

by: NXP Semiconductors

## 1 Introduction

Based on the Arm<sup>®</sup> Cortex<sup>®</sup>-M0+ core, LPC84x is a low-cost, 32-bit MCU family operating at frequencies of up to 30 MHz. The LPC84x MCU family supports up to 64 KB of flash memory and 16 KB of SRAM. This family features exceptional power efficiency in the low-current mode using the FRO as the clock source. The peripheral complement of the LPC84x MCU family includes a CRC engine, four I<sup>2</sup>C-bus interfaces, up to five UARTs, up to two SPI interfaces, one capacitive touch interface, one multi-rate timer, one self-wake-up timer, one SCTimer/PWM, one general-purpose 32-bit counter/timer, one DMA, one 12-bit ADC, two 10-bit DACs, one analog comparator, function-configurable I/O ports through a switch matrix, one input pattern match engine, and up to 54 general-purpose I/O pins.

This application note demonstrates how to implement the six-step commutation or brushless DC motor control on the LPC84x family of MCUs. The LPC84x devices contain an SCTimer/PWM unit. The SCTimer/PWM unit is highly integrated among the analog peripherals. The switch matrix supports flexible assignment of the SCTimer/PWM functions to the pins.

This application note is also a reference and a starting point for motor-control system developers using the LPC84x family of MCUs.

## 2 Brushless DC (BLDC) motor principle

BLDC motors consist of a permanent-magnet rotor with a three-phase stator winding. As the name implies, BLDC motors do not use brushes for commutation and they are commutated electronically. Typically, three Hall sensors are used to detect the rotor position and the commutation is based on these sensor inputs.

In a BLDC motor, the electromagnets do not move. The permanent magnets rotate and the three-phase stator windings remain static. This solves the problem with transferring current to a moving rotor. To do this, the brush-commutator assembly is replaced by an intelligent electronic controller. The controller performs the same power distribution as in a brushed DC motor, but it uses a solid-state circuit rather than a commutator/brush system. See [Figure 1](#).

The torque generated by the motor depends on the applied current, which is a result of the applied voltage. The achieved speed is linked to the acceleration of the motor up to the condition at which the delivered torque equals the load torque. The effective current (and thus the torque) drops when increasing speed due to the BEMF generated by the motor. This effect is reflected in a voltage-dependent speed.

### Contents

<b>1 Introduction</b> .....	<b>1</b>
<b>2 Brushless DC (BLDC) motor principle</b> .....	<b>1</b>
<b>3 Controlling a BLDC motor using SCTimer/PWM</b> .....	<b>5</b>
<b>4 Hardware</b> .....	<b>10</b>
<b>5 Running example application</b> .....	<b>11</b>
<b>6 Conclusion</b> .....	<b>13</b>
<b>7 References</b> .....	<b>13</b>



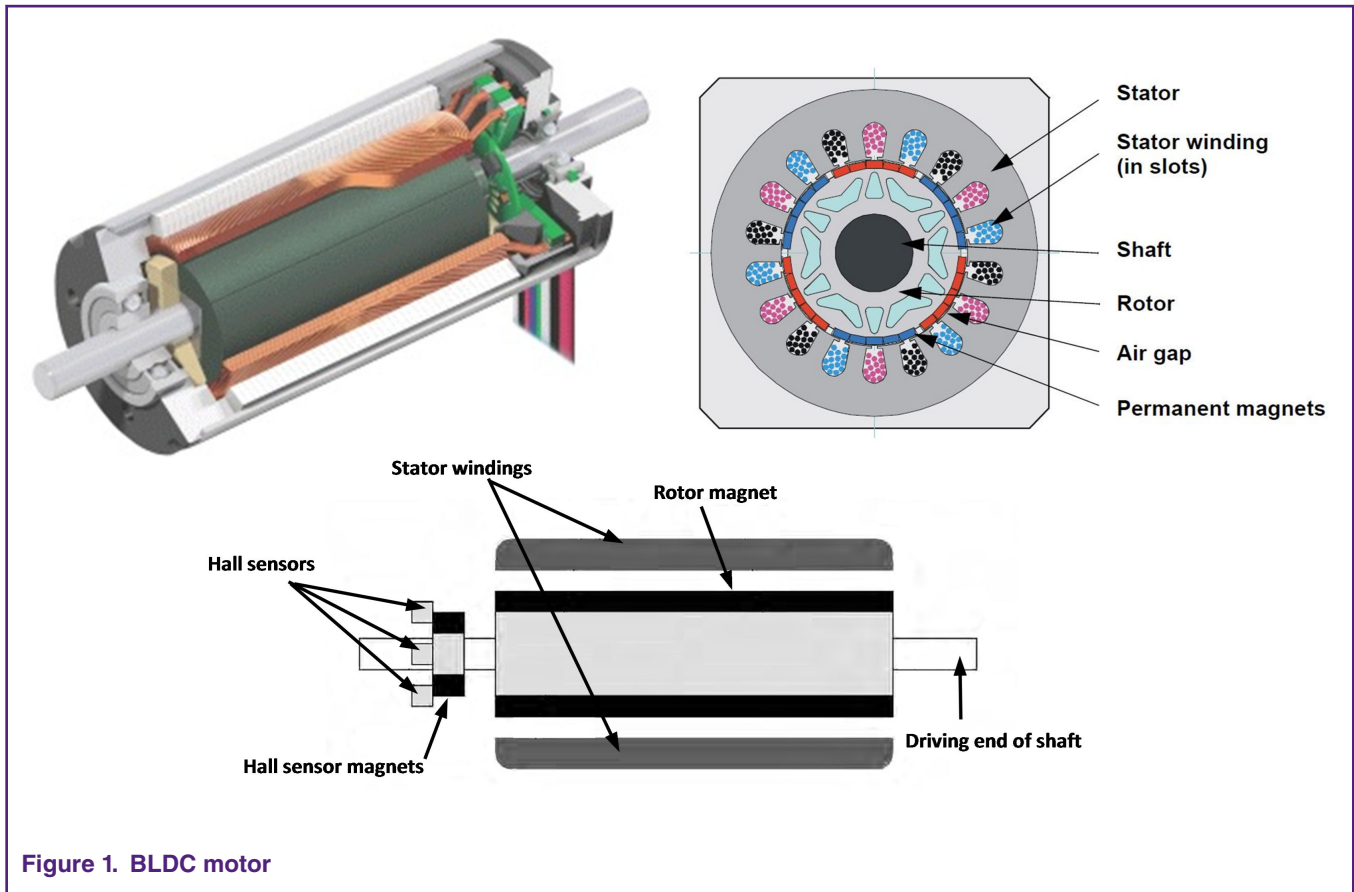


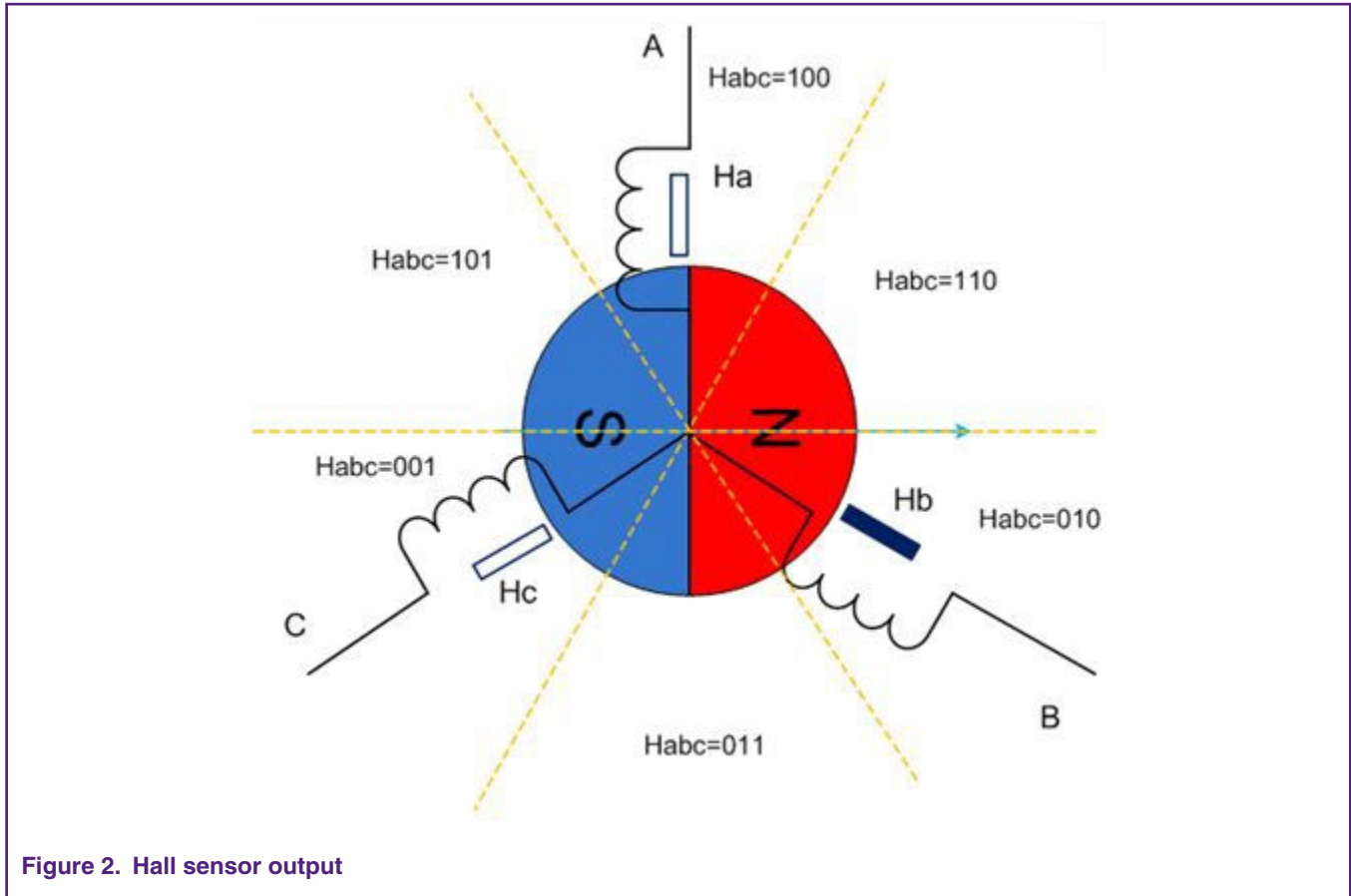
Figure 1. BLDC motor

## 2.1 Six-step commutation

The Hall effect sensor is a sensing switch that outputs a logic level based on the magnetic field detected. The Hall effect sensors (Ha, Hb, and Hc) are usually included in the motor.

For example, when the Ha sensor is under the N pole of the permanent magnet, it outputs signal 1, otherwise 0. See [Figure 2](#).

Combining the outputs of all three sensors gives 8 states from 000 to 111. Depending on how the Hall sensors are installed, signals 000 and 111 do not exist in most cases. The other six states can divide the 360 degrees into six areas. The exact point where the status changes from one to another is the position where the commutator changes the direction of the stator's magnetic field.



**Figure 2. Hall sensor output**

Figure 3 shows an example of commutation where the Hall sensor status is 010. For the rotor to spin clockwise, the clockwise rotating magnetic field must be generated in its nearest area (where the Hall sensor status is 011). This direction of the magnetic field can be generated by turning on coils A and C, which means that the current flows into A and runs out of C. When the rotor moves to area 011, the Hall sensor status changes to 011. At the same time, the commutator changes from AC to BC for the rotor to follow the rotating magnetic field.

Thus, the power sequence is AC -> BC -> BA -> CA -> CB -> AB -> AC.

Here is the summary of the commutation process.

- In one complete rotation of 360 electrical degrees, the excitation of the stator windings is changed six times and each change is called a commutation.
- The angle between the S-N pole (rotor) and the magnetic field (stator windings) is 60-120°. The commutation happens at 60°.
- The commutation position is when the status of a Hall sensor changes.
- At every moment, only two coils are powered.

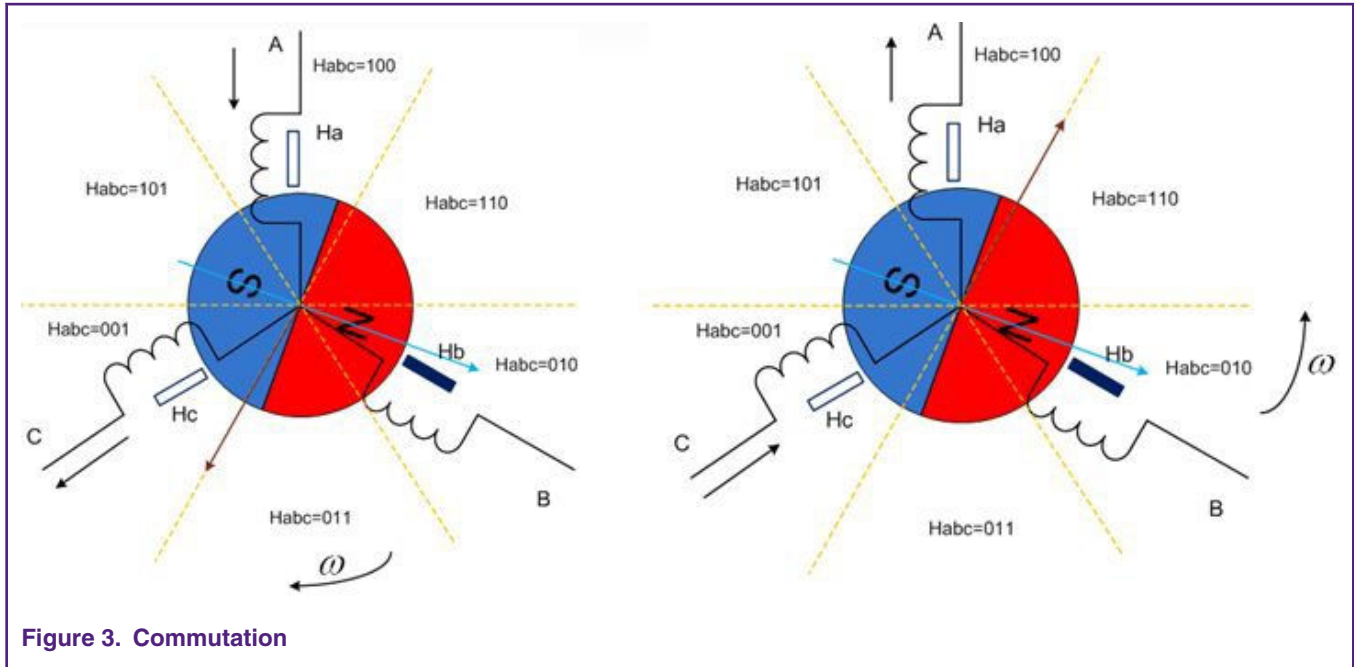


Figure 3. Commutation

## 2.2 Commutation table

As discussed in [Six-step commutation](#), the six commutation positions are fixed in 360 electrical degrees. A commutation table can be built to describe the relationship between the sensor status and the stator winding excitation. With this commutation table, the MCU can easily control the commutation.

Table 1. Commutation table

Step	Hall sensors			Phase		
	a	b	c	A	B	C
1	0	1	1	NC	+	-
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	-

Here is the terminology used in the commutation table:

- The "Hall sensors" header column provides the Hall sensor status captured from the motor.
- The "Phase" header column determines how the stator windings are excited.
- The "+" symbol means that the current flows into that terminal.
- The "-" symbol means that the current flows out of that terminal.
- "NC" means that no voltage is applied on that terminal.

### 3 Controlling a BLDC motor using SCTimer/PWM

The MCU that drives the motor must have a power output. Due to this standard, the MCU is usually used only to generate the driving signal and detect the Hall signal. A power stage is used to power the motor. In the described case, the FRDM MC LVBLDC board is used.

#### 3.1 Basic motor-control topology

Figure 4 shows a basic three-phase motor-control topology where a three-phase inverter is used to control the voltage applied on the motor phases. The PWM control signal comes from the MCU. In this motor-drive board, voltage level "1" turns the transistor on and "0" turns the transistor off.

There are two types of switching in the theory of BLDC motor driving: bipolar and unipolar. Each of these two approaches needs a different waveform.

This document describes only the unipolar switching. In the unipolar approach, a complementary PWM is applied on one phase and the circuit is closed through one bottom transistor in the other phase (and this transistor is permanently on).

For example, if PWM1 and PWM2 (Phase A) are complementary and have a duty cycle (with dead time), PWM4 is permanently on and the others are off, which means that phases A and B are excited.

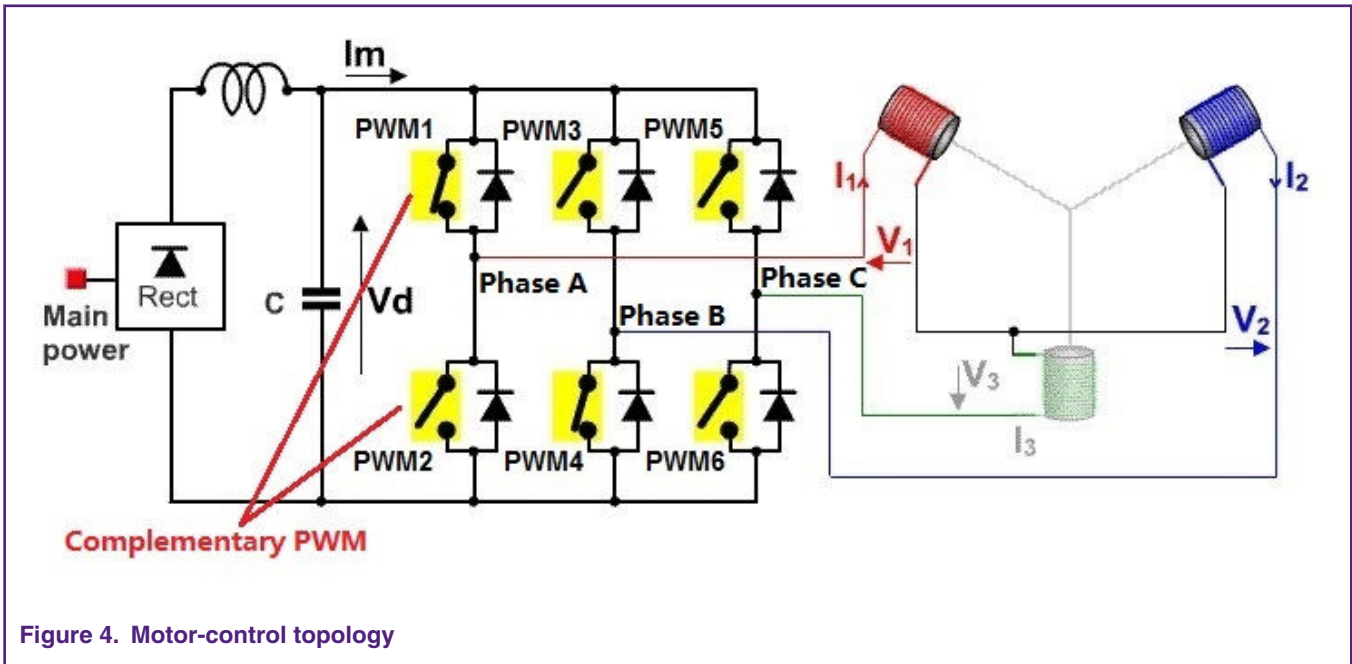


Figure 4. Motor-control topology

The waveforms for the PWM signals and the Hall sensor values driving a BLDC engine in 360 electrical degrees are shown in Figure 5. The PWM duty cycle changes the speed of the rotor. In Figure 5, the Hall sensor status is 110->100->101->001->011->010 and the excitation of the stator windings is CB->CA->BA->BC->AC->AB.

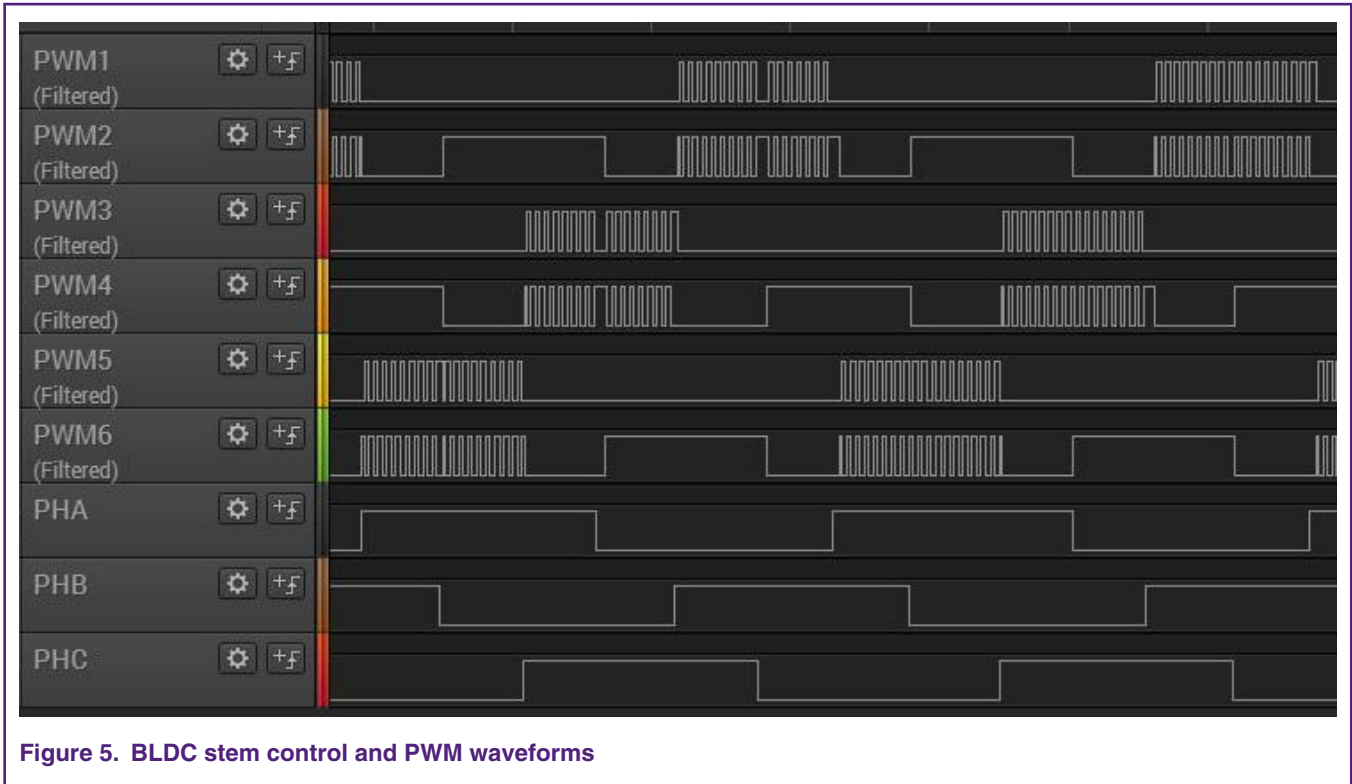


Figure 5. BLDC stem control and PWM waveforms

The details of the technique are shown in Figure 6. It shows the dead time insertion in the complementary top and bottom signals. This requires the insertion of dead time, because the switching transient causes a DC-bus short circuit with fatal power stage damage.

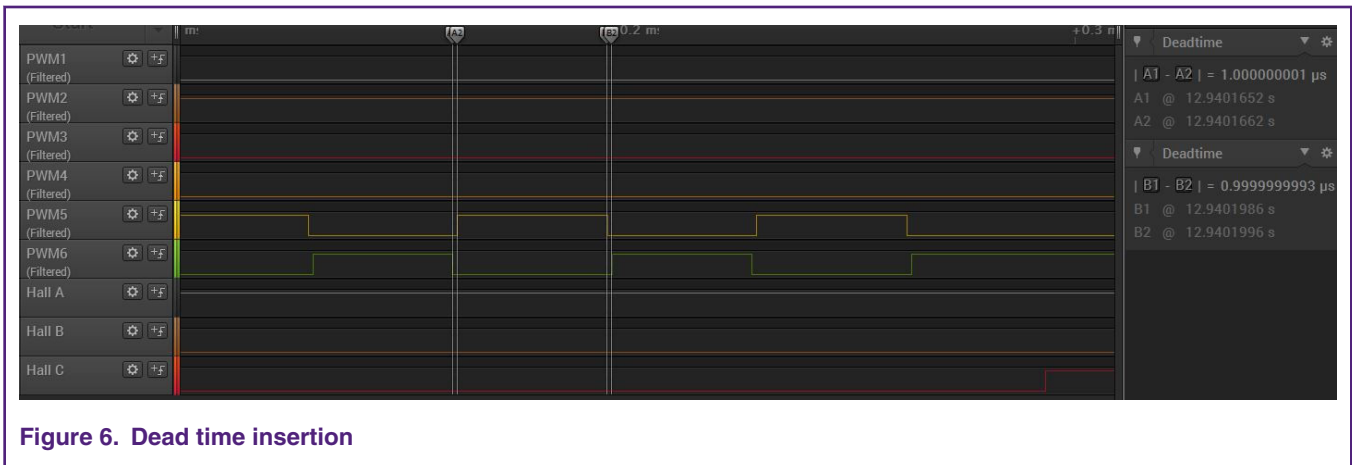


Figure 6. Dead time insertion

### 3.2 Motor control on SCTimer/PWM

It is necessary to generate a PWM for motor control. Because this PWM is usually fast, it is necessary to use a peripheral PWM module. The SCTimer/PWM module is assembled on LPC devices.

#### 3.2.1 SCTimer/PWM on LPC845 overview

The LPC84X SCTimer/PWM supports:

- Five inputs: four inputs with multiple connection options through the input MUX.
- Seven outputs.

- Eight match/capture registers.
- Eight events.
- Eight states.

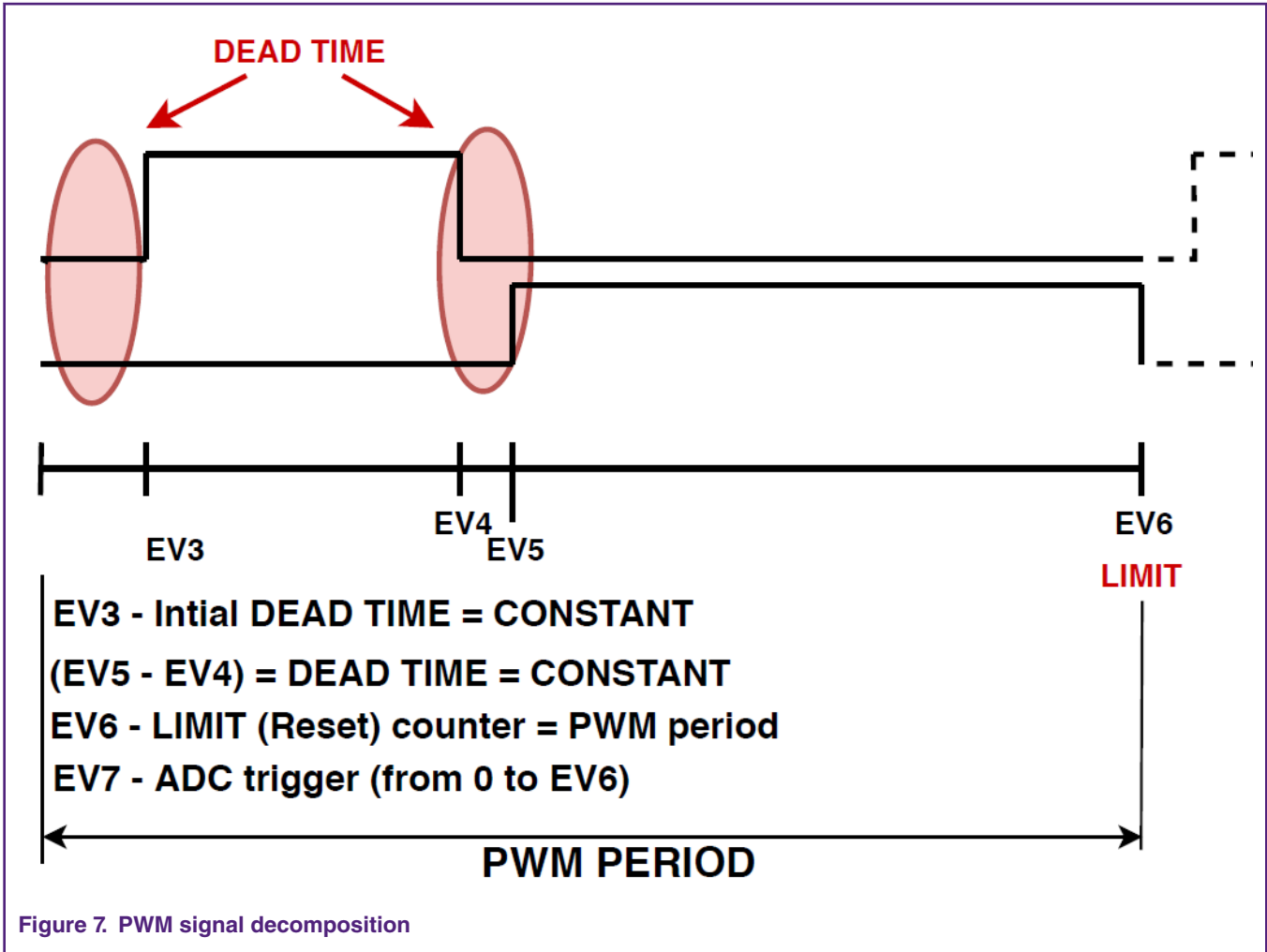
SCTimer/PWM features:

- Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
- Configurable as up counters or up-down counters.
- Configurable number of match and capture registers. Up to five match and capture registers in total.
- When there is a match and/or an input or output transition or level, create events to accomplish any or all of the following:
  - Stop, limit, or halt the timer.
  - Change the counting direction.
  - Set, clear, or toggle outputs.
  - Change the state.
  - Capture the counter value.
  - Generate an interrupt or a DMA request.
- The counter value can be loaded into the capture register triggered by a match or an input/output toggle.
- The counters can be used in conjunction with the match registers to toggle the outputs and create time-proportioned PWM signals.
- Up to six single-edge or dual-edge PWM outputs with an independent duty cycle and a common PWM frequency.

### 3.2.2 SCTimer-based BLDC motor waveform details

It is necessary to add dead time for a complementary PWM signal. The off state time in the generated signal (due to the hardware limitation of the transistor) is the reason for no shortcuts in the transient state.

In the presented software, the edge-align PWM approach is used (see [Figure 7](#)).



The frequency of the PWM is set by event 6 and this event resets the counter back to zero.

Event 3 has a constant value to achieve dead time. The difference between events 5 and 4 is also constant and it also defines the dead time.

Event 7 is used to generate the ADC trigger, which can be set anywhere to activate a signal specific to the application needs. In the example application, it is set to a 25 % duty cycle of the top transistor.

To change the duty cycle, change the position of EV4. EV5 must be set as EV4 + the dead time.

### 3.2.3 Implementing BLDC motor control using SCT

The basic description of the SCTimer is in *SCTimer/PWM Cookbook* (document [AN11538](#)).

As is described in *SCTimer/PWM Cookbook* (document [AN11538](#)), the SCT has an internal state machine. This document uses the following state machine on the SCT:



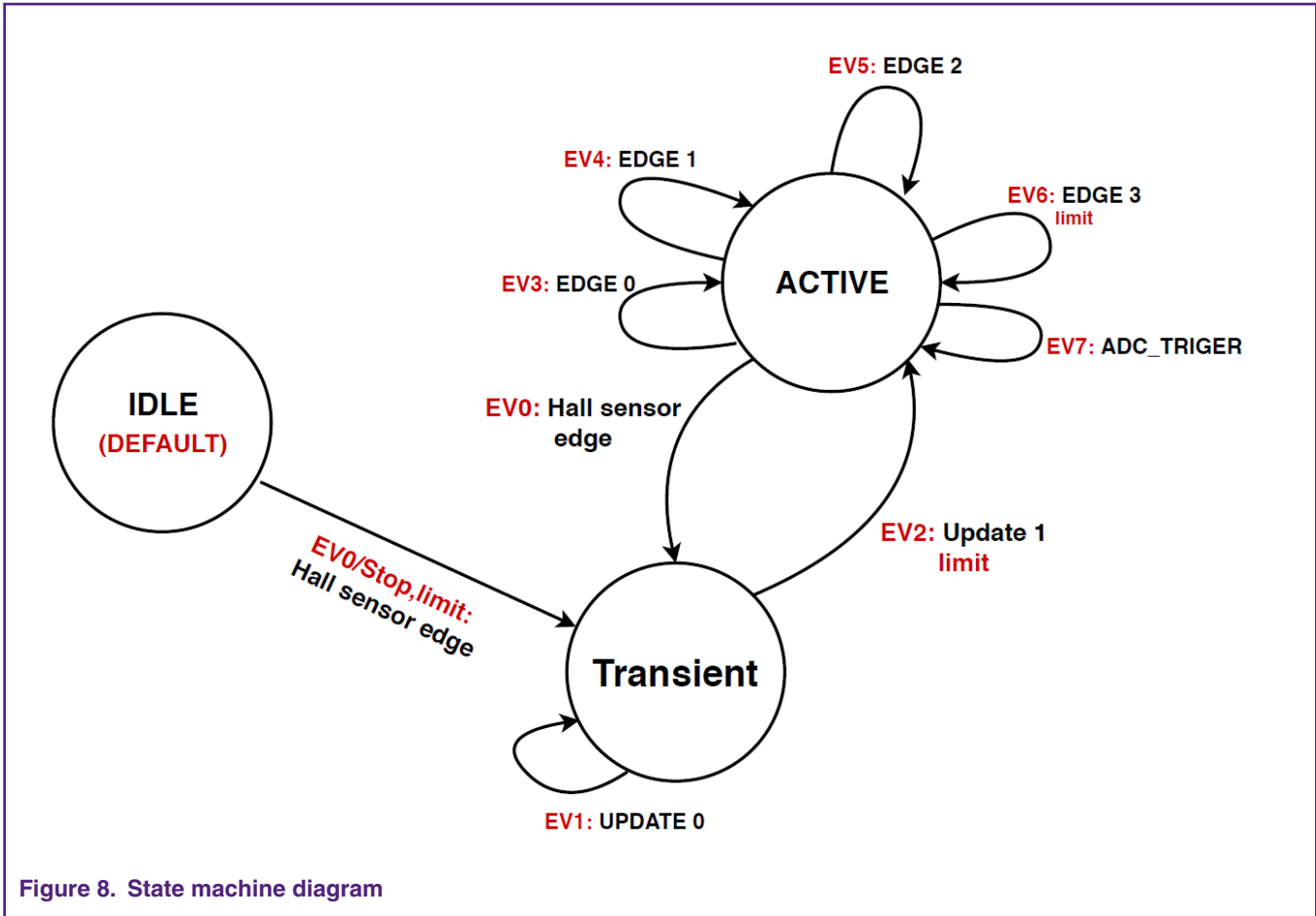


Figure 8. State machine diagram

The SCT is not a standard PWM module and it is necessary to modify it for motor-control purposes.

The above state machine is valid during the entire application run.

The complementary PWM waveform is generated in the active state. Four edges (labeled as edge 0/1/2/3) and four match events (one of them is also a limiting event and determines the PWM period) are needed. One more event active in this state is used to trigger ADC conversions.

The transient state is entered when the pattern-matching engine generates a rising edge and that happens when it detects an edge at one of the Hall sensors. This is a stop and limiting event and it generates the SCT interrupt.

The transitional state is used to prepare six SCT outputs that drive the motor for the next 1/6 of a revolution. In the transitional state, two events are defined. The first event is used to drive an output that needs to go low to become low. The second event is used to set one output to 1 (if different from the previous 1/6 of a revolution) and prepare the PWM outputs. This second event is also a limiting event and it is used to update the PWM duty cycle (if needed). See the MC\_INIT() function in file *sct\_mc.c*.

When the transitional state is entered, the ISR is used to update the SCT OUT SET/CLR registers (see the SCT0\_IRQHandler function in file *sct\_mc.c*). After this update, the SCT is set to ACTIVE again (keep in mind that the event that brought the SCT to the transitional state was a STOP event). When the SCT starts running again, it gets to the first event and updates the physical OUT levels (according to the Hall sensor readings and direction selected) and continues towards the second event. When the second event is reached, all outputs are prepared for the PWM signal to continue being generated at the selected pair of outputs with the updated duty cycle. When the second event is generated, the SCT transitions to the active state and the PWM runs again.

### 3.2.4 Speed control

The actual speed is calculated within the SCTimer module. The required speed can be set in FreeMASTER. The PI controller keeps the required speed during various load conditions by tweaking the duty cycle according to the PI controller setup. If the

load causes the DC-bus current to be at or over the limit, the algorithm switches to the current limiting mode until the higher load disappears.

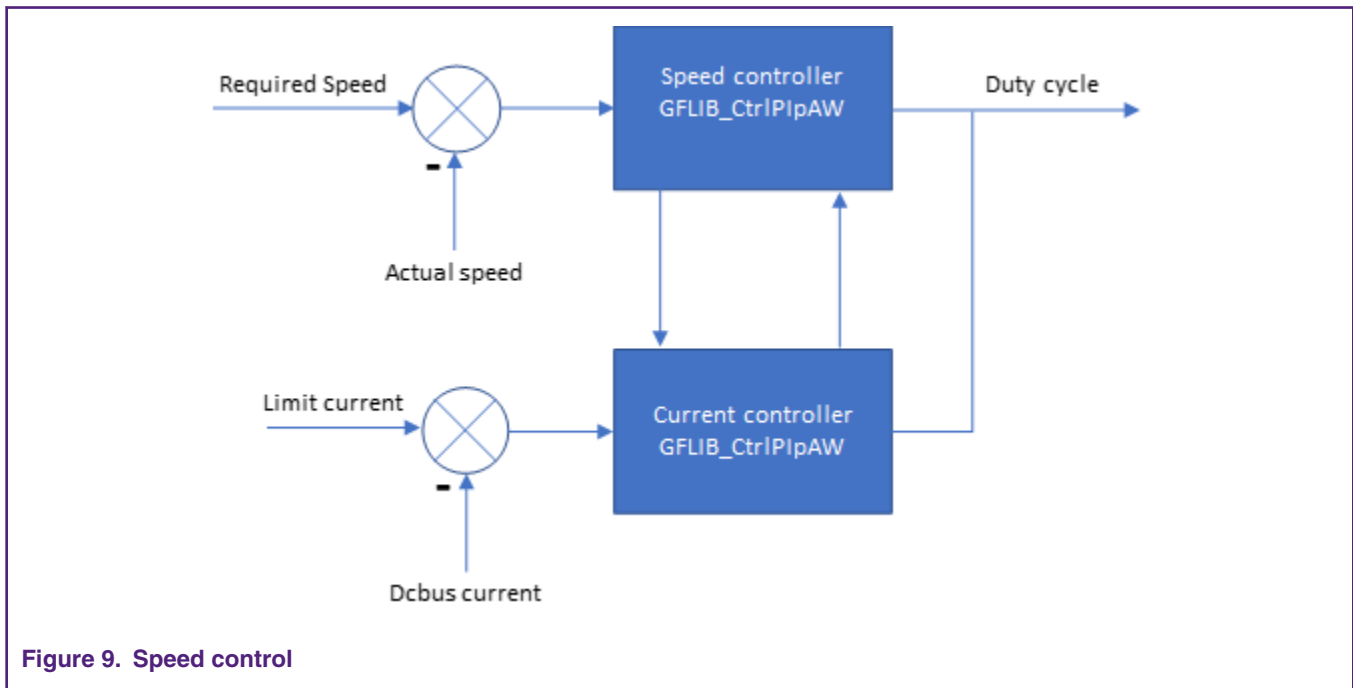


Figure 9. Speed control

## 4 Hardware

The hardware needed to run this demo consists of two parts: the LPCXpresso845max board and the FRDM-MC power stage. To connect them, use a cross-connection board or wires.

### 4.1 LPCXpresso845 board

The LPCXpresso845 board forms the basis of the control system and has these functions:

- It supplies the PWM control signal to the BLDC power stage board.
- It processes the Hall sensor signal and the bus voltage and current values.
- It communicates with the FreeMASTER PC application.

### 4.2 BLDC power stage board

The BLDC board is the Freedom Development Platform for Low-Voltage, 3-Phase BLDC Motor Control (FRDM MC LVBLDC). It is powered by 12 V DC. On this board, you can implement the six-step Hall sensor and sensorless algorithms. In this application note, the Hall sensors are configured and used.

### 4.3 Hardware connection

The hardware connection between the LPCXpresso845 and FRDM MC LVBLDC boards is shown in [Table 2](#).

The functions marked with “\*” are not used in the attached software.

**Table 2. Hardware connection**

Function	LPCXpresso845		FRDM MC LVBLDC	
	Connector	Function	Connector	Function
PWM A Top	J2-8	PIO0_24 (SCT-OUT0)	J3-15	PWM_AT
PWM A Bottom	J2-6	PIO0_21 (SCT-OUT1)	J3-13	PWM_AB
PWM B Top	J2-5	PIO0_20 (SCT-OUT2)	J3-11	PWM_BT
PWM B Bottom	J2-4	PIO1_21 (SCT-OUT3)	J3-9	PWM_BB
PWM C Top	J2-3	PIO1_20 (SCT-OUT5)	J3-7	PWM_CT
PWM C Bottom	J2-2	PIO0_28 (SCT-OUT6)	J3-5	PWM_CB
PHA (Hall)	J2-1	PIO0_1 (PME)	J3-3	ENC_A
PHB (Hall)	J1-10	PIO0_16 (PME)	J3-1	ENC_B
PHC (Hall)	J2-7	PIO0_25 (PME)	J1-3	ENC_I
Current A*	J6-5	PIO0_17 (ADC9)	J2-1	BEMF_A
Current B*	J6-4	PIO0_18 (ADC8)	J2-3	BEMF_B
Current C*	J6-3	PIO0_19 (ADC7)	J2-5	BEMF_C
Volt. DCB	J6-2	PIO0_23 (ADC3)	J2-7	VOLT_DCB
Current DCB	J6-1	PIO0_14 (ADC2)	J2-9	CUR_DCB
3V	J1-3	3V VCC	J3-8(4)	3V VCC
GND	J1-4	GND	J3-12(14)	GND

You can also build a cross-connection board. The source file for manufacturing this board is in the AN package, in the *HW* folder.

## 5 Running example application

The example application for this application note is created in the IAR Embedded Workbench® IDE and includes the RTCESL motor-control library. Use the FreeMASTER real-time debug monitoring tool to run this application.

To run the example application, use an external debugger (for example; J-Link) and install the FreeMASTER PC application.

### 5.1 Application structure

The example application contains the SDK library, the RTCESL library, and the FreeMASTER communication tool.

The main part of the application is in the *source* virtual folder, which contains these files:

- *freemaster\_cfg.h*
- *main.c*
- *main.h*
- *sct\_mc.c*
- *sct\_mc.h*

The *freemaster\_cfg.h* file contains the settings for the FreeMASTER communication tool.

## 5.2 FreeMASTER real-time debug monitor tool

To run the application, install FreeMASTER. FreeMASTER is available at [www.nxp.com/freemaster](http://www.nxp.com/freemaster) and contains the PC and embedded sides. The example application already contains the embedded side.

After installing the FreeMASTER PC side application, a \*.pmpx file is assigned to this application. The \*.pmpx file is the FreeMASTER project file. This file contains information about communication, visualization, and other.

The *SCT\_BLDC\_LPC845.pmpx* file is available in the software structure. When you double-click this file, FreeMASTER opens.

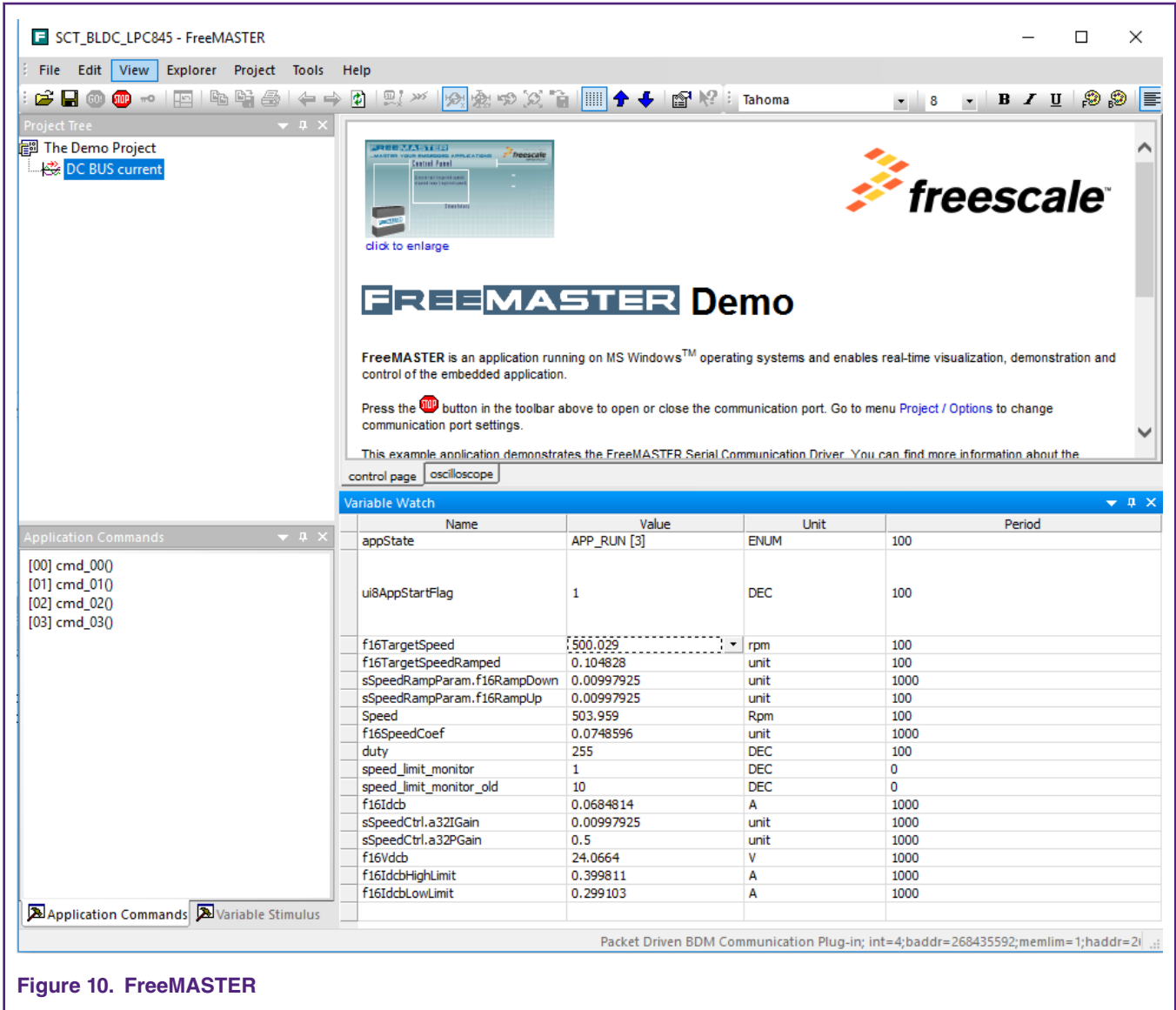


Figure 10. FreeMASTER

The "Variable Watch" area contains the variables that are read from the MCU and used to monitor and control the demo application.

## 5.3 Starting a motor

When the hardware is connected properly and the application in the MCU is running, connect FreeMASTER to the target device.

The example project is set to the packet-driven communication, which uses an external J-Link debugger. The external J-Link debugger is used for programming and communication.

To connect FreeMASTER to a target board, click the green "GO" button (it is located in the top-left side of the FreeMASTER window).

When the communication is running, start the motor. For this, change "ui8AppStartFlag" to "1" and set "f16TargetSpeed" in the range of 500-4700 rpm. After this, the RTCESL ramp function ramps the speed from zero to the target speed.

The motor rotation direction can be changed using the "s\_Mdirection" variable (change the value from 0 to 1 and the other way round).

## 6 Conclusion

This application note introduces the basic theory of BLDC motor control and the six-step commutation. This document also describes the PWM control through SCTimer. The FreeMASTER PC tool is used to control this application together with the RTCESL motor-control library. The manufacturing data for a cross-connection board are also included in the software package.

## 7 References

The following documents are available at [www.nxp.com](http://www.nxp.com):

1. *SCTimer/PWM Cookbook* (document [AN11538](#)).
2. *LPC84x User Manual* (document [UM11029](#)).

## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://nxp.com/support)

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](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2019.

All rights reserved.

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

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: October 2019

Document identifier: AN12602

