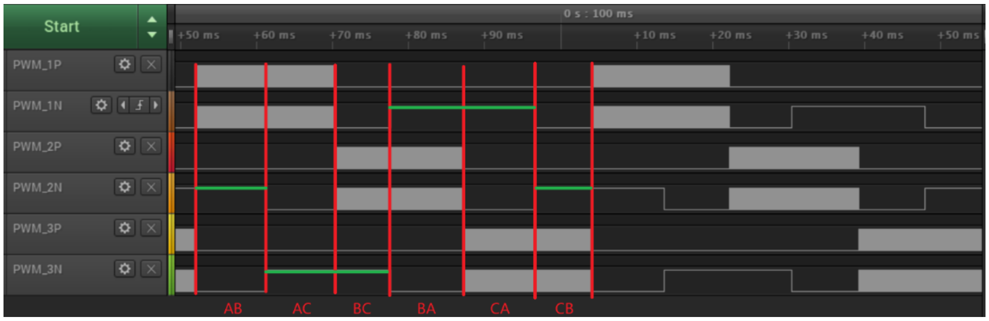

Figure 10. PWM waveform for clockwise rotation

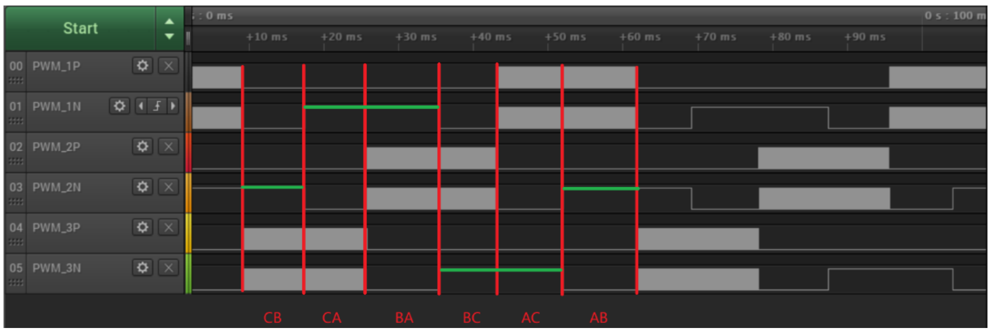

Figure 11. PWM waveform for counterclockwise rotation

Figure 12. PWM waveform with dead time

4 System hardware overview

System hardware of LPC51U68 BLDC application consists of three parts:

- One LPCXpresso51U68 (OM40005) evaluation board
- One FRDM-MC-LVBLDC Freedom development board
- One BLDC

The LPCXpresso51U68 (OM40005) evaluation board has been designed to enable evaluation of and prototyping with the LPC51U68 MCU leveraging the high-performance Cortex-M0+ core, power-efficiency and cost sensitive capabilities. The board also features an on-board CMSIS-DAP/SEGGER J-Link compatible debug probe.

The FRDM-MC-LVBLDC Freedom development board implements a 3-phase BLDC interface platform that adds BLDC motor control capabilities, such as rotational or linear motion, to user’s applications.

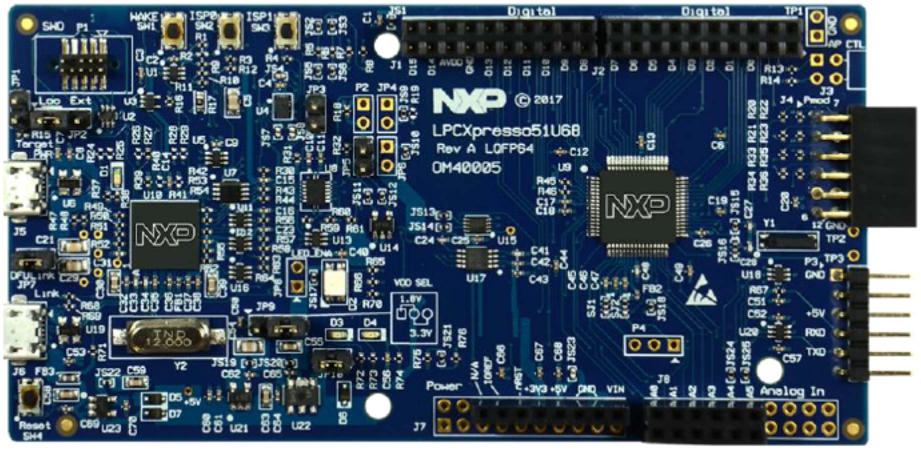

Figure 13. LPCXpresso51U68 evaluation board

Figure 14. FRDM-MC-LVBLDC driver board

Product model of the BLDC motor used in this document is 45ZWN24-40 from LINIX Motor Co., Ltd. This is a motor with a supply voltage up to 24 V and a power up to 40 W.

Figure 15. BLDC motor

Block diagram and actual hardware of LPC51U68 BLDC development platform are shown in Figure 16 and Figure 17. Symbol **PA**, **PB** and **PC** means phase A, phase B and phase C, as shown in Figure 16.

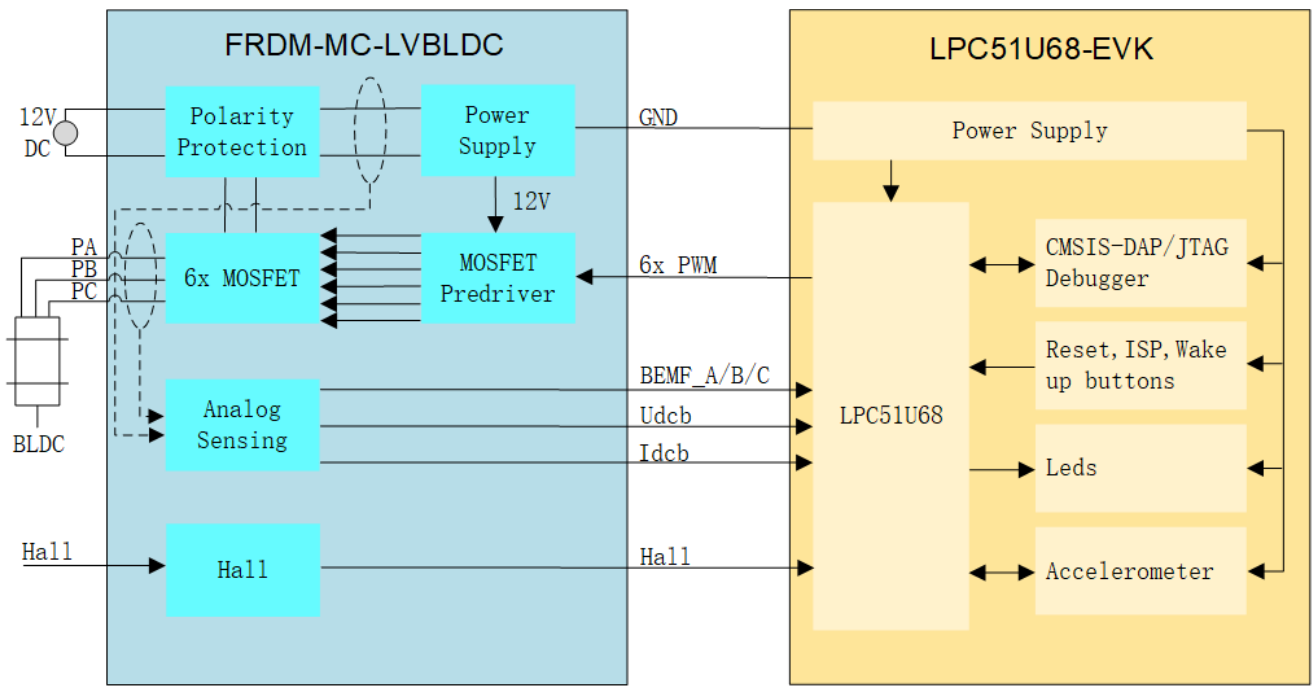

Figure 16. LPC51U68 BLDC development platform block diagram

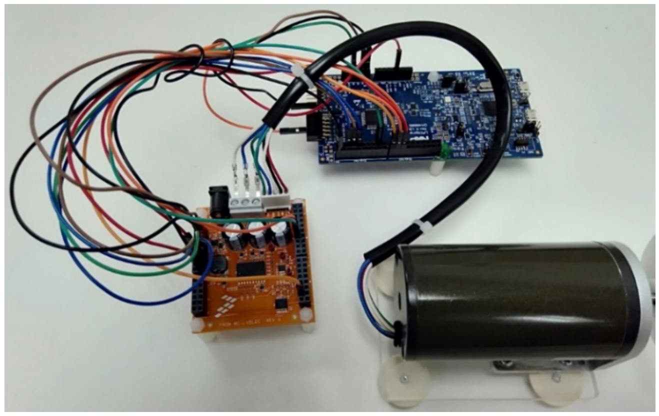

Figure 17. LPC51U68 BLDC development platform actual hardware

The IO pin assignment for LPC51U68 BLDC application and the hardware connection relationship between LPC51U68 evaluation board and FRDM-MC-LVBLDC Freedom development board are shown in Table 4, Table 5, and Figure 18.

Table 4. Connection between LPCXpresso51U68 and FRDM-MC-LVBLDC

Function	LPCXpresso51U68		FRDM MC LVBLDC	
	Connector	Function	Connector	Function
PWM A Top (PWM_1P)	J1-16	PIO0_7 (SCT0_OUT0)	J3-15	PWM_AT
PWM A Bottom (PWM_1N)	J2-15	PIO0_8(SCT0_OUT1)	J3-13	PWM_AB
PWM B Top (PWM_2P)	J2-13	PIO0_9(SCT0_OUT2)	J3-11	PWM_BT
PWM B Bottom (PWM_2N)	J2-11	PIO0_10(SCT0_OUT3)	J3-9	PWM_BB
PWM C Top (PWM_3P)	J8-7	PIO1_2(SCT0_OUT5)	J3-7	PWM_CT
PWM C Bottom (PWM_3N)	J2-20	PIO1_3(SCT0_OUT6)	J3-5	PWM_CB
Hall-A	J8-1	PIO0_2(PME_IN)	J3-3	ENC_A
Hall-B	J8-3	PIO0_3(PME_IN)	J3-1	ENC_B
Hall-C	J1-20	PIO0_5(PME_IN)	J1-3	ENC_I
Volt. DCB	J2-3	PIO1_0(ADC0_3)	J2-7	VOLT_DCB
Current DCB	J1-15	PIO1_1(ADC0_4)	J2-9	CUR_DCB
3V	J7-12	3V VCC	J3-8(4)	3V VCC
GND	J7-16(18)	GND	J3-12(14)	GND

Table 5. Connection between LPCXpresso51U68 and LPCXpresso51U68

Function	LPCXpresso51U68		LPCXpresso51U68	
	Connector	Function	Connector	Function
PME_OUT/SCT_IN0	J1-18	PIO0_6 (GPIO)	J4-9	PIO0_23(GPIO)

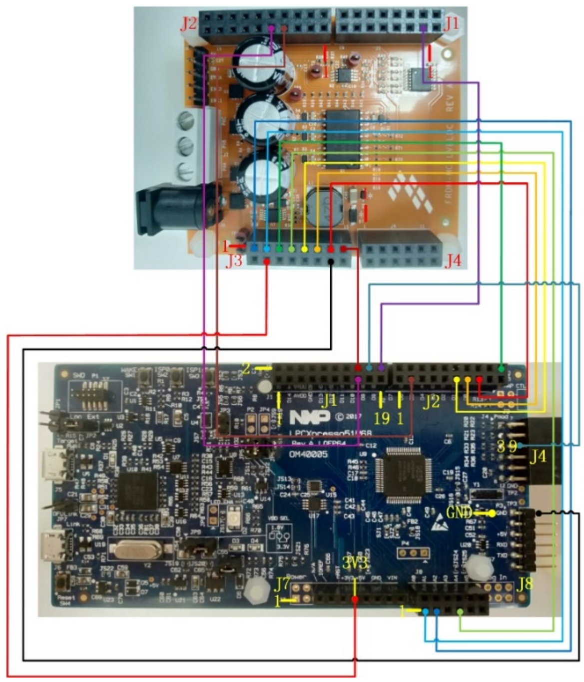

Figure 18. LPC51U68 evaluation board and FRDM-MC-LVBLDC driver board actual connection

5 System software overview

System software of LPC51U68 BLDC application can drive the motor to achieve the following functional requirements:

- Start and Stop.
- Rotating direction control. Motor is able to rotate clockwise and counterclockwise.
- Speed control. Motor speed can be regulated from 500 rpm to 2500 rpm. The speed range depends on the motor used.
- FreeMASTER is able to provide real-time monitoring of motor status including setting and displaying of start/stop, speed, rotating direction and other motor parameters.

The functional block diagram of this application combining hardware and software is as shown in Figure 19.

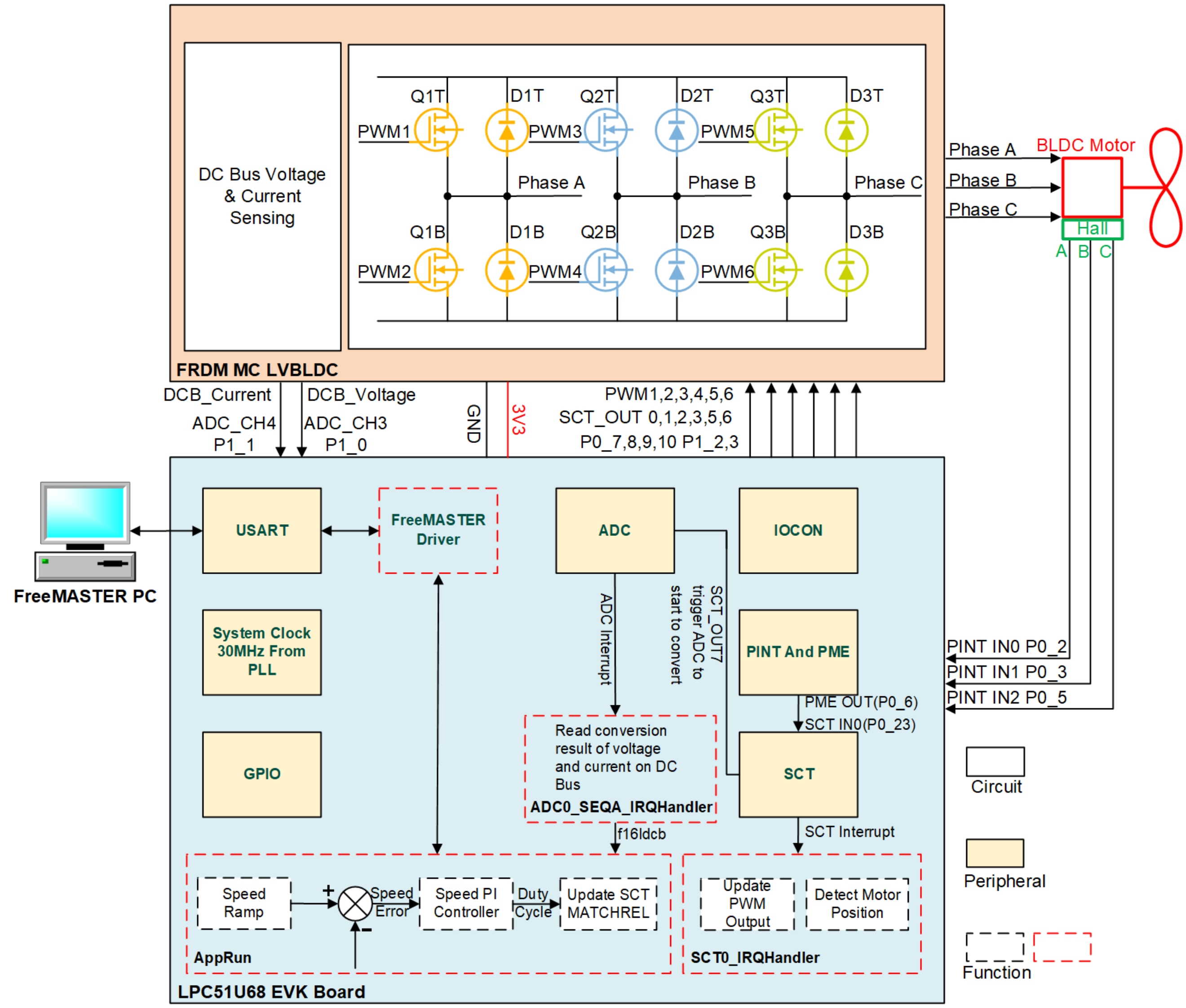

Figure 19. LPC51U68 BLDC functional block diagram

As shown in Figure 19, the overall control process of LPC51U68 BLDC application is mainly completed by the following three parts:

- `AppRun`

  This function regulates duty cycle of PWMs by obtaining the ramp speed so that the motor speed gradually reaches the preset target speed.

- `ADC0_SEQA_IRQHandler`

  `SCT_OUT7`, as shown in Figure 19, is the 7th output channel of SCTimer and it is used as the trigger source for ADC sequence conversion which only converts 3rd and 4th channel of ADC. Once a sequence conversion of ADC is done, an interrupt will be generated and the Interrupt Service Routine (ISR) named as `ADC0_SEQA_IRQHandler` will be called. You can obtain the digitized values of the direct current bus voltage and current in this ISR. These two digitized values are usually used to detect the overcurrent, overvoltage and undervoltage faults.

- `SCT0_IRQHandler`

  As mentioned above, the commutation is based on the rotor position detection using Hall sensor. Outputs of phase A,B and C of Hall sensor are connected to `P0_2`, `P0_3`, and `P0_5` pins of LPC51U68. `P0_2`, `P0_3`, and `P0_5` pins are configured as input source 0, 1 and 2 of Pattern Match Engine (PME) in LPC51U68. PME can detect rising and falling edges on the three pins. Once an edge is detected on any of these three pins, PME will send a signal which is a level or edge to SCT input 0. When SCT input 0 receives a level or edge signal, SCT will generate an interrupt and interrupt service routine named as `SCT0_IRQHandler` will be called to obtain motor position to complete commutation and update six PWM outputs.

6 SCTimer/PWM-based BLDC motor control

6.1 LPC51U68 SCTimer/PWM features

The SCTimer/PWM supports:

- Eight inputs
- Eight outputs
- Ten match/capture registers
- Ten events
- Ten states

Counter/timer features:

- Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
- Counters clocked by system clock or selected input.
- Configurable as up counters or up-down counters.
- Configurable number of match and capture registers. Up to 10 match and capture registers total.
- Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
- Counter value can be loaded into capture register triggered by a match or input/output toggle.

PWM features:

- Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals. PWM waveforms can change based on the current state.
- Up to 8 single-edge or 4 dual-edge PWM outputs with independent duty cycle and common PWM cycle length.

Event creation features:

- In bi-directional mode, events can be enabled based on the count direction.
- The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
- Selected events can limit, halt, start, or stop a counter or change its direction.
- Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.
- Match register 0 can be used as an automatic limit.
- In bi-directional mode, events can be enabled based on the count direction.
- Match events can be held until another qualifying event occurs.

State control features:

- A state is defined by events that can happen in the state while the counter is running.
- A state changes into another state as a result of an event.
- Each event can be assigned to one or more states.
- State variable allows sequencing across multiple counter cycles.

6.2 SCTimer/PWM-based PWM control state machine

As mentioned above, SCTimer/PWM contains a configurable state machine. This document uses the following state machine to complete BLDC motor control. This state machine is always available during the entire application run.

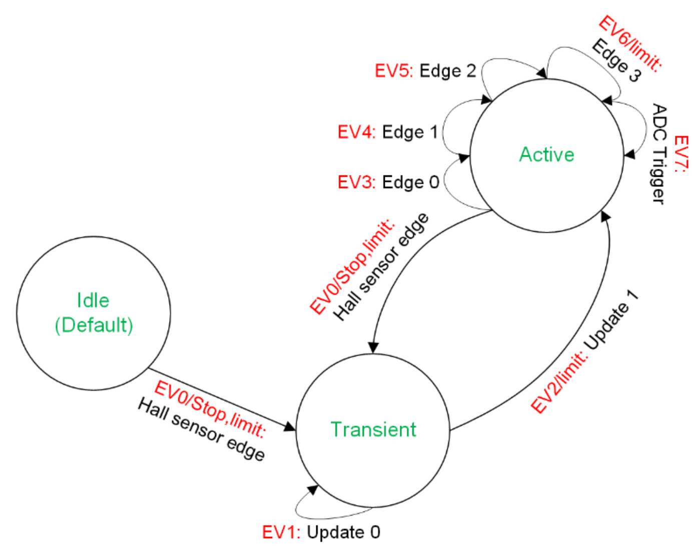

Figure 20. SCTimer/PWM-based PWM control state machine

As shown in Figure 20, this state machine consists of three states: Idle, Transient and Active, and eight events, EV0 to EV7.

- Idle

  The Idle state is the initial and default state.

- Active

  The Active state is used to generate 6-channel complementary PWM signals. Four events, EV3, EV4, EV5 and EV6, correspond to four edges, Edge 0, Edge 1, Edge 2 and Edge 3, respectively. Event labeled as EV6 is a limit event, which is used to determine the PWM period and the events labeled as EV4 and EV5 are used to determine the PWM duty cycle. In addition, Active state also contains an event labeled as EV7 which is used to trigger ADC conversion to obtain the direct current bus voltage and current.

  Event labeled as EV0 occurs when the PME detects an edge at one of the Hall sensor outputs. This event is available both in Idle and Active. SCTimer should stop counting and reset the count value to 0 to prepare for commutation when this event occurs. Therefore, EV0 event is a stop and limit event.

- Transient

  The Transient state is entered when EV0 event occurs and it is used to update six-channel SCTimer outputs in order to perform commutation. Two events are defined in this state. The first event labeled as EV1 is used to drive outputs that need to go low to become low. The second event labeled as EV2 is used to set one output that needs to go high to become high and to prepare six-channel PWM signals. This second event is also a limiting event and it is used to update the PWM duty cycle.

  When the state machine switches to the Transient state, SCTimer interrupt service routine named as `SCT0_IRQHandler` is called to update six-channel SCTimer outputs by configuring the SCT OUT SET/CLR registers. Once this update is done, state machine returns to Active state again.

6.3 SCTimer/PWM-based PWM waveform with dead time

It is essential to insert dead time for the complementary PWM signals to prevent the two MOS transistors which may be Q1T/Q1B or Q2T/Q2B or Q3T/Q3B from being turned on simultaneously. Once this happens, it will cause permanent and irreversible damage to the power output stage.

This document adopts edge-aligned mode to generate PWM with dead time and the waveform details are as shown in Figure 21.

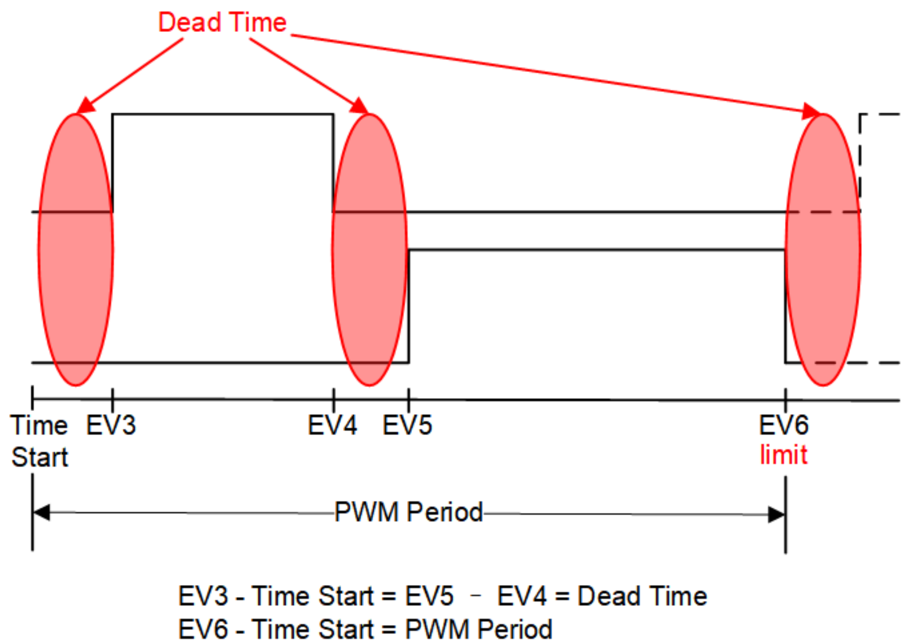

Figure 21. PWM waveform with dead time

PWM period is determined by EV6 event and EV6 is a limit event to reset SCTimer counter to 0. Time duration between EV3 and Time Start is equal to the one between EV5 and EV4. The two time durations are all dead time. PWM duty cycle is determined by EV4 and EV5. If you want to regulate PWM duty cycle, move EV4 position and EV5 must be equal to the sum of EV4 and dead time.

7 System software implementation

This section elaborates on the software design of the LPC51U68 BLDC application, including main program flow, key peripheral configuration and key function analysis.

7.1 Main program flow

After power-on reset, LPC51U68 begins to execute main program flow which is implemented by main function. Main function first initializes on-chip peripherals, FreeMASTER protocol and application state machine, then enters an infinite loop to periodically call the `FMSTR_Example_Poll` function to query the commands sent from the FreeMASTER tool on PC end. The main program flow is as shown in Figure 22.

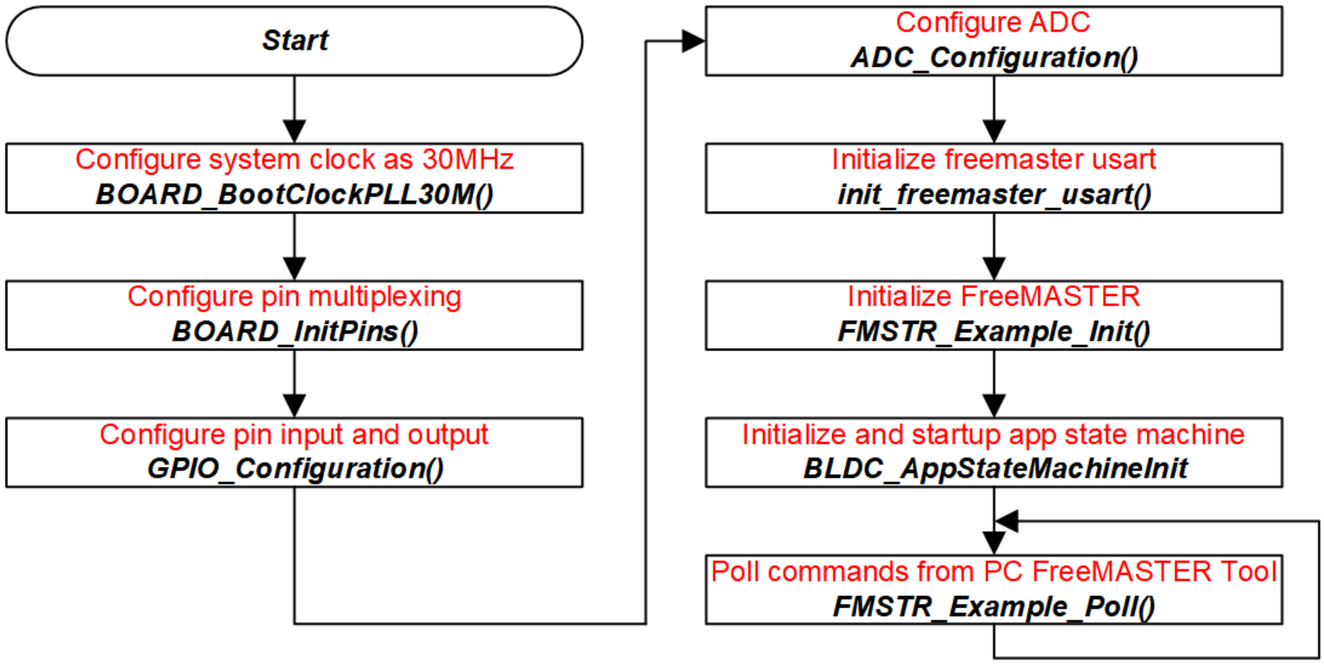

Figure 22. Main program flow

7.2 ADC configurations

ADC is triggered by SCTimer 7th output channel and converts direct current bus voltage on 3rd channel and current on 4th channel for overvoltage, undervoltage and overcurrent detection. Figure 23 shows the configurations for ADC sequence control register A and Figure 24 shows the ADC configuration flow.

```
ADC0->SEQ_CTRL[0] = ADC_SEQ_CTRL_CHANNELS(1<<3 | 1<<4) |  //ADC_IN3/4
                    ADC_SEQ_CTRL_TRIGGER(2)            |  //SCT_OUT7 hw trigger
                    ADC_SEQ_CTRL_TRIGPOL(1)            |  //trigger @ positive edge
                    ADC_SEQ_CTRL_SYNCBYPASS(0)         |  //enable synchronization
                    ADC_SEQ_CTRL_START(0)              |  //do not START
                    ADC_SEQ_CTRL_BURST(0)              |  //no BURST
                    ADC_SEQ_CTRL_SINGLESTEP(0)         |  //no SINGLESTEP
                    ADC_SEQ_CTRL_LOWPRIO(0)            |  //set SEQ A as high priority
                    ADC_SEQ_CTRL_MODE(1)               |  //retrieve data at the end of sequence
                    ADC_SEQ_CTRL_SEQ_ENA(1);              //enable sequence
```

Figure 23. ADC Sequence Control Register A configuration

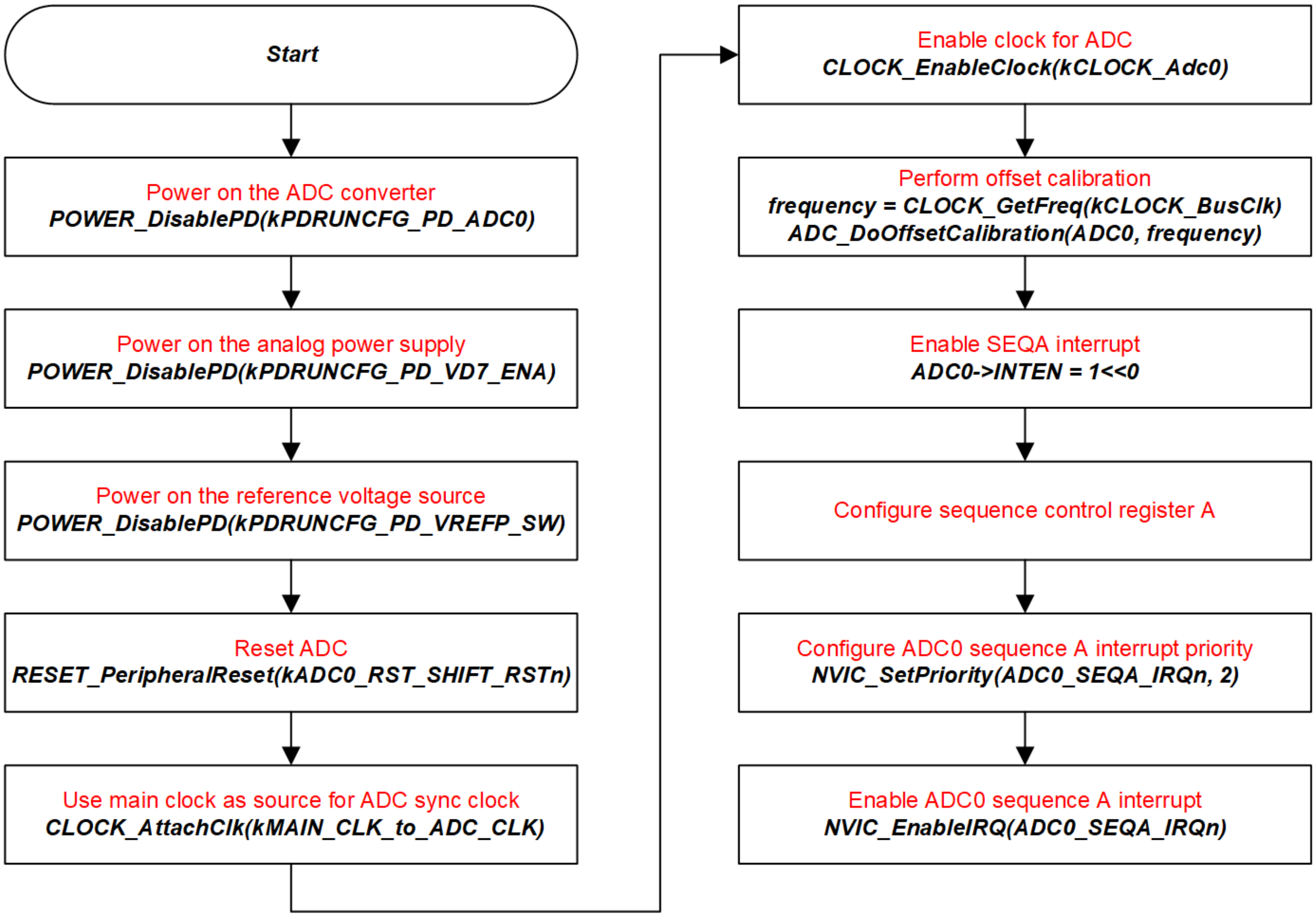

Figure 24. ADC configuration flow

7.3 Pin multiplexing and GPIO configurations

Pin multiplexing and GPIO configuration are performed by `BOARD_InitPins` and `GPIO_Configuration()` functions. Table 6 lists the pin function multiplexing and GPIO configuration of LPC51U68 BLDC application.

Table 6. Pin multiplexing and GPIO configurations

Pin	Function	Using in BLDC application	Input/Output (if it is GPIO)
P0_0	FC0_RXD_SDA_MOSI	USART RXD	—
P0_1	FC0_TXD_SCL_MISO	USART TXD	—
P0_2	PIO0_2	Hall Sensor Phase A	Input
P0_3	PIO0_3	Hall Sensor Phase B	Input
P0_5	PIO0_5	Hall Sensor Phase C	Input
P0_6	PIO0_6	PME Output	Output
P0_23	PIO0_23	SCTimer/PWM IN Channel 0	Input

Table continues on the next page...

Table 6. Pin multiplexing and GPIO configurations (continued)

Pin	Function	Using in BLDC application	Input/Output (if it is GPIO)
P0_7	SCT0_OUT0	SCTimer/PWM Out Channel 0	—
P0_8	SCT0_OUT1	SCTimer/PWM Out Channel 1	—
P0_9	SCT0_OUT2	SCTimer/PWM Out Channel 2	—
P0_10	SCT0_OUT3	SCTimer/PWM Out Channel 3	—
P1_2	SCT0_OUT5	SCTimer/PWM Out Channel 5	—
P1_3	SCT0_OUT6	SCTimer/PWM Out Channel 6	—
P1_0	ADC0_3	ADC IN Channel 3	—
P1_1	ADC0_4	ADC IN Channel 4	—
P0_21	PIO0_21	Debug	Output

7.4 Application state machine

LPC51U68 BLDC application state machine performs overall system control. Its roles are as follows:

- Motor start and stop control.
- Calculate the error between the actual speed and the target speed, and convert the error to PWM duty cycle, and finally calculate SCTimer match load register, `MATCHREL`, setting value according to PWM duty cycle and dead time.
- Configure PME.
- Configure SCTimer and startup SCTimer/PWM-based PWM control state machine, as described in SCTimer/PWM-based PWM control state machine.

Figure 25 shows the state transition diagram of application state machine.

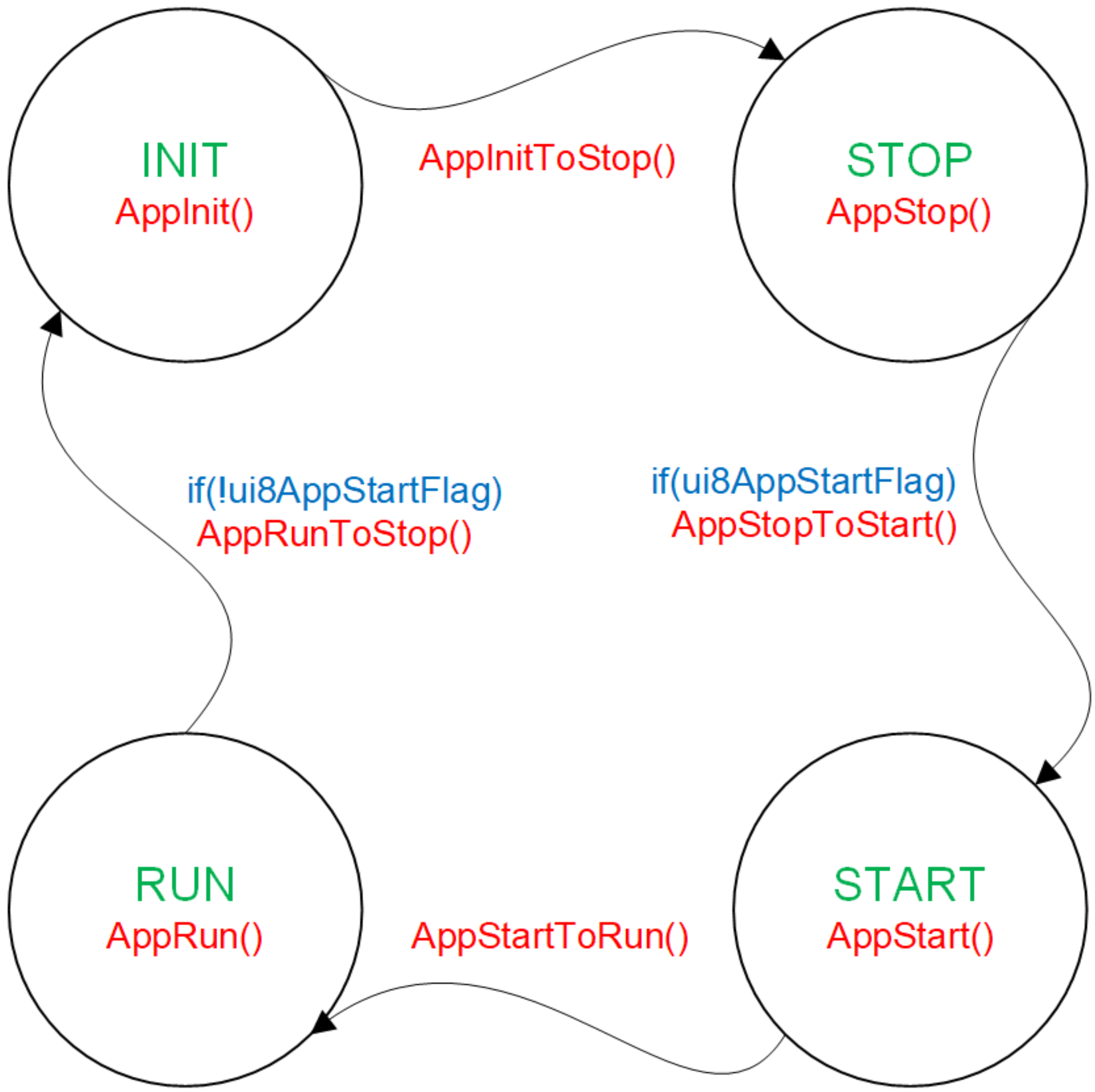

Figure 25. Application state machine

- **AppInit** is the state function of INIT state which is the initial state of application state machine. It is used to:
  1. Initialize variables related to BLDC application
  2. Call the `MC_INIT` function to initialize PME and SCTimer and startup PWM control state machine
  3. Call the `AppInitToStop` function to switch the state machine state to STOP.
- **AppStop** is the state function of STOP state. This function determines whether to call the `AppStopToStart` function to switch the state machine state to START according to the startup flag, `ui8AppStartFlag`. The `ui8AppStartFlag` flag can be controlled by FreeMASTER GUI running on the PC. If it is **1**, it indicates that the motor is started. Otherwise, the motor is stopped.
- **AppStart** is the state function of START state. SCTimer starts up by clearing halting bit in SCTimer control register. Meanwhile, SCTimer/PWM-based PWM control state machine is also started to run. After SCTimer starts up, `AppStartToRun` is called to switch the state machine state to RUN.
- **AppRun** is the state function of RUN state. This function adjusts the output of the PI controller according to the error between the actual speed and the target speed, and then PI controller output is converted to PWM duty cycle.

1. As mentioned in SCTimer/PWM-based PWM waveform with dead time, PWM duty cycle is determined by EV4 and EV5 events.
2. According to the `SCT_INIT` function which is used to intialize SCTimer, EV4 event corresponds to match reload register 3 which is defined as macro MR3 and EV5 event corresponds to match reload register 4 which is defined as macro MR4. PWM duty cycle updating is done when MR3 and MR4 have been configured.
3. If motor startup flag named as `ui8AppStartFlag` is 0, `AppRunToStop` is executed to switch the state machine state to INIT.

7.5 Pattern match engine and SCTimer configurations

`PME_Init` and `SCT_Init` are used to configure PME and SCTimer. The code project accompanying this document has detailed comments to explain the technical details about the two functions.

7.6 Get motor position

`GET_MPOS` is called by SCT0_IRQHandler which is the interrupt service routine of SCTimer to get the position of motor to determine which sector the motor is located in. The code for this function is as shown in Figure 26.

```
unsigned char GET_MPOS(void)
{
  //BLDC rotor position variable
  unsigned char s_MotorPosition;
  //Count variable
  unsigned int s_i;
  //Hall sensor phase A status
  uint32_t pha;
  //Hall sensor phase B status
  uint32_t phb;
  //Hall sensor phase C status
  uint32_t phc;

  //a short delay
  for(s_i=0; s_i<10; s_i++)
  {
  };
  //clear BLDC rotor position variable
  s_MotorPosition = 0;
  //Get Hall sensor A phase status
  pha = ((GPIO->PIN[0] & (1 << M_PHA)) >> M_PHA) << 2;
  //Get Hall sensor B phase status
  phb = ((GPIO->PIN[0] & (1 << M_PHB)) >> M_PHB) << 1;
  //Get Hall sensor C phase status
  phc = ((GPIO->PIN[0] & (1 << M_PHC)) >> M_PHC) << 0;
  //Get BLDC rotor position
  s_MotorPosition = pha | phb | phc;

  return s_MotorPosition;
}
```

Figure 26. Get motor position

7.7 Update SCTimer output

As described in BLDC power output stage, this document adopts unipolar mode to drive the motor. Array defined as `hall_to_pwm`, as shown in Figure 27, is used to converts the Hall sensor states into PWM output combinations. Macro defined as `PWM_TB_H_Lx` specifies the channels on which complementary PWM pair is applied, the channel on which high level is applied and the channels on which low level is applied.

```
#define PWM_TB_H_Lx(PT, PB, H, L2, L1, L0) (
                                            PT<<28 | PB<<24 |
                                            H<<12 |
                                            L2<<8 | L1<<4 | L0<<0
                                           )

const unsigned long hall_to_pwm[2][8] =
{
  {
    0,                          //000
    PWM_TB_H_Lx(5,6,4,3,2,1),  //001
    PWM_TB_H_Lx(3,4,2,6,5,1),  //010
    PWM_TB_H_Lx(5,6,2,4,3,1),  //011
    PWM_TB_H_Lx(1,2,6,5,4,3),  //100
    PWM_TB_H_Lx(1,2,4,6,5,3),  //101
    PWM_TB_H_Lx(3,4,6,5,2,1),  //110
    0                           //111
  },
  { 0,                          //000
    PWM_TB_H_Lx(3,4,6,5,2,1),  //001
    PWM_TB_H_Lx(1,2,4,6,5,3),  //010
    PWM_TB_H_Lx(1,2,6,5,4,3),  //011
    PWM_TB_H_Lx(5,6,2,4,3,1),  //100
    PWM_TB_H_Lx(3,4,2,6,5,1),  //101
    PWM_TB_H_Lx(5,6,4,3,2,1),  //110
    0                           //111
  }
};
```

Figure 27. Hall-to-PWM array

Function named as `update_sct_out` extracts the channels on which complementary PWM pair is applied, the channel on which high level is applied and the channels on which low level is applied and updates SCTimer output channels. Key code segments are as shown in Figure 28 and Figure 29.

```
pwm_out_t  = out_pwm1_6_2_sct_out[(pwm_control>>28) & 0x0F];
pwm_out_b  = out_pwm1_6_2_sct_out[(pwm_control>>24) & 0x0F];
pwm_out_h  = out_pwm1_6_2_sct_out[(pwm_control>>12) & 0x0F];
pwm_out_l2 = out_pwm1_6_2_sct_out[(pwm_control>>8 ) & 0x0F];
pwm_out_l1 = out_pwm1_6_2_sct_out[(pwm_control>>4 ) & 0x0F];
pwm_out_l0 = out_pwm1_6_2_sct_out[(pwm_control>>0 ) & 0x0F];
```

Figure 28. Parse the SCTimer output channel

```
SCT0->OUT[pwm_out_t].SET = 1<<app_pwm_ev_edge0;
SCT0->OUT[pwm_out_t].CLR = 1<<app_pwm_ev_edge1 | 1<<app_pwm_ev_update0;

SCT0->OUT[pwm_out_b].SET = 1<<app_pwm_ev_edge2;
SCT0->OUT[pwm_out_b].CLR = 1<<app_pwm_ev_edge3 | 1<<app_pwm_ev_update0;

SCT0->OUT[pwm_out_h].SET = 1<<app_pwm_ev_update1;
SCT0->OUT[pwm_out_h].CLR = 0;

SCT0->OUT[pwm_out_l2].SET = 0;
SCT0->OUT[pwm_out_l2].CLR = 1<<app_pwm_ev_update0;

SCT0->OUT[pwm_out_l1].SET = 0;
SCT0->OUT[pwm_out_l1].CLR = 1<<app_pwm_ev_update0;

SCT0->OUT[pwm_out_l0].SET = 0;
SCT0->OUT[pwm_out_l0].CLR = 1<<app_pwm_ev_update0;
```

Figure 29. Update SCTimer output

7.8 SCTimer interrupt service routine

`SCT0_IRQHandler` is the SCTimer interrupt service routine and it is mainly used to perform the commutation based on the motor position from Hall sensor. Figure 30 shows the program flow.

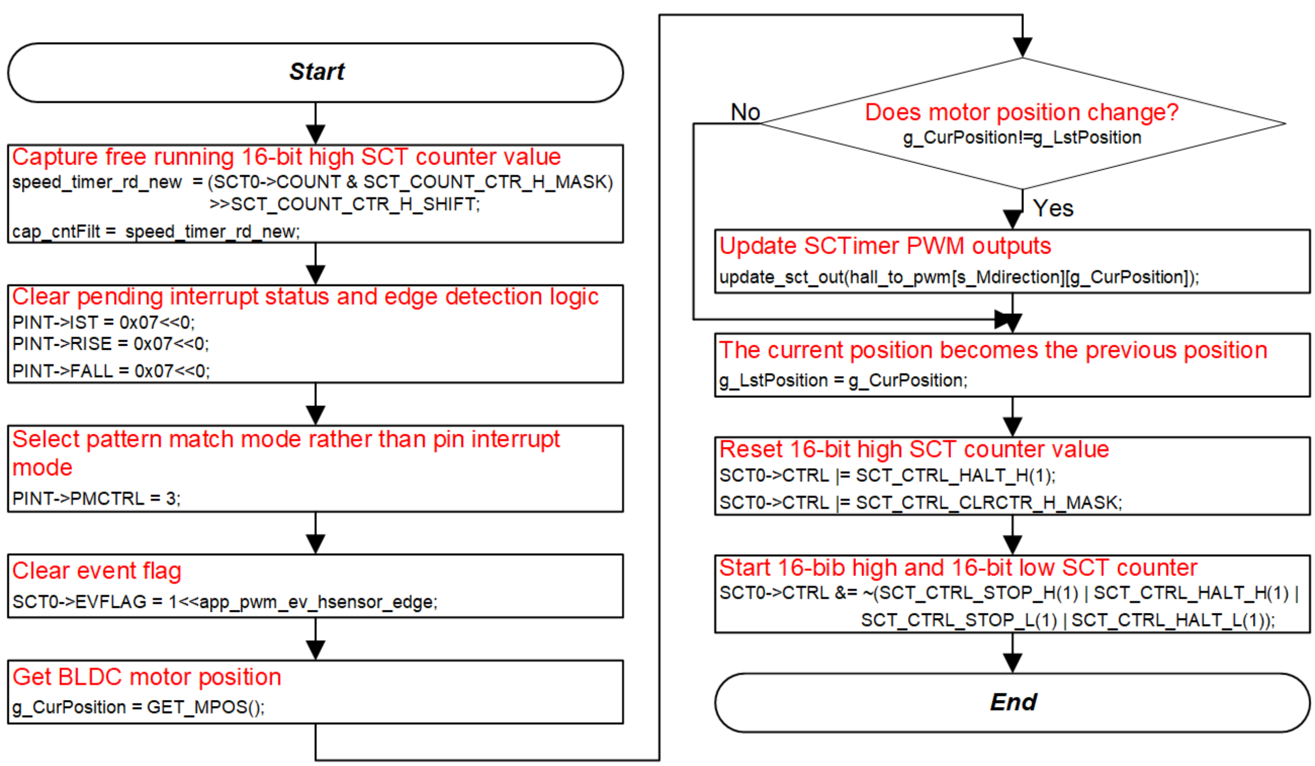

Figure 30. SCT0_IRQHandler Flow

8 FreeMASTER run-time debugging tool

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool and supports *non-intrusive monitoring* of variables on a running system and can *display multiple variables* on oscilloscope-like displays as standard widgets (gauges, sliders, and more) or as data in text form, offering simple-to-use data recorders.

FreeMASTER is used as a real-time monitoring tool for LPC51U68 BLDC application. Figure 31 shows the user interface for this application and Table 7 lists common-used variables. The project file is named as *SCT_BLDC_LPC51U68.pmpx* which is located in *boards\lpcxpresso51u68\driver_examples\sctimer\common*.

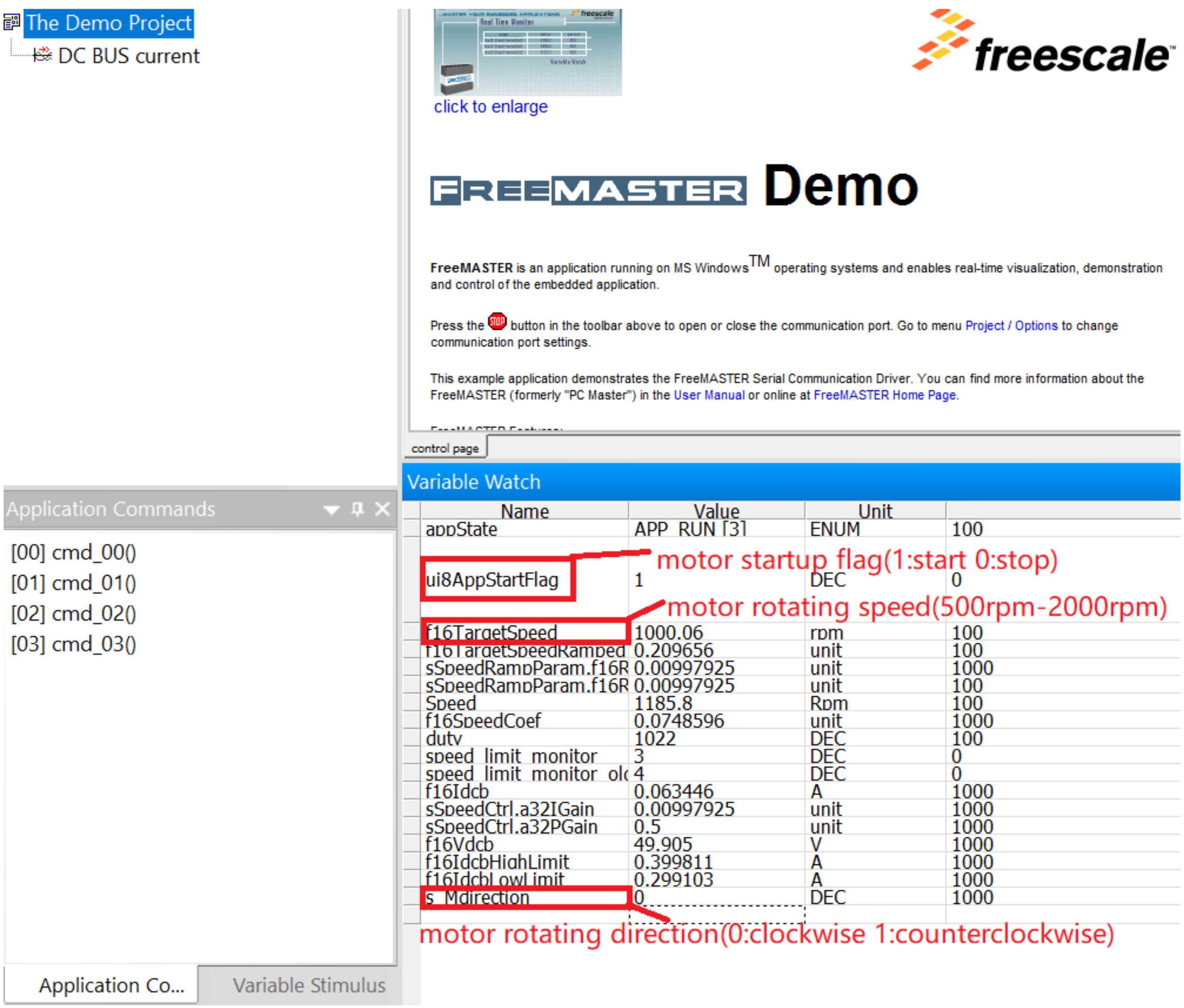

Figure 31. LPC51U68 BLDC application FreeMASTER UI

Table 7. Commonly-used FreeMASTER variables

Variable	Value	Unit	Attributes	Function
ui8AppStartFlag	0	—	read-write	Stop

Table continues on the next page...

Table 7. Commonly-used FreeMASTER variables (continued)

Variable	Value	Unit	Attributes	Function
	1	—		Start
f16TargetSpeed	500 - 2500	rpm	read-write	Target speed
Speed	Depending on actual measurement	rpm	read-only	Actual speed
s_Mdirection	0	—	read-write	Clockwise rotation
	1	—		Counterclockwise rotation
f16TargetSpeedRamped	0.00213623* f16TargetSpeed	—	read-only	Target ramp speed
f16Idcb	Depending on actual measurement	A	read-only	Direct current bus current
f16Vdcb	Depending on actual measurement	V	read-only	Direct current bus voltage
f16IdcbHighLimit	0.399811	A	read-only	Direct current bus current upper limit
f16IdcbHighLimit	0.299103	A	read-only	Direct current bus current lower limit

9 How to use LPC51U68 BLDC application demo

Please follow the steps below in order to run this application:

1. Prepare LPCXpresso51U68 evaluation board and FRDM-MC-LVBLDC motor driver board.
2. Compile the code project accompanying with this document to generate the executable program and download it to the LPCXpresso51U68 evaluation board. Then exit debug mode.
3. According to Table 4, Table 5, and Figure 18, use jumpers to connect the LPCXpresso51U68 evaluation board and FRDM-MC-LVBLDC motor driver board, as shown in Figure 17.
4. Connect the motor wires such as three phase outputs, HALL signals, 5 V and GND to the FRDM-MC-LVBLDC board.
5. Open FreeMASTER project file for this application. The project file is *SCT_BLDC_LPC51U68.pmpx* which is located in *boards\lpcxpresso51u68\driver_examples\sctimer\common*.
6. Click **Project** -> **Option**, as shown in Figure 32. Figure 33 shows the configure communication interface.

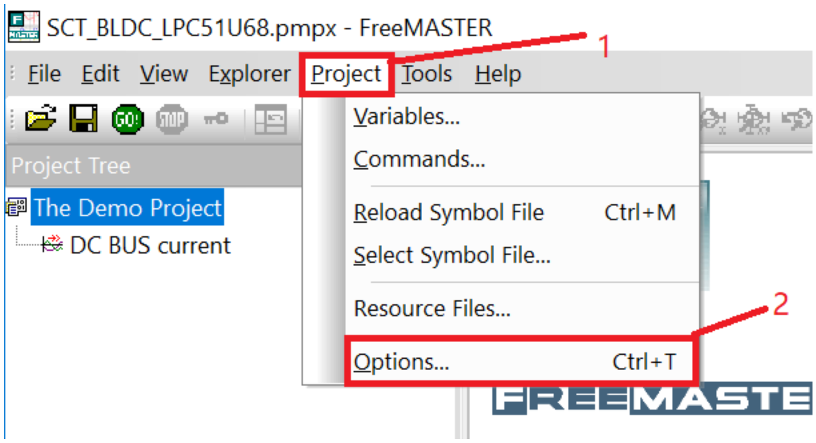

Figure 32. Enter project configuration interface

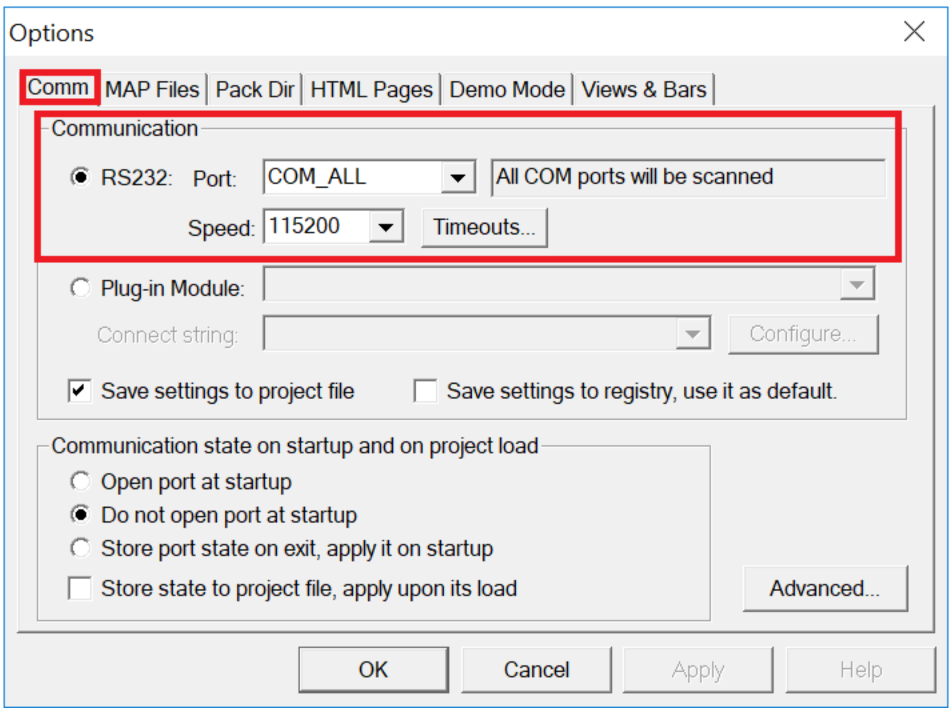

Figure 33. Configure communication interface

7. Click **MAP Files** to configure symbol file path, as shown in Figure 34. The symbol file path is *boards\lpcxpresso51u68\driver_examples\sctimer\simple_pwm\iar\debug\sctimer_simple_pwm.out*.

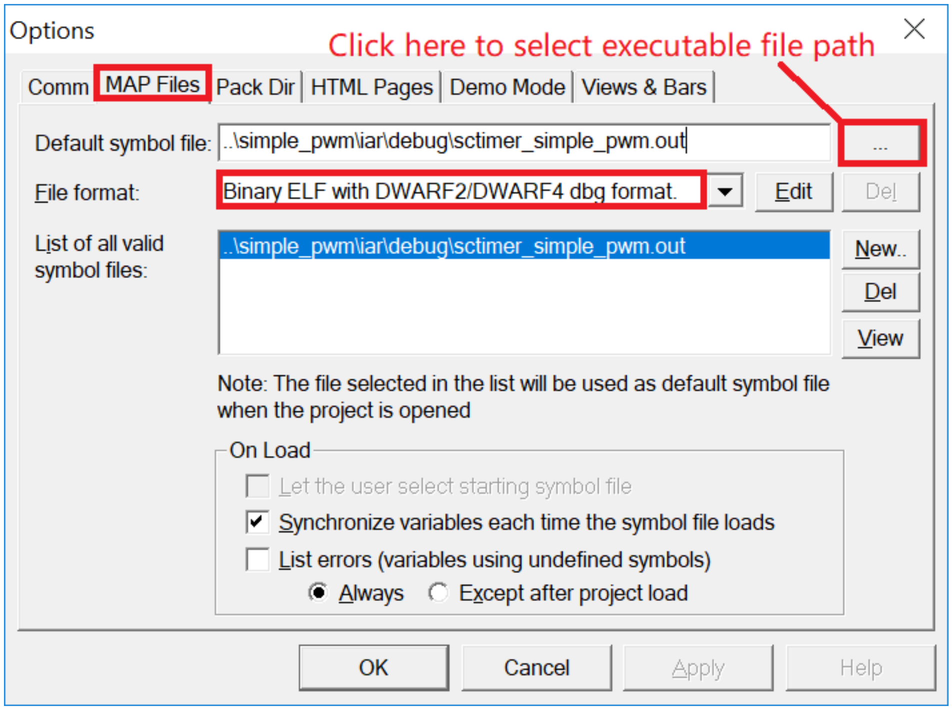

Figure 34. Configure symbol file path

8. Connect LPCXpresso51U68 evaluation board to PC using USB cable.
9. Supply the 12 V direct current voltage to the FRDM-MC-LVBLDC board and click the **GO** button on FreeMASTER.

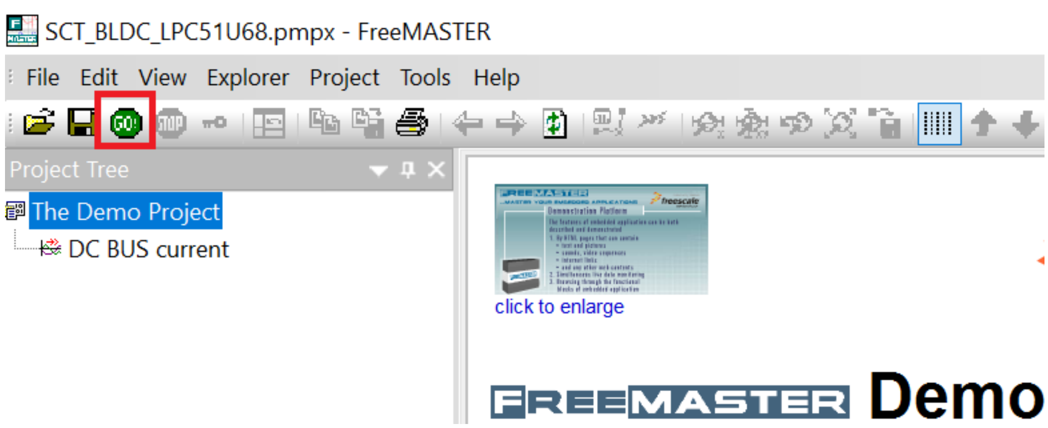

Figure 35. Start Up FreeMASTER Communication

10. LPC51U68 BLDC application user interface appears as shown in Figure 31.
    - Set **ui8AppStartFlag** to **1** or **0** to start or stop motor rotation.
    - Set **f16TargetSpeed** to specify desired speed.
    - Set **s_Mdirection** to select rotating direction, which is clockwise if **s_Mdirection** is **0** and counterclockwise if **s_Mdirection** is **1**.

10 LPC51U68 BLDC application run in IDE debug mode

LPC51U68 BLDC application usually runs in normal mode, that is, non-debug mode. However, if it runs in debug mode, pause debugging will affect the operation of this application.

Refer to How to use LPC51U68 BLDC application demo and establish the operating environment of the application including software and hardware. This application is developed based on IAR Embedded Workbench. Therefore, download application code to LPC51U68 and then the application enters debug mode. Make the application run continuously without breakpoints in debug mode, and then click the pause debugging button in IAR to pause debugging. Meanwhile, use an oscilloscope to observe six PWM outputs and a possible result is as shown in Figure 36.

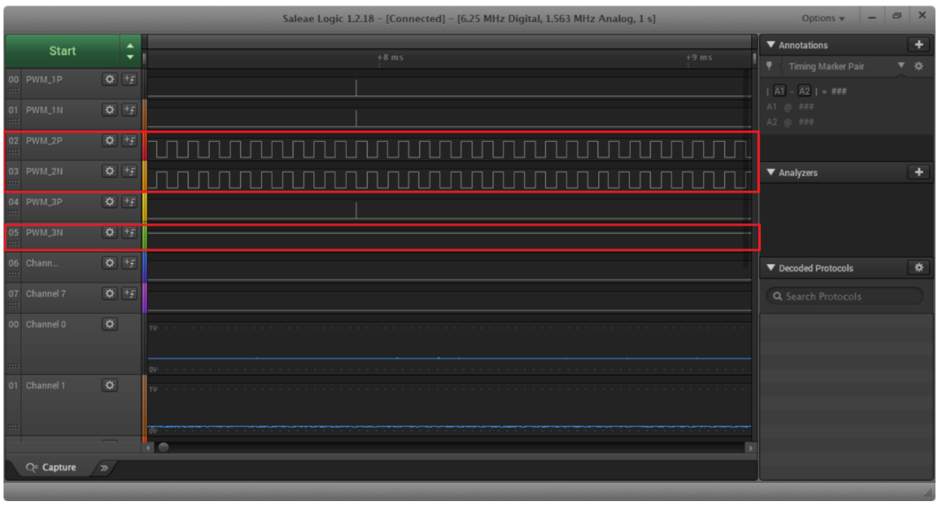

Figure 36. SCTimer PWM outputs when pausing debugging

As shown in Figure 3 and Figure 36, `PWM_2P` and `PWM_2N` which consist of one complementary PWM pair are applied on Q2T and Q2B MOS transistors. Meanwhile, PWM_3N which is a constant high level is applied on Q3B. Refer to Table 2 and this means that the current flows into the motor B-phase winding and flows out of the C-phase winding. If the high PWM duty cycle causes the phase current to be more than 2A, the motor may heat up in a short time.

How to solve this problem? As shown in Figure 37 which is from Figure 27, SCT connections in UM11071 - LPC51U68 User manual with version 1.2, signal defined as **debug halted** can be used as SCTimer input. This signal is generated when the application debugging is paused. SCTimer can use this signal as event source and is configured to clear six PWM outputs and stop SCTimer when this event occurs. Figure 38 shows the key code segment.

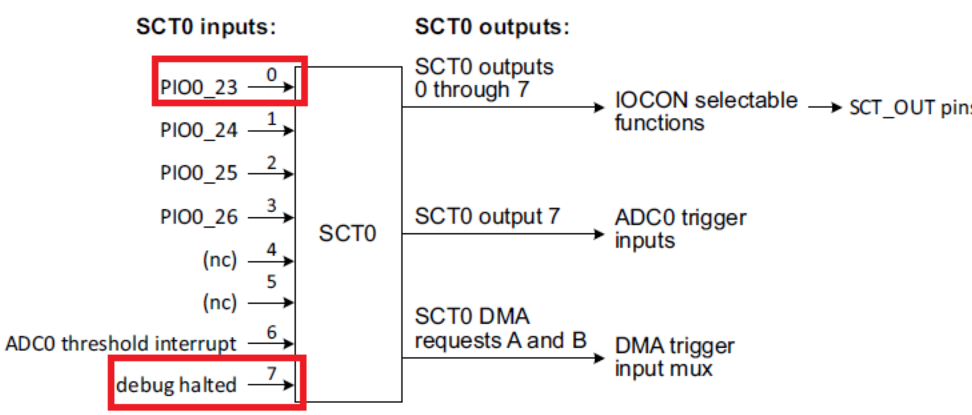

Figure 37. SCTimer inputs and outputs

```
//debug halt event is 8th event
//debug halt may be generated when application is in idle or active or transient state
SCT0->EV[8].STATE = (
                     (1<<app_pwm_st_idle) |
                     (1<<app_pwm_st_active) |
                     (1<<app_pwm_st_transient)
                    );
SCT0->EV[8].CTRL = SCT_EV_CTRL_DIRECTION(0x0) | //This event is triggered
                                                //regardless of the count direction
                   SCT_EV_CTRL_IOCOND(0x3) |    //high level is trigger condition
                   SCT_EV_CTRL_OUTSEL(0x0) |    //trigger source is input
                   SCT_EV_CTRL_IOSEL(0x7) |     //debug halt is connected to SCTimer input 7
                   SCT_EV_CTRL_COMBMODE(0x2) |  //only IO event can trigger this event
                   EVCTR_STATE_NO_CHANGE;       //state no change
SCT0->OUT[app_out_pwm1].CLR = 0x100;//only event 8 can clear PWM out1
SCT0->OUT[app_out_pwm1].SET = 0;
SCT0->OUT[app_out_pwm2].CLR = 0x100;//only event 8 can clear PWM out2
SCT0->OUT[app_out_pwm2].SET = 0;
SCT0->OUT[app_out_pwm3].CLR = 0x100;//only event 8 can clear PWM out3
SCT0->OUT[app_out_pwm3].SET = 0;
SCT0->OUT[app_out_pwm4].CLR = 0x100;//only event 8 can clear PWM out4
SCT0->OUT[app_out_pwm4].SET = 0;
SCT0->OUT[app_out_pwm5].CLR = 0x100;//only event 8 can clear PWM out5
SCT0->OUT[app_out_pwm5].SET = 0;
SCT0->OUT[app_out_pwm6].CLR = 0x100;//only event 8 can clear PWM out6
SCT0->OUT[app_out_pwm6].SET = 0;
SCT0->STOP |= ((1<<8) | (1<<24));//stop SCTimer when event 8 occurs
```

Figure 38. SCTimer Debug Halted Event Setting

Please follow the steps below to run this application in the debug mode:

1. Refer to How to use LPC51U68 BLDC application demo and follow the steps described in this chapter to establish the hardware and software environment required for this application to run.
2. Use jumpers to connect six test channels and ground of logic analyzer to the six PWM inputs and ground on FRDM-MC-LVBLDC.
3. Compile the code project accompanying with this document to generate the executable program and download it to the LPCXpresso51U68 evaluation board. The application enters debug mode after downloading.
4. Click the run button or pause button shown in Figure 39 to run or pause this application.

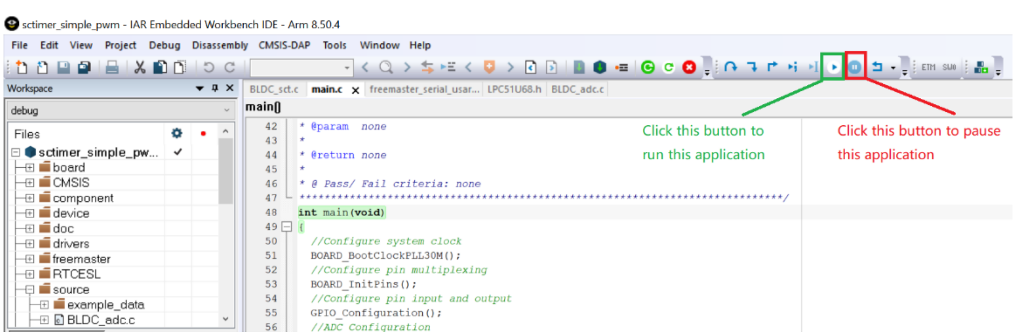

Figure 39. Application run or pause

5. Measure six PWM inputs on FRDM-MC-LVBLDC using logic analyzer and one possible result is shown in Figure 40. We can see that only PWM_2N and PWM_3N are high level, therefore, no current flows through the motor winding coil.

Figure 40. One possible PWM waveform when debug paused

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: 11/2020
Document identifier: AN13040