Freescale Semiconductor
Application Note

Document Number: AN4429
Rev. 0, 12/2011

Using Motor Control eFlexPWM (mcPWM) for BLDC Motors

by: Libor Prokop

Contents

1 Introduction

This application note describes setting the Enhanced Flex Pulse Width Modulator (eFlexPWM or mcPWM) for BLDC motor control applications. It includes a brief description of the eFlexPWM, the eFlexPWM peripheral's initialization, commutation techniques, and the PWM duty cycle control. The description in this application note will cover the advanced eFlexPWM modules built on the MCF56F825x/4x and MC56F84xx devices. Basically similar PWM modules are implemented on MCF5441x, PXS20, Power Architecture®, and other devices. All these PWM modules are targeted mainly at sophisticated motor control and inverter applications.

So, this application note can be applied to setting the eFlexPWM implemented on the MCF56F825x/4x and MC56F84xxx devices, the Motor Control Pulse Width Modulator mcPWM implemented on the MCF5441x, and the Flexible Motor Control Pulse Width Modulator Module implemented on PXS20. However, although the denomination of those PWM modules and register names differ slightly, the basic and main features are the same.

All settings in this application note will be described for the eFlexPWM on the MCF56F825x/4x devices. The setting up of the other above mentioned PWMs is the same, if not mentioned otherwise.

This application note does not describe setting up the PWM module implemented on the MC56F8006, MC56F83xx, or MCF56F80xx. BLDC motor control for those PWMs is described in other application notes and design reference manuals, such as DRM026, DRM108, and other articles. Also, no FlexTimer module, for example, implemented on the HCS08MP16, ColdFire, Kinetis, and other devices, setting is described here. Therefore, see AN3729, DRM117, and other articles in http://www.freescale.com for those devices.

2 Enhanced PWM (eFlexPWM) module overview

The eFlexPWM has four PWM submodules, each of which is set up to control a single half-bridge power stage, complementary channels, or possibly eight independent channels. It supports the generation of PWM signals to control electric motor and power management applications. The eFlexPWM time reference is a 16-bit counter that can be used as an unsigned or signed counter. The PWM module works with six output channels in the 3-phase motor control application. These six channels have one source clock signal and one common counter. Each channel has its own time for the rising and falling edge of the signal.

2.1 Enhanced PWM module (eFlexPWM) features

Some features of enhanced PWM module are:

- An eFlexPWM module contains four PWM submodules, each of which is set up to control a single half-bridge power stage.
- 16 bits of resolution for centre, edge aligned, and asymmetrical PWMs.
- Fractional delay for enhanced resolution of the PWM period and edge placement.
- PWM outputs that can operate as complementary pairs or independent channels.
- Ability to accept signed numbers for PWM generation.
- Independent control of both edges of each PWM output.
- Support for synchronization to external hardware or other PWMs.
- Double buffered PWM registers.
  - Integral reload rates from 1 to 16.
  - Halfcycle reload capability.
- Multiple output trigger events can be generated per PWM cycle via hardware.
- Support for double switching PWM outputs.
- Fault inputs can be assigned to control multiple PWM outputs.
- Programmable filters for fault inputs.
- Independently programmable PWM output polarity.
- Independent top and bottom dead time insertion.
- Each complementary pair can operate with its own PWM frequency and dead time values.
- Individual software control for each PWM output.
- All outputs can be programmed to change simultaneously via a FORCE_OUT event.
- The PWMX pin can optionally output a third PWM signal from each submodule.
- Channels not used for PWM generation can be used for buffered output compare functions.
- Channels not used for PWM generation can be used for input capture functions.
- Enhanced dual edge capture functionality.
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - Crossbar Switch (XBAR) module outputs.
  - External ADC input, taking into account values set in ADC high- and low-limit registers.

2.2 MC56F8247 PWM signal path

This section describes the PWM signal path from the beginning to its output on the MC56F8247.

It is necessary to understand how the PWM signal is generated across the submodules. In Figure 1 each pair of PWM signals is generated in a separate submodule. Submodule 0 is the master module and performs the synchronization of the remaining modules.

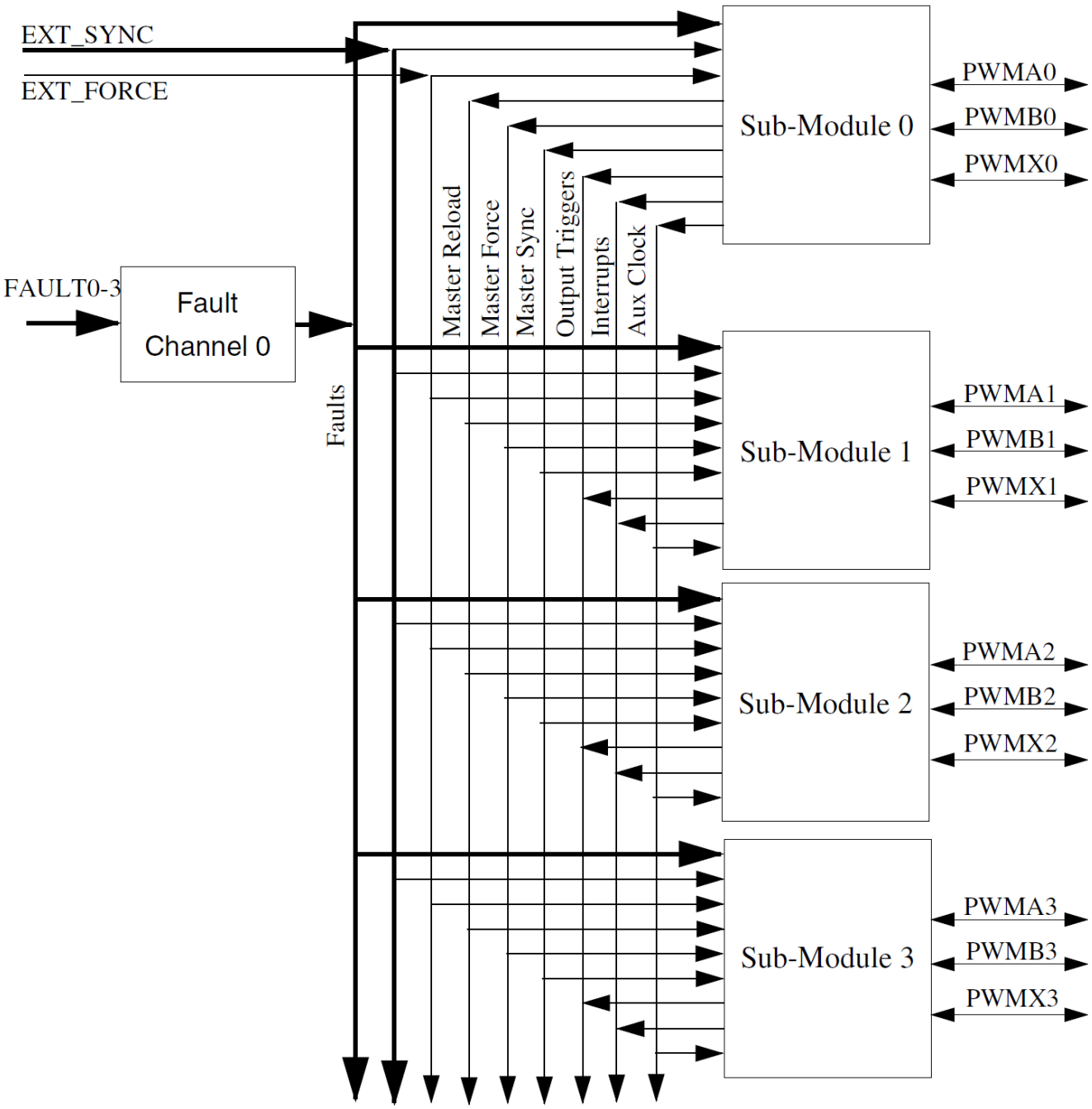

Figure 1. PWM modules

Let us focus on one submodule and analyze in detail (see Figure 2). There are six comparators with a 16-bit counter input and an init signal. Pay attention to those six values and the init signal.

The init signal determines the beginning of the PWM period and the VAL1 value ends it. The PWM period then counts from a value given by the init through signal to VAL1. There is a VAL0 value that defines the mid-point of the PWM period, and this is where the half-cycle reload is defined.

The VAL2 and VAL3 values determine the top PWM on-times and off-times, and, similarly VAL4 and VAL5 determine the bottom PWM on-times and off-times. The output top and bottom PWM values are named PWM23 and PWM45.

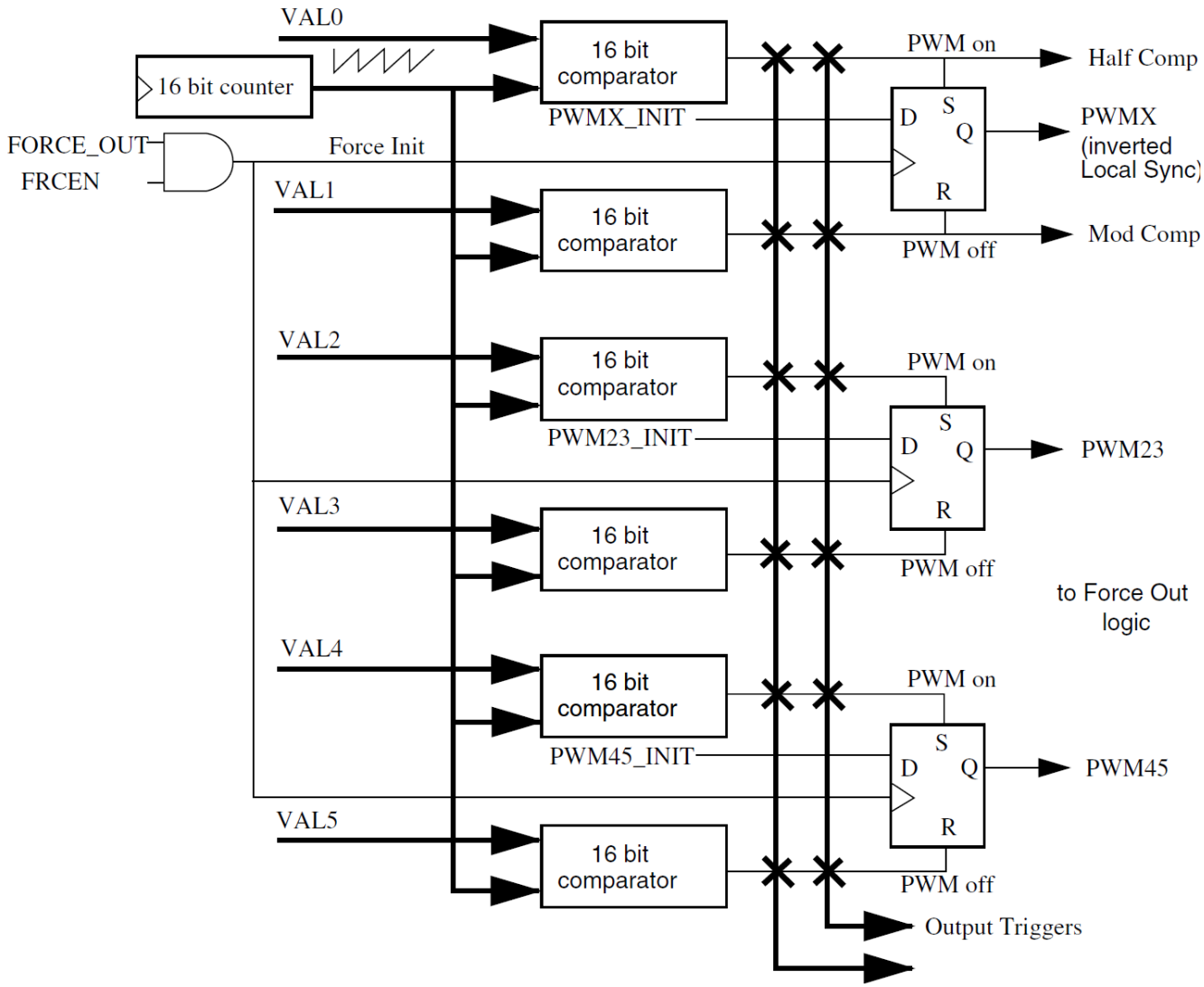

Figure 2. PWM generator

In Figure 3, it can be seen how the outputs from the PWM generator are forced out. Select the PWM as it is or its inverted value, the predefined values OUT23 and OUT45, and the external values EXTA and EXTB on the outputs.

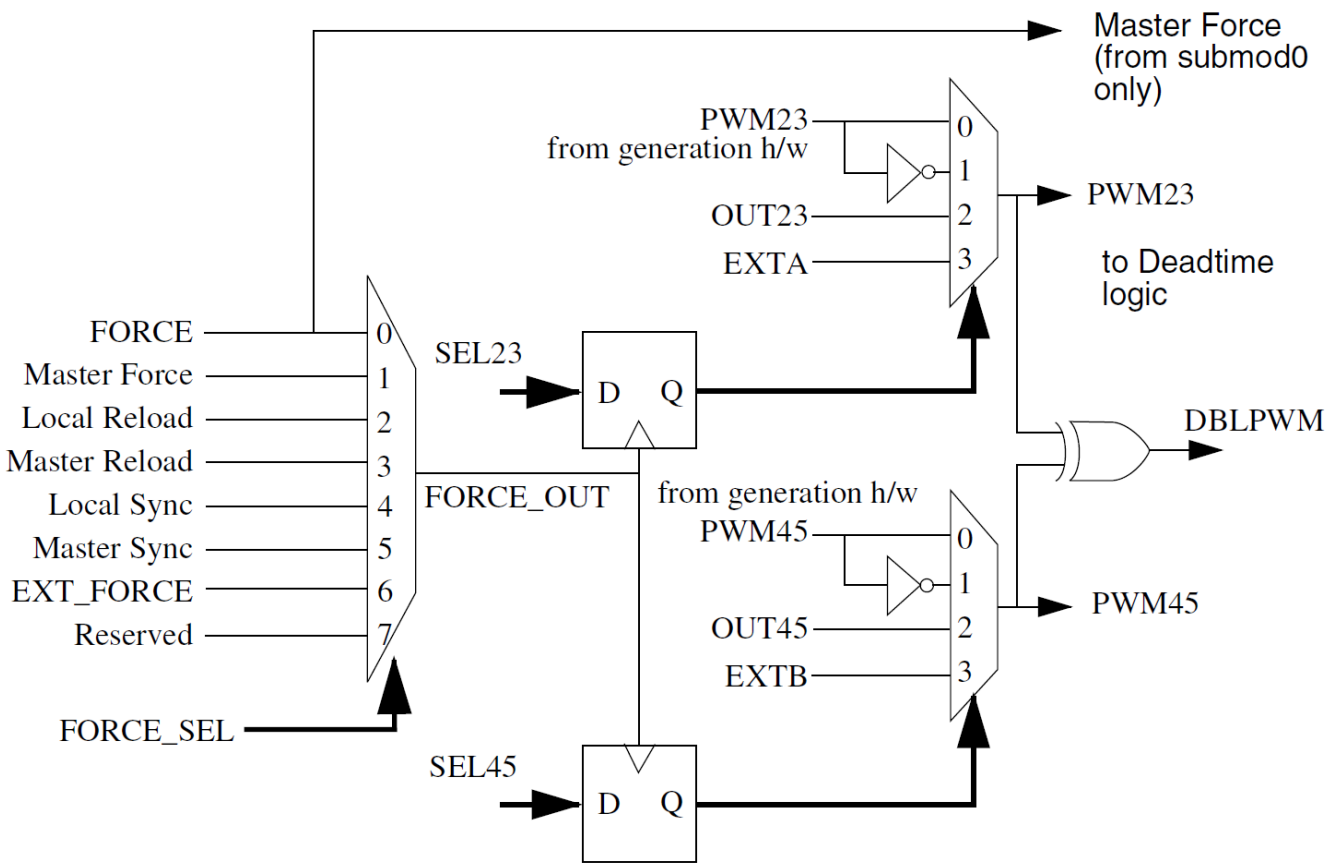

Figure 3. Force-out logic

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired, as shown in Figure 4, in the complementary channel operation. MCTRL[IPOL] determines the signal to be connected to the output pin, that is, PWM23 or PWM45.

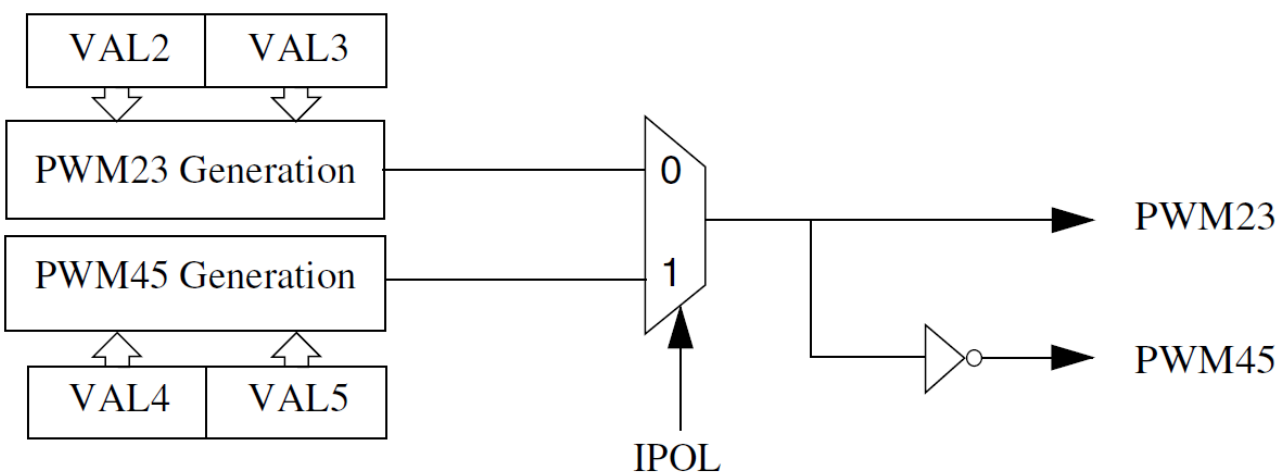

Figure 4. Complementary channel pair

In Figure 5, the output from the force-out logic continues to the dead time insertion logic. If the PWM generation is independent, this logic is bypassed. If it is complementary, the polarity of the signal is selected and then the particular dead time is inserted.

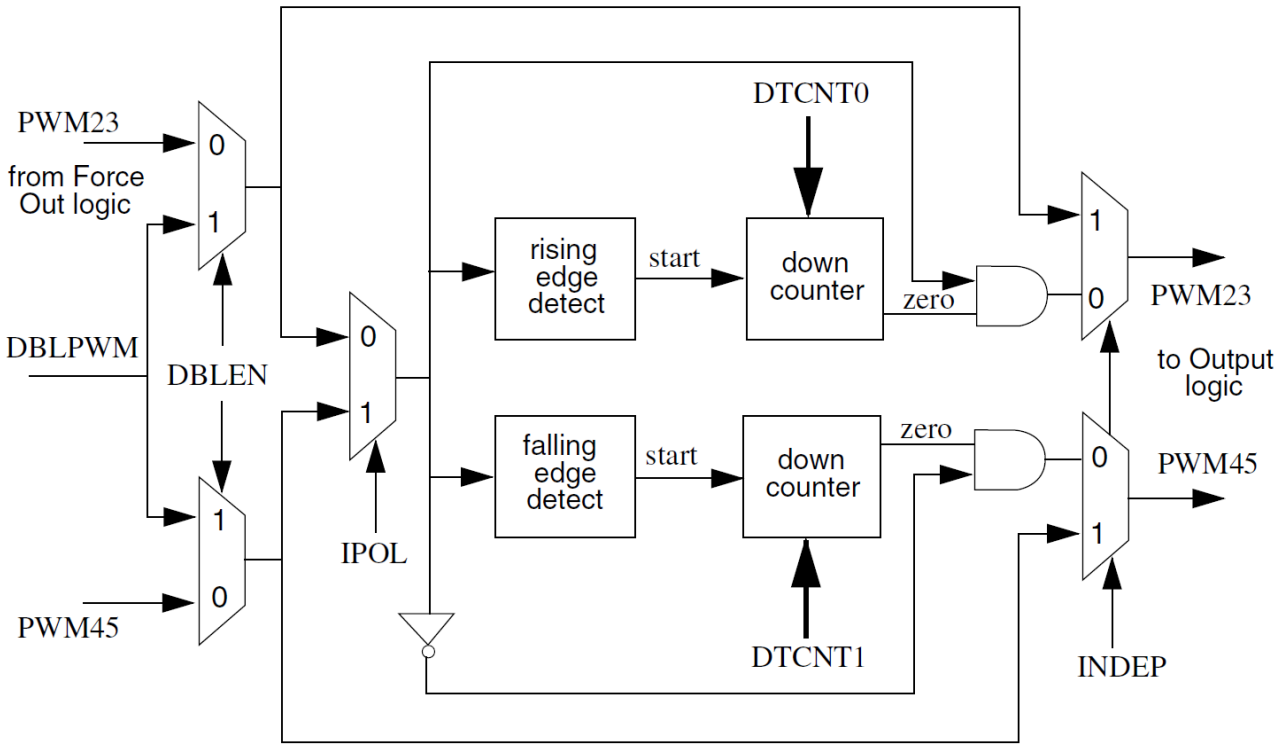

Figure 5. Dead time insertion logic

The last stage of the signal is the output logic (see Figure 6). The PWM continues via the fractional delay blocks in the case of a fractional PWM.

In case of no fault, the signal goes into the MASKA and MASKB gates that determine if the signal is masked or not. The following gate then determines the polarity of the signal. Finally, the output gate that turns on and off the PWM output depends on the conditions on the PWMA_EN and PWMB_EN. There is a logic that depends on the fault status. The logic can be set up to force a 0, 1, or a tristate on the output.

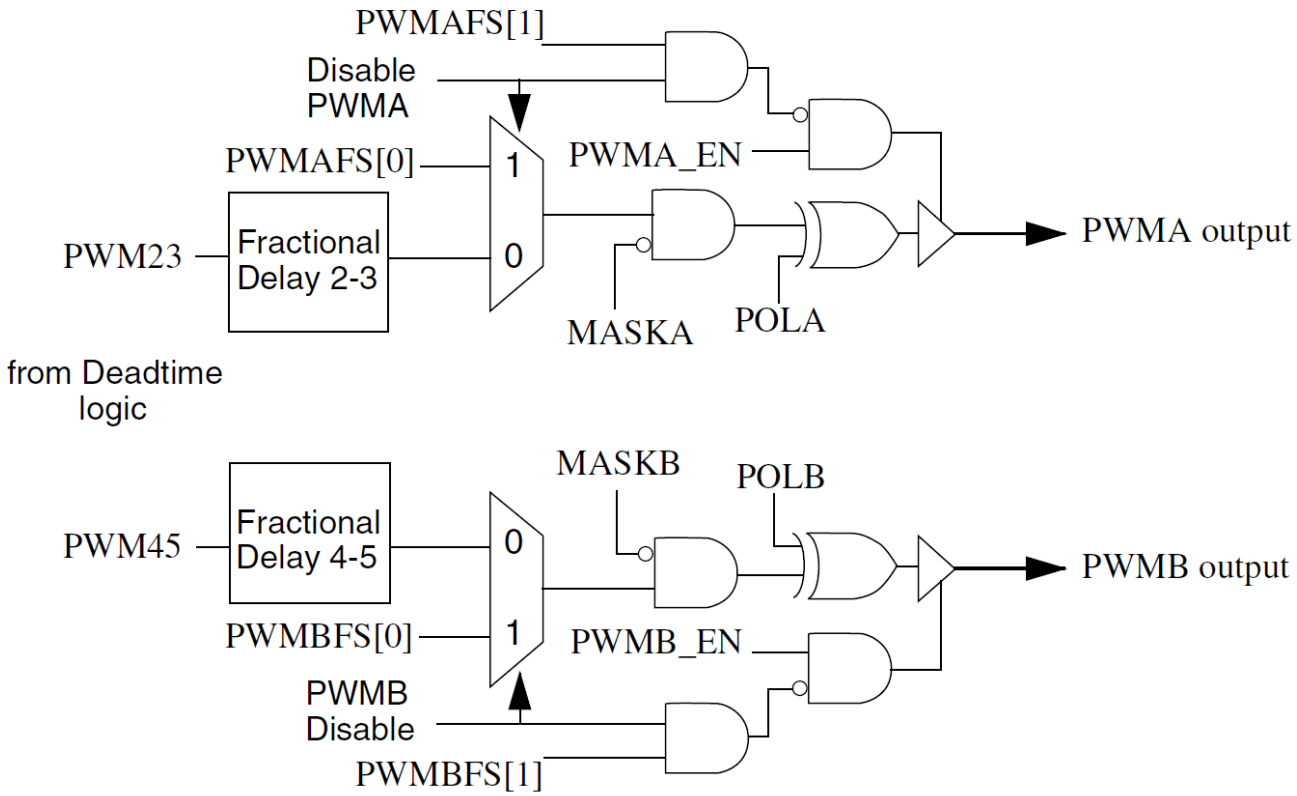

Figure 6. Output logic

3 PWM module and 3-phase BLDC motor control

This section concentrates on the utilization of a PWM module to control a 6-step BLDC commutation of a 3-phase trapezoidal BLDC motor.

3.1 6-step commutation for a 3-phase BLDC motor control

The BLDC motor rotation is controlled by a 6-step commutation technique, sometimes called 60, 120 degree control. A simplified principle is shown in Figure 7. The 6-step technique creates the voltage system according to Figure 7, with six vectors over one electronic rotation. One of the most important characteristics of the 6-step BLDC motor control is that one of the phases is switched off at any given time. The applied voltage is controlled with a PWM technique.

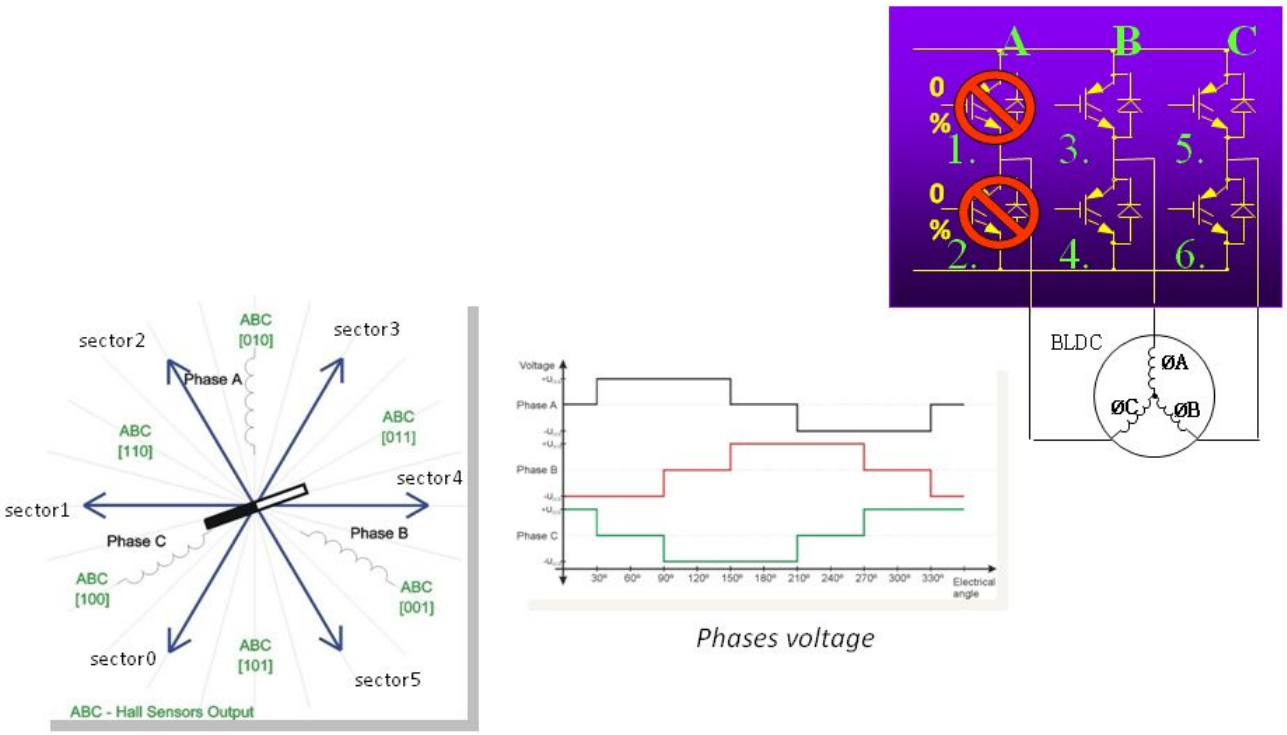

Figure 7. 6-step BLDC commutation for a 3-phase voltage system

3.2 3-phase power stage

The 3-phase 6-step voltage system is created by a 3-phase power stage with six IGBTs (MOSFET) power switches controlled by the MCU on-chip PWM module (see Figure 8). This PWM module is realized by six channels of a PWM module.

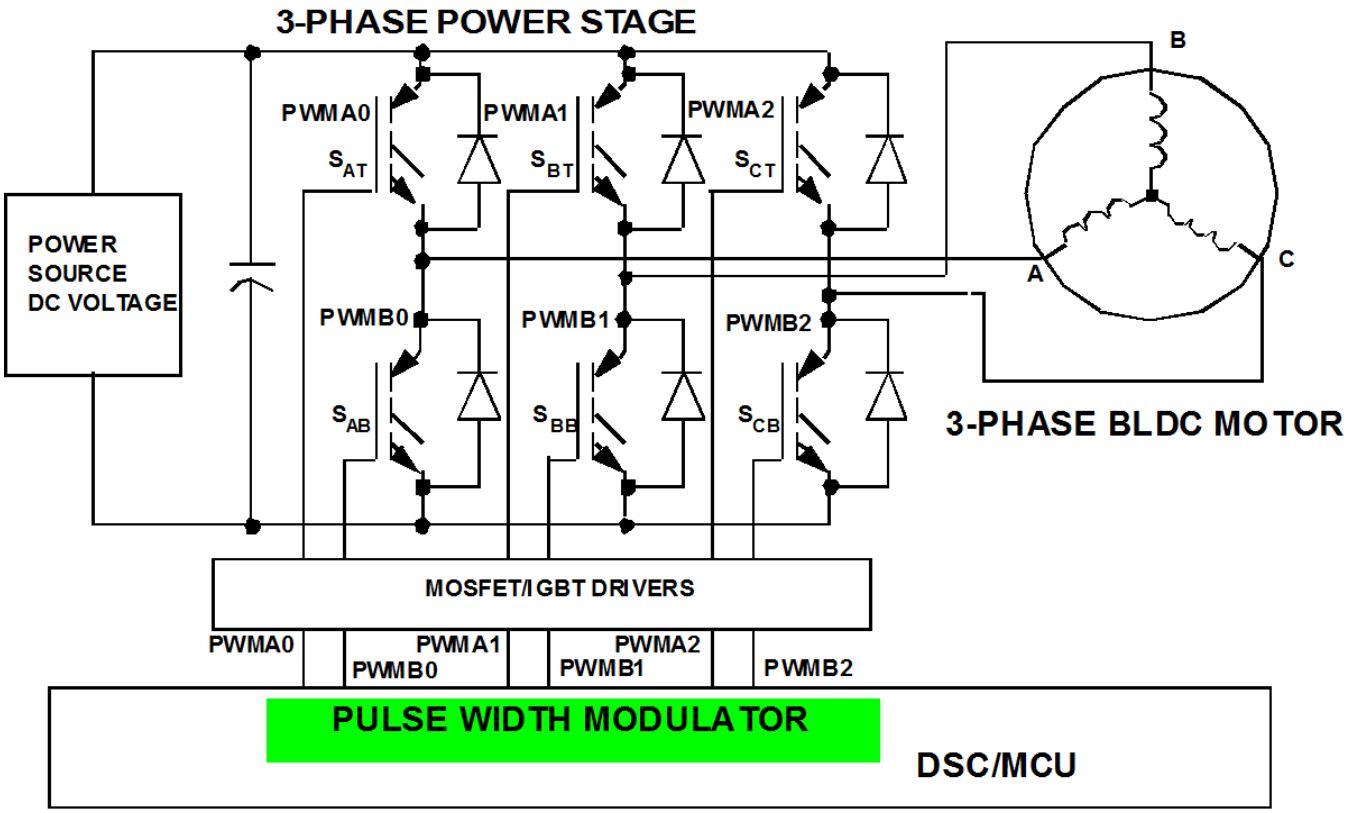

Figure 8. Power stage — motor topology

3.3 Voltage amplitude controlled by PWM

The amplitude of a 3-phase voltage system needs to be controlled. The most common BLDC control topology uses the power stage from Figure 8 with a constant power source DC voltage. Therefore, the 3-phase average voltage amplitude is controlled by a PWM technique of the top and bottom transistors. The 6-step controller uses one of the following two PWM techniques:

1. Bipolar PWM switching
2. Unipolar PWM switching

There are a few derivatives of the two PWM switching techniques. According to the operating quadrants of the power stage voltage and current, following two derivatives are found:

1. 4-quadrant power stage control
2. 2-quadrant power stage control

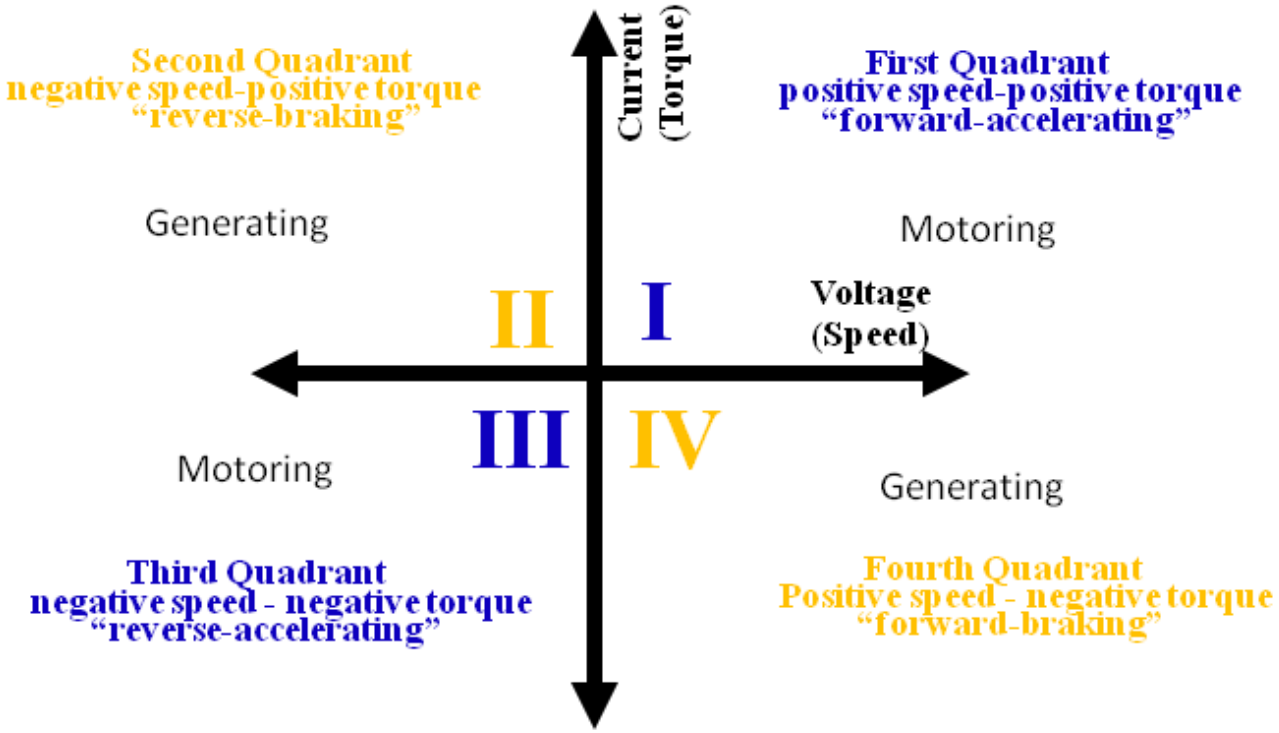

Figure 9. Quadrant of operations

The bipolar/unipolar switching and the operating quadrant control are determined by all the six SAt SCb switchings. The 2-quadrant operation provides a defined voltage and current of the same polarity, that is, positive voltage with a positive current or a negative voltage and negative current, which are the operating quadrants I and III in Figure 9. The 4-quadrant PWM switching covers the operation of the generated voltage in all four quadrants. This is provided using complementary switching of the top and bottom transistors. The benefits of the 4-quadrant PWM operation are:

- Motoring and generating mode control, which gives the possibility of braking the motor
- Linear operation in all four quadrants
- Lower static power losses when using MOSFET switches

The FlexPWM is able to control the 3-phase power stage with unipolar and bipolar PWM, with any of the required operations. In the following sections, we will mainly concentrate on the more advanced 4-quadrant operations.

3.3.1 3-phase BLDC 6-step control with bipolar (hard) PWM switching

The PWM commutation pattern of the bipolar complementary, 4-quadrant PWM switching technique is shown in Figure 10.

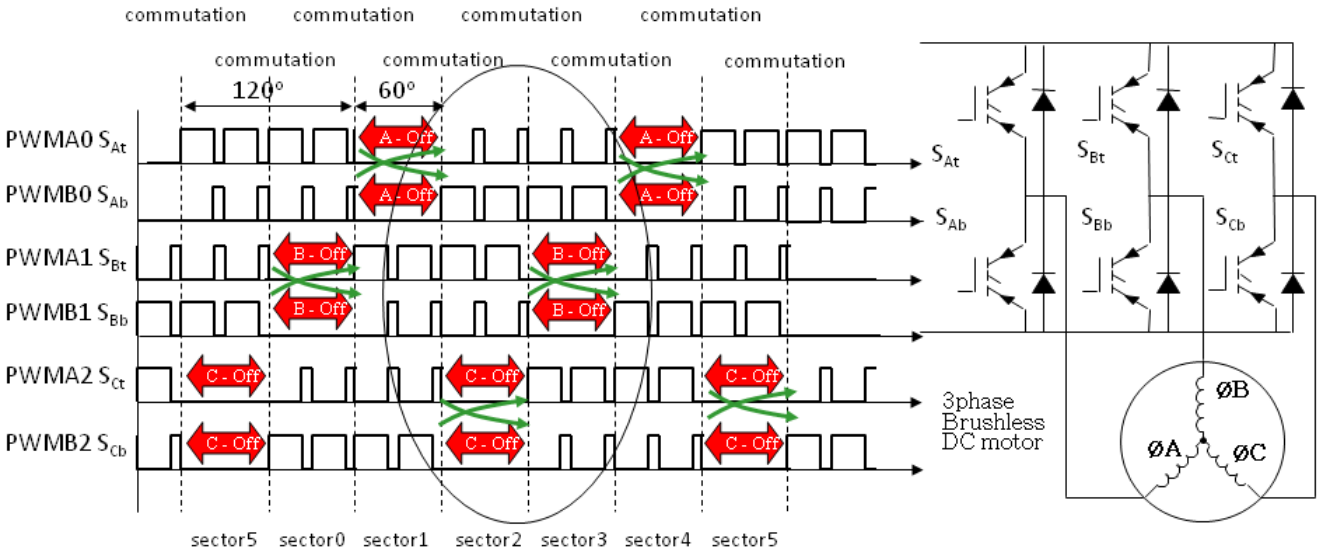

Figure 10. Bipolar PWM switching

Details of the technique are shown in Figure 10. From Figure 11, the characteristic of the bipolar complementary 4-quadrant switching can be clearly seen. The bipolar switching requires that the top and bottom switch PWM signals need to be swapped. Another important detail is the introduction of dead time insertion in the complementary top and bottom signals. This dead time insertion is typical for all 4-quadrant power stage operations. The 4-quadrant operation is enabled by the complementary operation of the top and bottom switches. The bottom switch of one phase is almost the negative of the top switch.

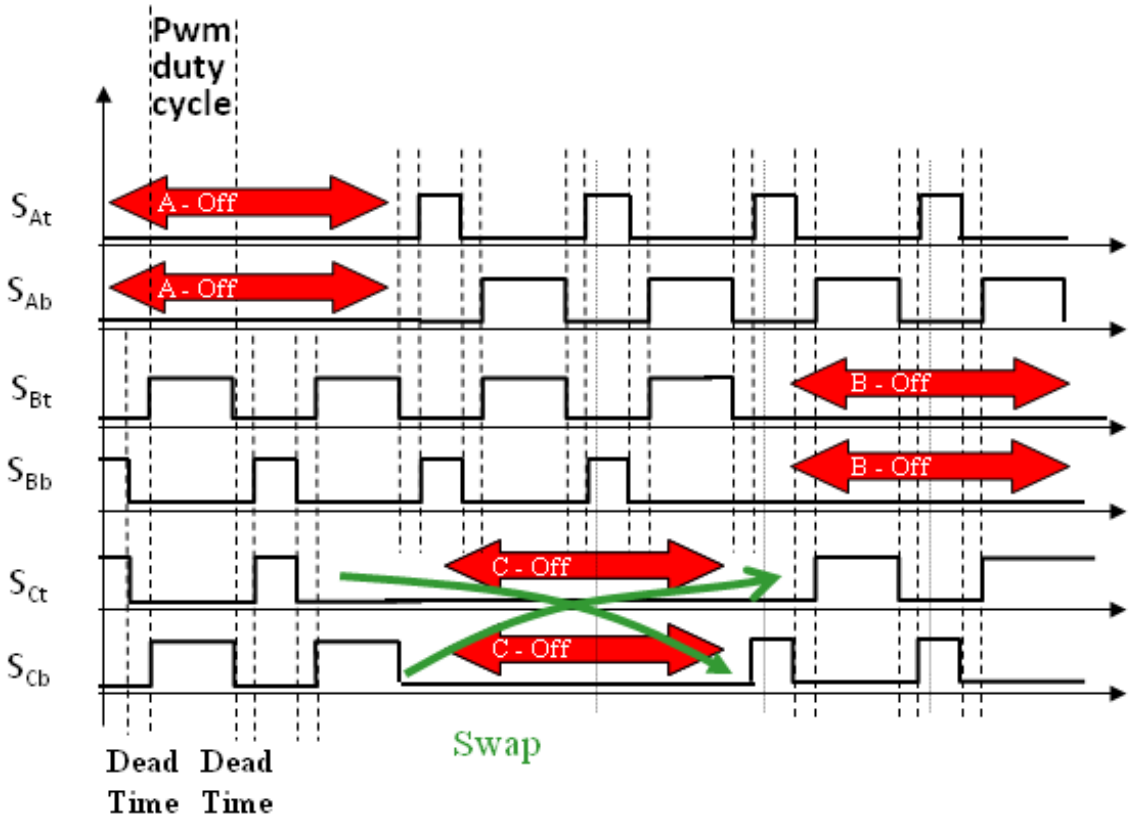

Figure 11. Bipolar PWM switching detail

The 4-quadrant operation requires the insertion of a dead time because the switching transient will cause a DC-Bus short circuit with fatal power stage damage. The bipolar PWM switching is not as popular as the unipolar switching due to a worse electromagnetic emission of the motor. This is because the PWM ripple is twice that of the DC-Bus voltage. On the other hand, this switching is better for sensorless rotor position sensing.

3.3.2 3-phase BLDC 6-step control with unipolar (soft) PWM switching

There are further modifications to the unipolar PWM switching. One of the most common 6-step commutation with unipolar complementary 4-quadrant PWM switching techniques is shown in Figure 12. The unipolar PWM switching can control the BLDC motor with an almost identical voltage and current performance. The only difference is that the PWM duty cycle = 0 creates an average voltage of 0. The main advantage of the unipolar PWM switching is better electromagnetic compatibility (EMC) due to half of the voltage ripple as compared to the bipolar switching

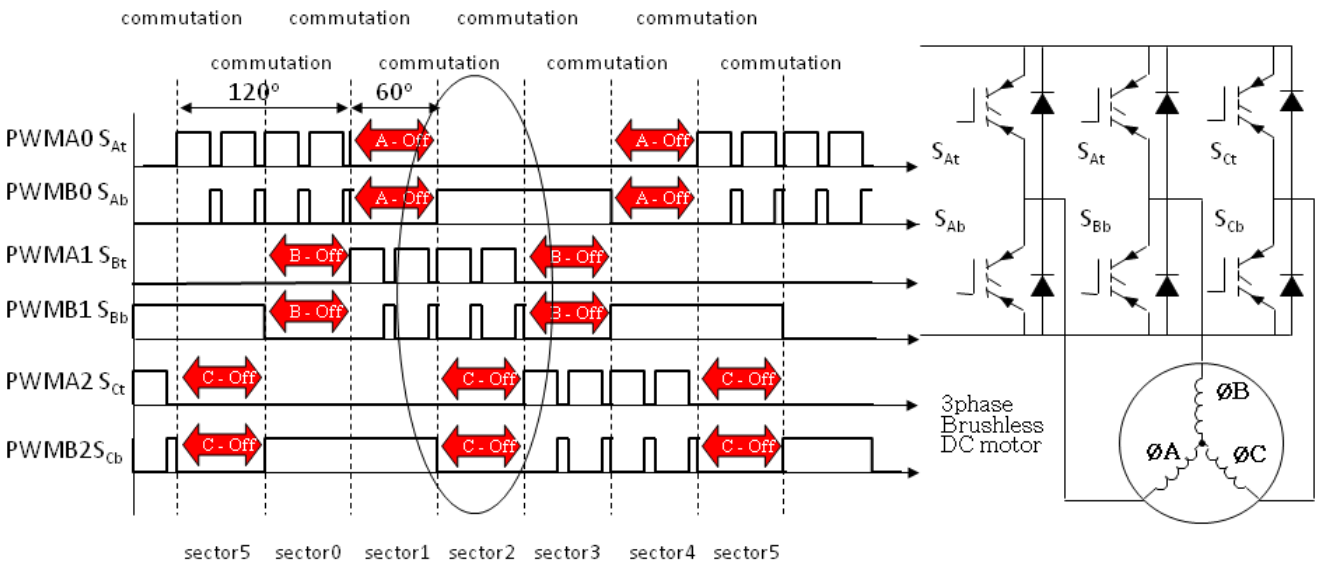

Figure 12. Unipolar PWM switching

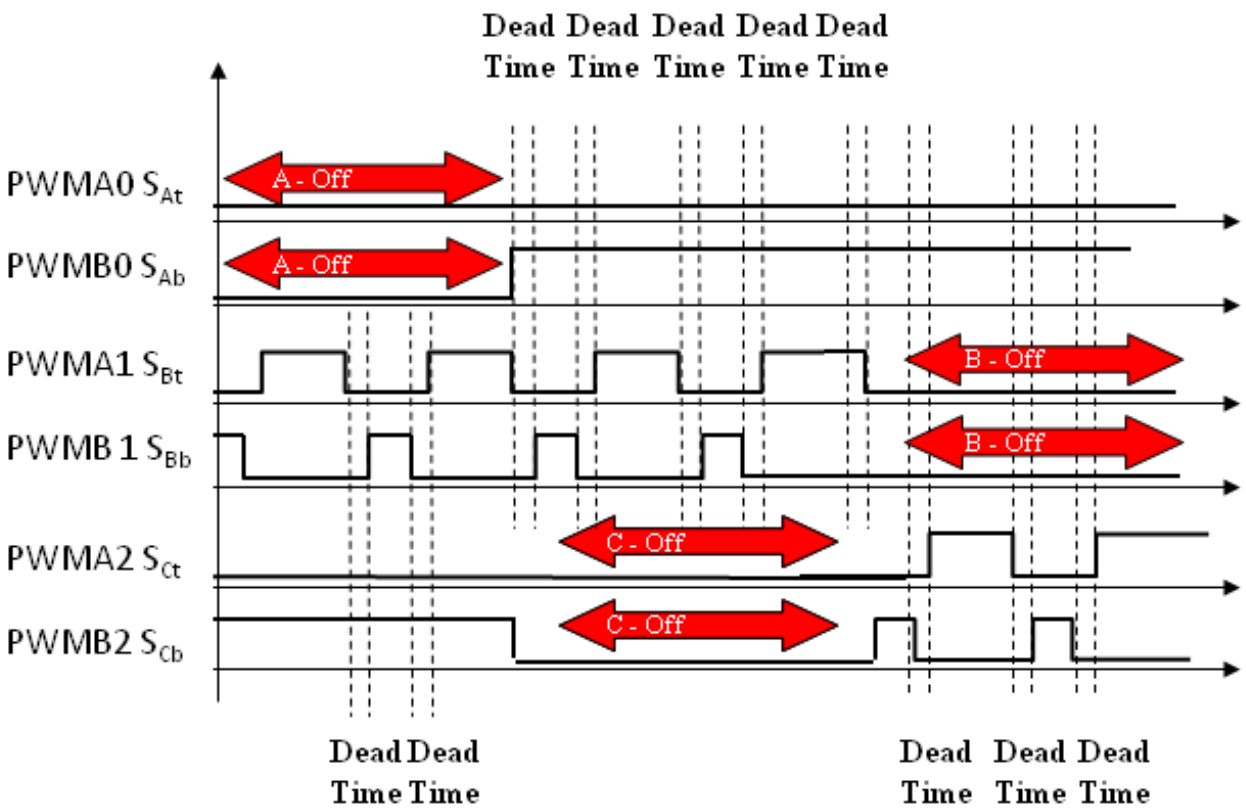

Figure 13. Unipolar PWM switching detail

4 Using MC56F8247 FlexPWM module for BLDC motor control

This section describes setting the MC56F8247 FlexPWM for BLDC motor control.

4.1 Common FlexPWM initialisation

Some PWM module settings are specific for the BLDC commutation technique which is used here. Other settings are common for any BLDC or even any motor control applications. This section describes the common settings.

4.1.1 MCF56F8247 PWM module activation

The PWM module activation is device specific. The PWM module requires the power and clock signals to be enabled.

Configuration of the MCF56F8247 device is described in this section. To enable the clock for the PWM module sub-blocks 0, 1, and 3, you must set up the PWMCH0 bit (3), the PWMCH1 bit (2), and the PWMCH2 bit (1) of the register SIM_PCE2 to 1. It is necessary to set up the GPIO E pins and the Crossbar Switch (XBAR) signals to be used as the PWM outputs and fault inputs. The GPIOE bit (2) of SIM_PCE0 must be 1 to turn on the GPIO E module. The E5 bit (6) and the E4 bit (4) of SIM_GPS3 must be cleared. These bits select the PWM 2A and 2B as PWM, not XBAR. The bits 0–5 of the GPIOE_PER register must be set to use these pins as PWM, not GPIO.

4.1.2 Complementary edge-aligned PWM

For a BLDC 4-quadrant operation, the complementary PWM mode is necessary. To select the complementary mode for each pair of PWM outputs, it is necessary to clear the INDEP bit (13) of the PWM_CTRL2 register of the submodules 0, 1, and 2 to the value of 0.

Edge-aligned PWM is more suitable for BLDC motor commutation. This is provided through setting the PWM_SMnVAL2 (PWM_SMnVAL4) registers to PWM_SMnINIT = 0 and PWM_SMnVAL3 (PWM_SMnVAL5) to the required duty cycle with the maximum of PWM_SMnVAL1. On the other hand, in the case of centre-aligned PWM, the VAL2 and VAL3 values must be supplied symmetrically between the PWM_SMnINIT and PWM_SMnVAL1values. This setting needs to be applied to the three PWM submodules n = 0, 1, 2.

Example 1 Pseudo code – PWM duty cycle Setting

```
short pwmDutyCycle;/* signed 16 bit  <- MODULO, + MODULO> */
MCPWM.PWM_SMnVAL2 = 0;
MCPWM.PWM_SMnVAL3 = pwmDutyCycle;/* n = 0, 1, 2*/
```

4.1.3 PWM duty cycle initialization

In submodule 0, the register PWM_SM0CTRL2 bit RELOAD_SEL = 0 determines that the local RELOAD signal is used to reload the registers. The submodules 1 and 2 are synchronized with submodule 0 by RELOAD_SEL = 1. This means that the master RELOAD signal from submodule 0 is used to reload registers PWM_SM1CTRL2 and PWM_SM2CTRL2.

4.1.4 PWM clock frequency

The clock source of the sub-block 0 will be set up to the IPBus clock that means the CLK_SEL bits of the PWM_SM0CTRL2 register will be 0. Sub-blocks 1 and 2 must be set up to use the clock source of the sub-block 0, CLK_SEL = 10 . The prescaler is set up by the PRSC bits of the PWM_CTRL register. These bits must be 0 for each sub-block.

Finally, the LDOK bits (0, 1, 2) of PWM_MCTRL for each sub-block must be set to the values to be applied,.

Example 2 Pseudo code – PWM Clock Setting

```
/* CLK_SEL = 00 The IPBus clock is used as the clock for the local prescaler and counter. */
MCPWM.PWM_SM0CTRL2 = 0x0000;

/* submodules n = 1,2 CTRL2 Setting*/
/* CLK_SEL = 10  Submodule 0's clock (AUX_CLK) is used as the source clock for the local
prescaler and counter. */
MCPWM.PWM_SMnMCTRL2 = 0x000A;
```

The clock for the PWM module is fIPBus = 60 MHz. To achieve the PWM frequency of fPWM and keeping the prescaler /1 set up the modulo:

```
MODULO = fIPBus/fPWM
```

MODULO = fIPBus/fPWM for edge-aligned PWM, the INIT register 0 of each sub-block and the VAL1 register is MODULO in the sub-block 0. To enable the PWM module, set up the RUN bit (8) of the register PWM_MCTRL to 1. To achieve a PWM frequency of 15 kHz with a fIPBUS 60 MHz the VAL1 = MODULO = 4000 = 0x0FA0U.

Example 3 Pseudo code – PWM Modulo Setting

```
MCPWM.PWM_SM0INIT = 0;
MCPWM.PWM_SM0VAL1 = MODULO = 4000 = 0x0FA0;
```

4.1.5 Dead time setting

For the 4-quadrant PWM operation, the complementary PWM with dead time insertions is necessary. The dead time is derived from the PWM module clock fIPBUS.

DTCNT0 = DTCNT1 = dead time/fIPBus

To set up the dead time = 2 µs at fIPBUS = 60 MHz, it is necessary to fill the registers DTCNT0 and the DTCNT1 with the values of DEADTIME0_INIT = 0x0078U.

These registers must be updated in all the used sub-blocks (0, 1, 2).

Example 4 Pseudo code – PWM Dead Time Setting

```
/* dead time 1 IPBus clock cycles (regardless of the setting of CTRL[PRSC]) */
MCPWM.PWM_SMnDTCNT0 = DEADTIME0_INIT;
/* dead time 1 IPBus clock cycles (regardless of the setting of CTRL[PRSC]) */
MCPWM.PWM_SMnDTCNT1 = DEADTIME_INIT1;
```

4.1.6 PWM trigger and the timer to ADC synchronization

The PWM OUT_TRIG0 signal is usually needed for synchronization with the ADC sampling or other timers. The trigger signal is generated when the PWM counter PWM_SMnCNT = PWM_SMnVAL0.

Example 5 Pseudo code – PWM OUT_TRIG0 Trigger Setting

```
/* OUT_TRIG0 will set when the counter value matches the VAL0 value. */
MCPWM.PWM_SM0TCTRL =  0x0001;

/* value for trigger OUT_TRIG0 signal */
MCPWM.PWM_SM0VA0 = abs(pwmDutyCycle)/2;
```

The *abs(pwmDutyCycle)/2* is usually used for sampling in the middle of the first edge-aligned PWM stage. This is usual for current sensing.

4.1.7 Output polarity and fault settings

Some 3-phase inverters require negative logic. The FlexPWM output polarity is defined in the PWM_SMnOCTRL register by bits POLA, POLB, and POLX.

At a fault state, the A, B, and C outputs of each submodule are disabled with FAULTx (x = 0,1,2,3) signals according to the PWM_SMnDISMAP configuration. DISA = FAULT3, FAULT2, FAULT1, FAULT0.

One of usual h/w configuration is that the FAULT0, for example, overcurrent, and FAULT1 signals disable the top and bottom signals of all submodules. Such configuration requires that the PWM_SMnDISMAP register of the sub-block 0, 1, and 2 must be filled with the values PWM_0_DISMAP_INIT = 0xF033.

To set up the off output of the channels in the case of a fault signal, it is necessary to fill the PWM_PWMAFS and PWM_PWMBFS bits of the sub-block 0, 1, and 2 PWM_OCTRL registers to the value of 0. A fault state output is forced to a logic 0 state prior to consideration of the output polarity.

Example 6 Pseudo code – Output Polarity and Settings

```
/* POLA = 1 output A - low-level active, PWMAFS = 0,  PWMBFS = 0, PWMXFS = 0 Fault state
output is forced to a logic 0 state prior to consideration of the output polarity control */
MCPWM.PWM_SMnOCTRL = 0x0400;

/* disable DISXn, DISBn, DISAn outputs at FAULTx defined in the s/w constant
PWM_0_DISMAP_INIT */

MCPWM. PWM_SMnDISMAP = PWM_0_DISMAP_INIT;
```

Any possible fault flags (FFLAG) are to be cleared in the PWM_FSTS register by writing 1 to all fault flags, that is 0x0F.

4.1.8 PWM interrupts

The FlexPWM module allows a reload, fault interrupts, and reload error interrupts. The reload error is generated when the PWM values are not coherent. An additional interrupt can be generated on the compare event of the values 0–6. The user can only select the particular compare events as the source for the interrupt.

Each PWM sub-block has independent interrupts. The setting is provided in the *PWM_SMnINTEN* register.

For BLDC motor control usually the fault interrupt is only needed. These interrupts are to be enabled in the *PWM_FCTRL* register. To enable interrupts at FAULT0 and FAULT1:

Example 7 Pseudo code – Fault Interrupt Enable

```
/* FIE10 = 0x0003 enable interrupts of FFLAG 0,1 caused by FAULT0 and 1*/
PWM_FCTRL |= FIE10;
```

4.1.9 PWM outputs and counter clocks enable

The *PWM_MCTRL[RUN]* read/write bit enables the clocks to the PWM generator. When this bit equals zero, the submodule counter is reset. A reset clears this field.

The *PWM_OUTEN* enables the outputs PWMAn, PWMBn, and PWMXn of any PWM submodule.

Example 8 Pseudo code – PWM Outputs and Clock Enable

```
/* PWM clock enable */
MCPWM.PWM_MCTRL |= MCTRL_RUN210; /* MCTRL_RUN210 =  0x0700 */

/* PWM Output Enable */
MCPWM.PWM_OUTEN |= (OUTEN_PWMA_EN_SUB0|
OUTEN_PWMB_EN_SUB0|
OUTEN_PWMA_EN_SUB1\
OUTEN_PWMB_EN_SUB1\
OUTEN_PWMA_EN_SUB2\
OUTEN_PWMB_EN_SUB2; /* = 0x0770 */
```

4.2 6-step commutation using the FlexPWM

This section describes the following commutation technique settings:

- Complementary bipolar PWM 6-step BLDC commutation
- Coherent complementary unipolar PWM 6-step BLDC commutation
- Standard complementary unipolar PWM 6-step BLDC commutation
- 2-quadrant (independent) unipolar PWM 6-step BLDC commutation
- 2-quadrant (independent) bipolar PWM 6-step BLDC commutation

4.2.1 Commutation using PWM_MASK, PWM_DTSRCSEL, PWM_SWCOUT, and PWM_MCTRL

The BLDC commutation is mainly provided by utilizing one or all of the registers: *PWM_MASK*, *PWM_DTSRCSEL*, *PWM_SWCOUT*, and *PWM_MCTRL*. The *PWM_MASK* is for masking the off phase. The *PWM_DTSRCSEL* and *PWM_SWCOUT* are for setting the negative/positive PWM or SWCOUT controlled signal. And the *PWM_MCTRL* is used to select the PWM23 or PWM45 signals of different duty cycle using the IPOL bits. The registers are written by the software, but they are applied on the called Force-Out event. This event can be manually written into a register *PWM_SM0CTRL2[FORCE]*, connected to an external signal such as a timer or the PWM reload. In the sensorless BLDC control, the timer compare event can make the commutation without any software interaction. See the sections PWM trigger for s/w commutation and PWM trigger for h/w commutation (EXT_FORCE signal) for details. The PWM signals according to the commutation techniques are shown in Figure 10, Figure 11, Figure 12, and Figure 13.

4.2.2 6-step BLDC commutation requires one phase off

The BLDC commutation specific is that one motor phase needs to be completely off at any given time. So both the top and bottom PWM signals need to be in an “off” state. This is provided by using the *PWM_MASK* register, which can individually mask any PWM signal even in complementary mode.

The BLDC switching is provided according to one of the six BLDC sectors. Therefore, a commutation table is necessary. The output masktable using the *PWM_MASK* register is the same for any 4-quadrant complementary BLDC commutation techniques described in this application note.

Example 9 Pseudo code – PWM_MASK according to sectors

```
Sector0: MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1 = 0x0220;/* phase B off */

Sector1: MCPWM.PWM_MASK = MASKA_SUB0|MASKB_SUB0 = 0x0110;/* phase A off */

Sector2: MCPWM.PWM_MASK = MASKA_SUB2|MASKB_SUB2 = 0x0440; /* phase C off */

Sector3: MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1 = 0x0220;/* phase B off */

Sector4: MCPWM.PWM_MASK = MASKA_SUB0|MASKB_SUB0 = 0x0110;/* phase A off */

Sector5: MCPWM.PWM_MASK = MASKA_SUB2|MASKB_SUB2 = 0x0440;/* phase C off */
```

Where *MASKA_SUBn* is the n-th submodule A signal mask. *MASKA_SUBn* is the n-th submodule B signal mask . For example, in sector 0:

```
MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1 = 0x0220;
```

Switches off the motor phase controlled by PWMA1, and PWMB1.

In sector 1:

```
MCPWM.PWM_MASK = MASKA_SUB0|MASKB_SUB0 = 0x0110;
```

Switches off the motor phase controlled by PWMA0 and PWMB0. The *PWM_MASK* register can also be used to switch off all the PWM outputs, for example in a BLDC motor control application stop state.

4.2.3 Complementary bipolar PWM 6-step BLDC commutation

For bipolar PWM commutation, two active phases are controlled with a positive and negative signal of the same PWM pattern as in Figure 10 and Figure 11. This is provided by using the SMnSEL23 and SMnSEL45 of the register PWM_DTSRCSEL. The register PWM_DTSRCSEL needs to be set according to a 6-step commutation sector. This sector determines the BLDC motor voltage vector.

Example 10 Pseudo code – SMnSEL23 according to sectors

```
Sector0:
MCPWM.PWM_DTSRCSEL = SM2SEL23_NEG_PWM23|SM1SEL23_PWM23|SM0SEL23_PWM23 = 0x0400
Sector1:
MCPWM.PWM_DTSRCSEL = SM2SEL_NEG_PWM23|SM1SEL23_PWM23|SM0SEL23_NEG_PWM23 = 0x0404;
Sector2:
MCPWM.PWM_DTSRCSEL = SM2SEL_PWM23|SM1SEL23_PWM23|SM0SEL23_NEG_PWM23 =0x0004;
Sector3:
MCPWM.PWM_DTSRCSEL = SM2SEL_PWM23|SM1SEL23_NEG_PWM23|SM0SEL23_NEG_PWM23 = 0x0044;
Sector4:
MCPWM.PWM_DTSRCSEL = SM2SEL_PWM23|SM1SEL23_NEG_PWM23|SM0SEL23_PWM23 = 0x0040;
Sector5:
MCPWM.PWM_DTSRCSEL = SM2SEL_NEG_PWM23|SM1SEL23_NEG_PWM23|SM0SEL23_PWM23 =0x0440;
```

Where *SMnSEL_NEG_PWM23* is the n-th submodule negative PWM23 selection. *SMnSEL_PWM23* is the n-th submodule generated PWM23 signal selection

For example, in sector 0:

```
MCPWM.PWM_DTSRCSEL = SM2SEL23_NEG_PWM23| SM1SEL23_PWM23|SM0SEL23_PWM23 = 0x0400;
```

This controls the submodule PWM0 with generated PWM signal and the submodule 2 PWM2 with inverted PWM signal. The PWM1 setting is not important because the output is masked with the *MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1*. The setting of the SMnSEL45 is not important in complementary mode when all IPOLn=0.

In sector 1:

```
MCPWM.PWM_DTSRCSEL = SM2_NEG_PWM23| SM1SEL23_PWM23|SM0SEL23_NEG_PWM23|  = 0x0404;
```

Controls the submodule PWM1 with the generated PWM signal, and the submodule 2 PWM2 with the inverted PWM signal. The PWM0 setting is not important because the output is masked with the *MCPWM.PWM_MASK = MASKA_SUB1*| *MASKB_SUB1*.

In order to have a 4-quadrant power stage operation, the PWM duty cycle needs to be calculated according to the following formula:

pwmDutyCycle = MODULO*(*outRegulator*/2+1/2)

where:

pwmDutyCycle – is the PWM duty cycle variable as described in other sections

outRegulator – is the regulator output value scaled to (-1,1)

MODULO – is the system value of the PWM modulo

So, in this way we get a coherent (seamless) 4-quadrant control because the medium motor applied voltage is 0 when *pwmDutyCycle* = MODULO*1/2.

4.2.4 Coherent complementary unipolar PWM 6-step BLDC commutation

This section describes a unipolar complementary 4-quadrant BLDC commutation with PWM signals according to Figure 12 and Figure 13. This technique advantage is a 4-quadrant coherent operation where the *pwmDutyCycle* of a negative value generates a negative voltage vector and so a coherent(seamless) 4-quadrant operation.

Before any BLDC commutation, the *PWM_DTSRCSEL* register needs to be initialized:

Example 11 Pseudo code – SMnSEL23, SMnSEL45 initialization

```
MCPWM.PWM_DTSRCSEL= SM2SEL23_PWM23 | |SM2SEL45_PWM45 | SM1SEL23_PWM23 | SM1SEL45_PWM45 |
SM0SEL23_PWM23 | SM0SEL45_PWM45= 0x0000;
```

Where *SMnSEL23_ PWM23* is the n-th submodule PWM23 selection of the PWM23 signal, *SMnSEL45_ PWM45* is the n-th submodule PWM45 selection of the PWM45 signal.

NOTE

Some devices such as the PXS20 and MCF5441x require another PWM_DTSRCSEL setting due to their older PWM versions

Example 12 Pseudo code – SMnSEL23, SMnSEL45 initialization

```
MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23 | SM1SEL23_PWM23 | SM0SEL23_PWM2 | SM2SEL45_NEG_PWM45 |
SM1SEL45_NEG_PWM45 | SM0SEL45_NEG_PWM45= 0x0111;
```

The DTSRSEL register needs to be initialized for positive PWM23 signals selection and negative PWM45 signals selection because the IPOL bit setting provides selection of the 2,3 or negative 4,5 PWM. Regardless of the IPOL bit setting, the final PWM is a positive PWM23 or positive PWM45 signal.

When we smartly set the PWM value registers *MCPWM.PWM_SM0VAL3 = pwmDutyCycle* and *MCPWM.PWM_SM0VAL5 = -MCPWM.PWM_SM0VAL3*, we get the benefit of a natural 4-quadrant operation. When the PWM value register *MCPWM.PWM_SM0VAL5* is a lower (negative) value than *MCPWM.PWM_SM0VAL4 = 0*, the PWM signal is of a low level.

Example 13 Pseudo code – IPOLmn according to sector

```
Sector 0: MCPWM.PWM_MCTRL = IPOL01=0x3000;
Sector 1: MCPWM.PWM_MCTRL = IPOL1 =0x2000;
Sector 2: MCPWM.PWM_MCTRL = IPOL21 =0x6000;
```

```
Sector 3: MCPWM.PWM_MCTRL = IPOL2 =0x4000;
Sector 4: MCPWM.PWM_MCTRL = IPOL20 =0x5000;
Sector 5: MCPWM.PWM_MCTRL = IPOL0 =0x1000;
```

Where, *IPOLmn* are IPOL m and n bits.

So for example, in sector 0:

```
MCPWM.PWM_MCTRL = SM0IPOL01= 0x3000;
MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1 = 0x0220;
```

Gives the following PWM module output. PWM0 is a PWM signal determined by *MCPWM.PWM_SM0VAL3* = *pwmDutyCycle*. The PWMA1 and PWMB1 are masked in and so it offs the phase. Finally, the PWM2 gives a low level of the complementary channels PWMA2 and PWMB2, so the PWM2B is high. This is because the *MCPWM.PWM_SM0VAL5* is a negative value.

And the regulator output *pwmDutyCycle* can be simply used for a coherent Ist other quadrant operation change. When the *pwmDutyCycle* is of a negative value the situation changes. PWM0 PWM2 gives a low level of the complementary channels PWMA2 and PWMB2. This is because the *MCPWM.PWM_SM0VAL4* is a negative value. The PWMA1 PWMB1 are masked in, and so it offs the phase. The PWM2 gives a PWM signal determined by *MCPWM.PWM_SM0VAL4* = – *pwmDutyCycle* > *0*. So finally, the voltage over the motor is the same vector, but a negative scalar value due to *pwmDutyCycle* <*0*.

In order to utilize the 4-quadrant operation, the PWM duty cycle needs to be calculated according to the following formula:

pwmDutyCycle = MODULO**outRegulator*

where:

pwmDutyCycle – is the PWM duty cycle variable as described in other sections

outRegulator – is the regulator output value scaled to (-1,1)

MODULO – is the system value of the PWM modulo

So, in this way we get a coherent (seamless) 4-quadrant control because the medium motor applied voltage is 0 when *pwmDutyCycle* = 0.

Warning:

The technique gives a benefit on all DSC56F84xx and newer devices.

Due to a bug on some devices the PWM module does not insert dead time when IPOL = 1.

Therefore:

- **dead time insertion must be provided with a h/w drive. Or**
- **a workaround with pwmDutyCycle >0 must be done and then the technique will not give a 4-quadrant BLDC operation. Or**
- **a Standard Complementary Unipolar PWM 6-step BLDC commutation can be used**

4.2.5 Standard complementary unipolar PWM 6-step BLDC commutation

A standard complementary unipolar PWM 6-step BLDC commutation gives the same PWM signals as the coherent version, but the *pwmDutyCycle* must be a positive value *pwmDutyCycle* >0. So the II quadrant can not be smoothly achieved using a pure duty cycle setting.

The commutation is then provided using *PWM_DTSRCSEL* together with an appropriate *PWM_SWCOUT* setting.

So, we initialize:

Example 14 Pseudo code – SMnOut23 initialization

```
MCPWM.PWM_SWCOUT = SMnOUT23| SMnOUT23=  0x0000;
```

And then the *PWM_DTSRCSEL*is set according to the required six-step sector:

Example 15 Pseudo code – SMnSEL according to sector

```
Sector0:MCPWM.PWM_DTSRCSEL=SM2SEL23_SWCOUT|SM1SEL23_PWM23|SM0SEL23_PWM23=0x0800;
Sector1:MCPWM.PWM_DTSRCSEL=SM2SEL23_SWCOUT|SM1SEL23_PWM23|SM0SEL23_SWCOUT=0x0808;
Sector2:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL23_PWM23|SM0SEL23_SWCOUT=0x0008;
Sector3:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL23_SWCOUT|SM0SEL23_SWCOUT=0x0088;
Sector4:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL23_SWCOUT|SM0SEL23_PWM23=0x0080;
Sector5:MCPWM.PWM_DTSRCSEL=SM2SEL23_SWCOUT|SM1SEL23_SWCOUT|SM0SEL23_PWM23=0x0880;
```

Where *SMnSEL23_ SWCOUT* is n-th submodule PWM23 selection of the SWCOUT[SM3OUT23] signal.

For example, in sector 0:

```
MCPWM.PWM_DTSRCSEL = SM2SEL23_SWCOUT| SM1SEL23_PWM23| SM0SEL23_PWM23= 0x0800;
MCPWM.PWM_MASK = MASKA_SUB1|MASKB_SUB1 = 0x0220;
```

Provides PWM0 control by the PWM23 signal. The PWM1 off motor phase with the top and bottom outputs PWMA1, PWMB1 at a low level. And PWM2 is a complementary level 0 due to SM2SEL_SWCOUT and because the SW control register is 0.

In sector 1:

```
MCPWM.PWM_DTSRCSEL = SM2SEL23_SWCOUT| SM1SEL23_PWM23 = 0x0800;
MCPWM.PWM_MASK = MASKA_SUB0|MASKB_SUB0 = 0x0110;
```

Provides PWM1 control by the PWM23 signal. The PWM0 off motor phase, with the top and bottom outputs PWMA1, PWMB1 at a low level. And PWM2 is complementary level 0 due to SM2SEL_SWCOUT and because the SW control register is 0.

This technique is simpler but does not enable a coherent Ist to IInd (or IIIrd to IVth) quadrant operation change with the *pwmDutyCycle* going from a positive to negative value. When the *pwmDutyCycle* is a negative value, the motor voltage is 0. The opposite motor voltage vector must be provided by applying a negative sector, that is, sector = sector + 3.

Therefore:

In order to have a 4-quadrant operation, the PWM duty cycle needs to be calculated according to the following formulae: *pwmDutyCycle* = abs(MODULO*outRegulator);

where:

pwmDutyCycle – is the PWM duty cycle variable as described in other sections

outRegulator – is regulator output value scaled to (–1,1)

MODULO – is the system value of the PWM modulo

When *outRegulator <0*, the negative sector needs to be applied.

4.2.6 2-quadrant (independent) unipolar PWM 6-step BLDC commutation

Some applications might require operations in quadrant I or I and III only. This means that a generating control mode is not used and the regulator operation is not linear when its output *pwmDutyCycle* < 0. In this region, the applied motor voltage is not exactly defined. The PWM control signals are then different. The main difference is that the top and bottom transistors are not switched in complementary mode.

The simpliest way to do 2-quadrant commutation is to keep the complementary mode setting and to mask the complementary top or bottom register using the *MCPWM.PWM_MASK* register according to the sector.

Another solution might be to use the independent PWM mode with the *INDEP* bit of the *PWM_SM0CTRL2[INDEP] = 1*. The BLDC commutation using the independent mode is simple, however this mode is not as safe as the complementary mode, because the top and bottom transistors can be switched at the same time.

The unipolar commutation can be provided with utilization of the *PWM_DTSRCSEL* and *MCPWM.PWM_MASK* register. The *PWM_SWCOUT* register needs to be set to provide a high-level signal for the bottom transistors.

Example 16 Pseudo code – SWCOUT and SMnCTRL2 initialization

```
MCPWM.PWM_SWCOUT = 0x00FF;
/* set INDEP */
PWM_SMnCTRL2 |=0x2000;
```

The *PWM_DTSRCSEL* is then controlled according to the commutation sector using a table:

Example 17 Pseudo code – SMnSEL according to sector

```
Sector0:MCPWM.PWM_DTSRCSEL=SM2SEL45_SWCOUT|SM1SEL23_PWM23|SM0SEL23_PWM23=0x0200;
Sector1:MCPWM.PWM_DTSRCSEL=SM2SEL45_SWCOUT|SM1SEL23_PWM23|SM0SEL45_SWCOUT=0x0202;
Sector2:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL23_PWM23|SM0SEL45_SWCOUT=0x0002;
Sector3:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL45_SWCOUT|SM0SEL45_SWCOUT=0x0022;
Sector4:MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL45_SWCOUT|SM0SEL23_PWM23=0x0020;
Sector5:MCPWM.PWM_DTSRCSEL=SM2SEL45_SWCOUT|SM1SEL45_SWCOUT|SM0SEL23_PWM23=0x0220;
```

For example, in sector 0:

```
MCPWM.PWM_DTSRCSEL = SM2SEL45_SWCOUT  |SM1SEL23_PWM23|SM0SEL23_PWM23 = 0x0200;
MCPWM.PWM_MASK = MASKA_SUB2|MASKA_SUB1|MASKB_SUB1| MASKB_SUB0 = 0x0220;
```

The output signals are PWMA0 (top transistor) with PWM23, and PWMB2 (bottom transistor) at a logical high level. The submodule 1 is off with PWMA1 and PWMB1 at a low level.

Then, in sector 1:

```
MCPWM.PWM_DTSRCSEL = SM2SEL45_SWCOUT |SM1SEL23_PWM23|SM0SEL45_SWCOUT = 0x0202;
MCPWM.PWM_MASK = MASKA_SUB2|MASKB_SUB1|MASKA_SUB0|MASKB_SUB0 = 0x0110;
```

The output signals are PWMA1 (top transistor) with PWM23, and PWMB2 (bottom transistor) at a logical high level. The submodule 0 is off, with PWMA1 and PWMB1 at a low level.

4.2.7 2-quadrant (independent) bipolar PWM 6-step BLDC commutation

Bipolar PWM control in independent PWM mode commutation can be provided with utilization of the *PWM_DTSRCSEL* and *MCPWM.PWM_MASK* registers similar to bipolar mode. In the independent bipolar PWM mode, it is also necessary to update the duty cycle on both the *MCPWM.PWM_SMnVAL3* and *MCPWM.PWM_SMnVAL5* with the *pwmDutyCycle* value.

Example 18 Pseudo code –SMnCTRL2 initialization

```
/* set INDEP */
PWM_SMnCTRL2 |=0x2000;
```

The *PWM_DTSRCSEL* is then controlled according to the commutation sector using a table:

Example 19 Pseudo code – SMnSEL according to sector

```
Sector0:
MCPWM.PWM_DTSRCSEL=SM2SEL45_NEG_PWM45|SM1SEL23_PWM23|SM0SEL23_PWM23=0x0100;
Sector1:
MCPWM.PWM_DTSRCSEL=SM2SEL45_NEG_PWM45|SM1SEL23_PWM23|SM0SEL45_NEG_PWM45=0x0101;
Sector2:
MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL23_PWM23|SM0SEL45_NEG_PWM45=0x0001;
Sector3:
MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL45__NEG_PWM45|SM0SEL45_NEG_PWM45=0x0011;
```

```
Sector4:
MCPWM.PWM_DTSRCSEL=SM2SEL23_PWM23|SM1SEL45_NEG_PWM45|SM0SEL23_PWM23=0x0010;
Sector5:
MCPWM.PWM_DTSRCSEL=SM2SEL45_NEG_PWM45|SM1SEL45_NEG_PWM45|SM0SEL23_PWM23=0x0110;
```

And, for example, in sector 0:

```
MCPWM.PWM_DTSRCSEL = SM2SEL45_NEG_PWM45 |SM1SEL_PWM23|SM0SEL23_PWM23 = 0x0100;
MCPWM.PWM_MASK = MASKA_SUB2|MASKA_SUB1|MASKB_SUB1| MASKB_SUB0 = 0x0220;
```

The output signals are PWMA0 (top transistor) with PWM23, and PWMB2 (bottom transistor) at a logical high level. The submodule 1 is off, with PWMA1 and PWMB1 of low level.

Then, in sector 1:

```
MCPWM.PWM_DTSRCSEL=SM2SEL45_NEG_PWM45|SM1SEL_PWM23|SM0SEL45_NEG_PWM45=0x0100;
MCPWM.PWM_MASK = MASKA_SUB2|MASKB_SUB1|MASKA_SUB0|MASKB_SUB0 = 0x0110;
```

The output signals are PWMA1(top transistor) with PWM23, and PWMB2 (bottom transistor) at a logical high level. The submodule 0 is off, with PWMA1 and PWMB1 of at a level.

4.2.8 PWM trigger for s/w commutation

The commutation trigger can be a h/w or the manually generated signal FORCE_OUT. The registers *PWM_DTSRCSEL*, *PWM_MASK*, *PWM_SWCOUT*, and *MCTRL[IPOL]* are double buffered. The trigger FORCE_OUT provides a loading of those buffered registers. Before each commutation, the registers need to be preloaded according to the required sector and commutation technique.

In the case the commutation is triggered manually, the submodule 0 FORCE_SEL = 000. The local force signal, *PWM_SM0CTRL2[FORCE]*, from submodule 0 is used to force the updates. The submodules 1 and 2 are then set to FORCE OUTPUT FORCE_SEL = 001. The master force signal from submodule 0 is used to force the updates. See Example 20 below.

Example 20 Pseudo code – FORCE OUT trigger setting

```
/* FORCE_SEL = 000 The local force signal, CTRL2[FORCE], from this submodule is used to
force the updates  */
MCPWM.PWM_SM0CTRL2 = 0x0000;

/* submodules n = 1,2 CTRL2 Setting*/
/* FORCE_SEL = 001 The master force signal from submodule 0 is used to force the updates. */
/* MCPWM.PWM_SMnCTRL2 &= 0xFFC7; */
MCPWM.PWM_SMnCTRL2 |= 0x0008;
```

The commutation is then triggered with:

```
/* FORCE  = 1 writing a 1 to this bit results in a FORCE_OUT event */
MCPWM.PWM_SM0MCTRL2 |= 0x0040;
```

4.2.9 PWM trigger for h/w commutation (EXT_FORCE signal)

For a h/w commutation trigger, the FORCE_OUT needs to be generated using the EXT_FORCE signal. The EXT_FORCE signal is connected from the Crossbar Switch (XBAR) module of the MC56F8247, the Crossbar Switch module is also available on almost any devices described in this application note, and is usually connected to a timer which controls the BLDC commutation timing.

In the case where the commutation is triggered with a h/w EXT_FORCE, the submodule 0 FORCE OUTPUT FORCE_SEL = 110. The EXT_FORCE, from submodule 0, is used to force the updates. The submodules 1 and 2 are then set to FORCE OUTPUT FORCE_SEL = 001. The master force signal from submodule 0 is used to force the updates. See Example 21 below.

Example 21 Pseudo code – FORCE OUT trigger setting

```
/* FORCE_SEL = 110 The external force signal, EXT_FORCE, from outside the PWM module causes
the updates  */
MCPWM.PWM_SM0CTRL2 = 0x0030;

/* submodules n = 1,2 CTRL2 Setting*/
/* FORCE_SEL = 001 The master force signal from submodule 0 is used to force the updates. */
/* MCPWM.PWM_SMnCTRL2 &= 0xFFC7; */
MCPWM.PWM_SMnMCTRL2 |= 0x0008;
```

The commutation is then triggered with the EXT_FORCE signal from the Crossbar Switch (XBAR) module, usually connected to a timer.

4.2.10 Use in 3-phase BLDC motor control application

This section describes a typical example of unipolar 6-step BLDC commutation with 4-quadrant operation according to the Coherent complementary unipolar PWM 6-step BLDC commutation on the MC56F8247 device.

4.2.11 PWM initialization example

The code for PWM initialization is divided into two parts. The first part of the code needs to repeat for the submodules *n* = 0,1,2. Some settings are different for submodule 0 and submodules 1 and 2

Example 22 Pseudo code – PWM Initialization for submodules *n* = 0,1,2

```
/* PWM Duty cycles init - usually SMnVAL2_INIT = 0 */
MCPWM.PWM_SMnVAL2 = 0;
MCPWM.PWM_SMnVAL3 = SMnVAL3_INIT;
MCPWM.PWM_SMnVAL4 = 0;
MCPWM.PWM_SMnVAL5 = SMnVAL5_INIT;

/*** PWM Modulo - Frequency Setting ***/
MCPWM.PWM_SM0INIT = 0;
MCPWM.PWM_SM0VAL1 = MODULO = 0x0FA0;
/* DEFAULT: FULL = 1 full cycle reload enabled */
MCPWM.PWM_SMnCTRL = 0x0400;

/*** Submodule0 CTRL2 settings ***/
/* FORCE_SEL = 000 The local force signal, CTRL2[FORCE], from this submodule is used to
force the updates
CLK_SEL = 00 The IPBus clock is used as the clock for the local prescaler and counter.
RELOAD_SEL = 0 The local RELOAD signal is used to reload registers */
MCPWM.PWM_SM0CTRL2 = 0x0000;

/*** Submodules n = 1,2 CTRL2 settings ***/
/* FORCE_SEL = 001 The master force signal from submodule 0 is used to force updates.
CLK_SEL = 10  Submodule 0's clock (AUX_CLK) is used as the source clock for the local
prescaler and counter.
RELOAD_SEL = 001 The master force signal from submodule 0 is used to force updates. */
MCPWM.PWM_SMnMCTRL2 = 0x000E;

/* OUT_TRIG0 will set when the counter value matches the VAL0 value. */
MCPWM.PWM_SM0TCTRL =  0x0001;
/* value for trigger OUT_TRIG0 synchronization signal */
MCPWM.PWM_SM0VAL0 = 0;

/*** Driver Output Polarity and dead time settings ***/
/* POLA = 1 output inverted PWMAFS = 0,  PWMBFS = 0, PWMXFS = 0Fault state output is forced
to logic 0 state prior to consideration of output polarity control */
MCPWM.PWM_SMnOCTRL = 0x0400;
```

```
/*** Fault disable map setting ***/
/* disable DISXn, DISBn, DISAn outputs at FAULTx defined in the s/w constant
PWM_0_DISMAP_INIT */
MCPWM. PWM_SMnDISMAP = PWM_0_DISMAP_INIT;

/*** Dead time duration settings ***/
/* dead time 1 IPBus clock cycles (regardless of the setting of CTRL[PRSC]) */
MCPWM.PWM_SMnDTCNT0 = DEADTIME0_INIT =  0x00F0;
/* dead time 1 IPBus clock cycles (regardless of the setting of CTRL[PRSC]) */
MCPWM.PWM_SMnDTCNT1 = DEADTIME1_INIT =  0x00F0;
```

The second part of the initialization code common to all the PWM submodules starts from here

Example 23 Pseudo code – PWM Common Initialization for all submodules

```
/* manual output control of all pwm submodules preset to 0 */
MCPWM.PWM_SWCOUT = 0x0000;

/* SM0SEL23 = Generated SM0PWM23, SM0SEL45 = Generated SM0PWM45
    SM1SEL23 = Generated SM1PWM23, SM1SEL45 = Generated SM3PWM45
    SM2SEL23 = Generated SM1PWM23, SM2SEL45 = Generated SM2PWM45 */
MCPWM.PWM_DTSRCSEL = 0x0000;

/* IPOL, RUN, CLDOK = 0, LDOK0,1,2 = 1 This read/set loads CTRL[PRSC] and the INIT,
FRACVALx, and VALx */
Dummy = MCPWM.PWM_MCTRL;
MCPWM.PWM_MCTRL = 0x0007;

/* DEFAULT 0 */
/* MCPWM.PWM_MCTRL2 = 0x0000; */

/*** Fault setting ***/
/* FLVL = 1 fault level is 1, FAUTO = 0 manual fault clearing, FSAFE = 0 normal mode, FIE =
fault interrupt enable!!!*/
MCPWM.PWM_FCTRL = 0xF000;

/* fault filtration setting FILT_CNT , FILT_PER */
MCPWM.PWM_FFILT = PWM_FFILT_INIT;

/*** Inverter mask - motor off setting ***/
/* MASKA0,1,2 = 1, MASKB0,1,2 = 1=> inverter off */
MCPWM.PWM_MASK = 0x0770;

/* BLDC sector internal s/w variable initialization */
unsigned short Sector/* BLDC sector pointer 16 bit unsigned <0,5> for 6-step operation */;
sector = FIRST_SECTOR_INIT;

/* PWM clock and output enable */
/* PWM clock enable */
MCPWM.PWM_MCTRL |= MCTRL_RUN210; /* MCTRL_RUN210 =  0x0700 */

/* PWM Output Enable */
MCPWM.PWM_OUTEN |= (OUTEN_PWMA_EN_SUB0|
OUTEN_PWMB_EN_SUB0|
OUTEN_PWMA_EN_SUB1\
OUTEN_PWMB_EN_SUB1\
OUTEN_PWMA_EN_SUB2\
OUTEN_PWMB_EN_SUB2; /* = 0x0770 */
```

4.2.12 BLDC commutation example

After the PWM module is initialized, the BLDC motor commutation can be controlled according to the *sector* variable. The following code uses s/w commutation, using *PWM_SM0CTRL2[FORCE]* = 1 as defined by the initialization *CTRL2[FORCE_SEL] = 000*.

Example 24 Pseudo code – BLDC Commutation

```
#define MCTRL_IPOL_SUB2                0x4000
#define MCTRL_IPOL_SUB1                0x2000
#define MCTRL_IPOL_SUB0                0x1000

const UWord16 6s_ss_IPOL23_table [] = {
(MCTRL_IPOL_SUB0 | MCTRL_IPOL_SUB1),
(MCTRL_IPOL_SUB1),
 (MCTRL_IPOL_SUB1| MCTRL_IPOL_SUB2),
 (MCTRL_IPOL_SUB2),
(MCTRL_IPOL_SUB0 | MCTRL_IPOL_SUB2),
 (MCTRL_IPOL_SUB0) };

const UWord16  6s_ss_IPOL45_table [] = {
 (MCTRL_IPOL_SUB2),
 (MCTRL_IPOL_SUB0| MCTRL_IPOL_SUB2),
 (MCTRL_IPOL_SUB0),
 (MCTRL_IPOL_SUB0| MCTRL_IPOL_SUB1),
 (MCTRL_IPOL_SUB1),
 (MCTRL_IPOL_SUB1| MCTRL_IPOL_SUB2) };

#define MASK_MASKA_MASK0x0F00
#define MASK_MASKA_SUB30x0800
#define MASK_MASKA_SUB20x0400
#define MASK_MASKA_SUB10x0200
#define MASK_MASKA_SUB00x0100
#define MASK_MASKB_MASK0x00F0
#define MASK_MASKB_SUB30x0080
#define MASK_MASKB_SUB20x0040
#define MASK_MASKB_SUB10x0020
#define MASK_MASKA_SUB00x0010

const UWord16   6s_ss_MASK_table [] = {
 (MASK_MASKA_SUB1|MASK_MASKB_SUB1),
 (MASK_MASKA_SUB0| MASK_MASKB_SUB0),
 (MASK_MASKA_SUB2| MASK_MASKB_SUB2),
 (MASK_MASKA_SUB1| MASK_MASKB_SUB1),
 (MASK_MASKA_SUB0| MASK_MASKB_SUB0),
 (MASK_MASKA_SUB2| MASK_MASKB_SUB2) };

/* IPOLn = 0 - PWM23  or 1 - PWM45 according to sector n = 0,1,2 */
/* clear  IPOLn=0 - PWM23  according to 6s_ss_IPOL23_table[sector] n = 0,1,2  */
MCPWM.PWM_MCTRL &= ~( MCTRL_IPOL_MASK & 6s_ss_IPOL23_table[sector]);
/* set  IPOLn=1 - PWM45  according to 6s_ss_IPOL23_table[sector] n = 0,1,2  */
MCPWM.PWM_MCTRL |= (MCTRL_IPOL_MASK & 6s_ss_IPOL45_table[sector]);

/* MASKAn = 0 PWA output normal, 1 PWA output masked, MASKn, MASKXn */
MCPWM.PWM_MASK = 6s_ss_MASK_table[sector];

/* for s/w manually triggered commutation, submodule 0 FORCE = 1  */
PWM_SM0CTRL2 |= 0x0040;
```

4.2.13 BLDC duty cycle setting example

The PWM duty cycle setting for a 4-quadrant unipolar switching with edge alignment is then provided according to the *pwmDutyCycle* variable, see Coherent complementary unipolar PWM 6-step BLDC commutation for details.

Example 25 Pseudo code – PWM Duty Cycle Setting

```
short pwmDutyCycle;/* signed 16 bit  <- MODULO, + MODULO> */
/* MCPWM.PWM_SM0VAL2 = 0; */
MCPWM.PWM_SM0VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM0VAL4 = 0; */
MCPWM.PWM_SM0VAL5 = - pwmDutyCycle;

/* MCPWM.PWM_SM1VAL2 = 0; */
```

```
MCPWM.PWM_SM1VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM1VAL4 = 0; */
MCPWM.PWM_SM1VAL5 = - pwmDutyCycle;

/* MCPWM.PWM_SM2VAL2 = 0; */
MCPWM.PWM_SM2VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM2VAL4 = 0; */
MCPWM.PWM_SM2VAL5 = - pwmDutyCycle;

/* OUT_TRIG0 timing */
MCPWM.PWM_SM0VA0 = abs(pwmDutyCycle)/2;
/* LDOK0,1,2 = 1 This read/set loads VALx */
MCPWM.PWM_MCTRL |= 0x0007;
```

Warning:

The technique gives a benefit on all DSC56F84xx and newer devices.

Due to a bug on some devices, the PWM module does not insert dead time when IPOL = 1.

Therefore:

- **dead time insertion must be provided with a h/w driver**
- **or a workaround needs to be done and then the technique will not give a 4-quadrant BLDC operation. The workaround is shown below.**

The PWM duty cycle setting is then provided according to the *pwmDutyCycle* variable.

Example 26 Pseudo code – PWM Duty Cycle Setting on some devices

```
unsigned short pwmDutyCycle;/* unsigned 16 bit  <0, + MODULO> */
/* MCPWM.PWM_SM0VAL2 = 0; */
MCPWM.PWM_SM0VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM0VAL4 = 0; */
MCPWM.PWM_SM0VAL5 = 0;

/* MCPWM.PWM_SM1VAL2 = 0; */
MCPWM.PWM_SM1VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM1VAL4 = 0; */
MCPWM.PWM_SM1VAL5 = 0;

/* MCPWM.PWM_SM2VAL2 = 0; */
MCPWM.PWM_SM2VAL3 = pwmDutyCycle;
/* MCPWM.PWM_SM2VAL4 = 0; */
MCPWM.PWM_SM2VAL5 = 0;

MCPWM.PWM_SM0VA0 = abs(pwmDutyCycle)/2;
/* LDOK0,1,2 = 1 This read/set loads VALx */
MCPWM.PWM_MCTRL |= 0x0007;
```

Warning:

The PWM_DTSRCSEL = 0 on all DSC56F84xx because the PWM45 signals are not inverted. But on some devices the PWM45 signals are inverted, so the following setting is necessary for appropriate functionality when IPOL = 1.

Therefore, SMnSEL45 needs to be reinverted:

```
/* SM0SEL23 = Generated SM0PWM23, SM0SEL45 = Inverted SM0PWM45
    SM1SEL23 = Generated SM1PWM23, SM1SEL45 = Inverted SM3PWM45
    SM2SEL23 = Generated SM1PWM23, SM2SEL45 = Inverted SM2PWM45
   Inverted PWM45 signals with noninverted PWM23 are necessary when using IPOL control for a
correct functionality */
MCPWM.PWM_DTSRCSEL = 0000 0001 0001 0001;
```

5 References

1. MC56F825x/4x Reference Manual - MC56F825XRM Rev.2, Freescale Semiconductor 10/2010, available at http://www.freescale.com
2. 3-Phase Sensorless BLDC Motor Control Using MC9S08MP16 - Design Reference Manual, DRM117, Freescale Semiconductor 2009, available at http://www.freescale.com
3. PXS20 Microcontroller Reference Manual - PXS20RM Rev. 1, Freescale Semiconductor 06/2011, available at http://www.freescale.com
4. Migrate PWM from MC56F8013 to MC568247 – AN4319, Freescale Semiconductor 06/2011, available at http://www.freescale.com

6 Conclusion

This application note describes a typical use of the eFlexPWM module and its derivatives in BLDC motor control applications. The application note was based on the MC56F8247 device.

However, the eFlexPWM module is a complex and flexible peripheral and there are many current and future Freescale devices utilizing these or basically the same, PWM modules. Also there are many other solutions for the BLDC PWM control signals. Most of the techniques can be implemented on the described PWM. But, it was not possible to describe all the devices and control techniques. The user should consult the manuals to utilize other functionalities.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4429
Rev. 0, 12/2011