

DSP56F80x MC PWM Module in Motor Control Applications

Use of the DSP56F80x MC PWM Module in
Motor Control Applications and Examples of
Settings for Different Motor Types

Leos Chalupa

1. Introduction

This Application Note describes the usage of the DSP56F80x on-chip Motor Control PWM (Pulse Width Modulation) Module in different 3-phase applications (AC Induction motor, PM Synchronous motor, BLDC motor, SR motor etc.) and DC Brushed motor. The following sections show examples of the typical PWM waveforms, control signals, on-the-fly settings, service code steps and other settings needed to operate the MC PWM module.

2. Motorola DSP Advantages and Key Features

The Motorola DSP56F80x family is well suited for digital motor control, combining the DSP's calculation capability with MCU's controller features on a single chip. These DSPs offer many dedicated peripherals, such as Pulse Width Modulation (PWM) unit, Analog-to-Digital Converter (ADC), Timers, communication peripherals (SCI, SPI, CAN), on-board Flash and RAM. Several members of the family exist, including the DSP56F801, DSP56F803, DSP56F805 and DSP56F807, each with different peripheral sets and on-board memory configurations.

Contents

1.	Introduction	1
2.	Motorola DSP Advantage and Key Features	1
3.	Simple PWM Theory 3	
3.1	PWM Technique.....	3
3.2	Quadrants Operation.....	5
3.3	Bipolar and Unipolar Switching Modes	6
3.3.1	Bipolar Switching.....	6
3.3.2	Unipolar Switching	7
3.4	Center and Edge Aligned PWM	8
4.	MC PWM Module.....	11
4.1	Description.....	11
4.1.1	Prescaler	11
4.1.2	PWM Generator	12
4.1.3	Channel Mask and Swap	12
4.1.4	MUX and Current Sense	13
4.1.5	Deadtime Insertion - Top/Bottom generation	17
4.1.6	Fault Protection	17
4.1.7	Fault Pin Filters	18
4.1.8	Polarity Control.....	19
4.1.9	Interrupt Control.....	19
5.	Motor Control PWM Generation	20
5.1	AC Ind. Motor and PM Synchronous Motor.....	20
5.1.1	3-ph PWM Static Setting	21
5.1.2	3-ph Waveforms Generation	23
5.1.3	Cycle-by-cycle Modification of PWM Waveform	24
5.2	BLDC Motor.....	26
5.2.1	PWM for Classical BLDC Motor Control.....	27
5.2.2	PWM for Back-EMF Detection BLDC Motor Control.....	33
5.3	SR Motor	39
5.3.1	PWM for Classical SRM.....	40
5.4	DC Brushed Motor	52
5.4.1	PWM for DC Brushed Motor... ..	52
6.	Conclusion	58
7.	References	58

Preliminary Copy

DSP56F80x MC PWM Module in
Motor Control Applications

A typical member of the family, the DSP56F805, provides the following peripheral blocks:

- Two Pulse Width Modulator modules (PWMA & PWMB), each with six PWM outputs, three Current Sense inputs, and four Fault inputs, fault tolerant design with deadtime insertion, supporting both Center- and Edge- aligned modes
- Twelve bit, Analog to Digital Convertors (ADCs), supporting two simultaneous conversions with dual 4-pin multiplexed inputs, ADC can be synchronized by the PWM modules
- Two Quadrature Decoders (Quad Dec0 & Quad Dec1), each with four inputs, or two additional Quad Timers A & B
- Two dedicated General Purpose Quad Timers totalling 6 pins: Timer C with 2 pins and Timer D with 4 pins
- CAN 2.0 A/B Module with 2-pin ports used to transmit and receive
- Two Serial Communication Interfaces (SCI0 & SCI1), each with two pins, or four additional GPIO lines
- Serial Peripheral Interface (SPI), with configurable 4-pin port, or four additional GPIO lines
- Computer Operating Properly (COP) Watchdog timer
- Two dedicated external interrupt pins
- Fourteen dedicated General Purpose I/O (GPIO) pins, 18 multiplexed GPIO pins
- External reset pin for hardware reset
- JTAG/On-Chip Emulation (OnCE)
- Software-programmable, Phase Lock Loop-based frequency synthesizer for the DSP core clock

Table 2-1. Memory Configuration

	DSP56F801	DSP56F803	DSP56F805	DSP56F807
Program Flash	8188 x 16-bit	32252 x 16-bit	32252 x 16-bit	61436 x 16-bit
Data Flash	2K x 16-bit	4K x 16-bit	4K x 16-bit	8K x 16-bit
Program RAM	1K x 16-bit	512 x 16-bit	512 x 16-bit	2K x 16-bit
Data RAM	1K x 16-bit	2K x 16-bit	2K x 16-bit	4K x 16-bit
Boot Flash	2K x 16-bit	2K x 16-bit	2K x 16-bit	2K x 16-bit

The flexible PWM module, 16-bit Quadrature Timer module and ADC modules are especially helpful in motor control. The PWM's flexible configuration enhances motor control efficiency.

The PWM module has the following features:

- Three complementary PWM signal pairs, or six independent PWM signals
- Features of complementary channel operation
- Deadtime insertion
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or center-aligned PWM signals
- 15 bits of resolution

- Half-cycle reload capability
- Integral reload rates from one to 16
- Individual software-controlled PWM output
- Programmable fault protection
- Polarity control
- 20mA current sink capability on PWM pins
- Write-protectable registers

The PWM module can provide six-step BLDC commutation control, where one motor phase is left unpowered and the PWM controls the SR motor phase currents as well as 3-phase sinewave for AC Induction motors.

3. Simple PWM Theory

3.1 PWM Technique

PWM technique can be studied using the circuitry shown in [Figure 3-1](#). This circuitry is an example of the most commonly used power stage topology in the motor control field.

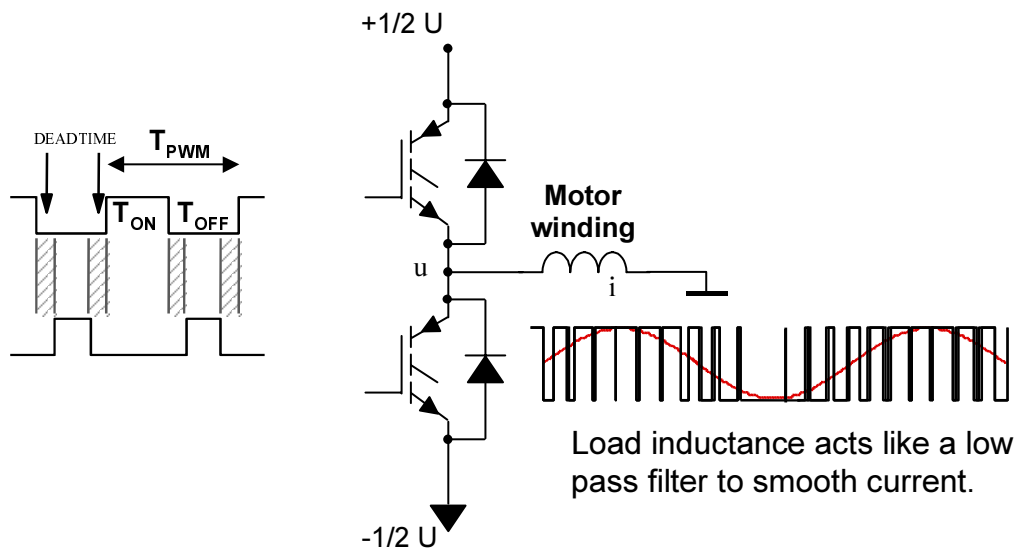


Figure 3-1. Half Bridge Power Stage

The top and bottom transistors (power switches) are controlled by a complementary pair of PWM signals with inserted deadtime. The deadtime assures that there is enough relaxation time to properly switch the transistor OFF before the complementary one is switched ON.

Notes: Never turn both transistors ON at the same time.

To simplify the further PWM technique explanation, the deadtime is not taken into an account. This is still a good approximation since the deadtime is usually small fraction of the PWM cycle.

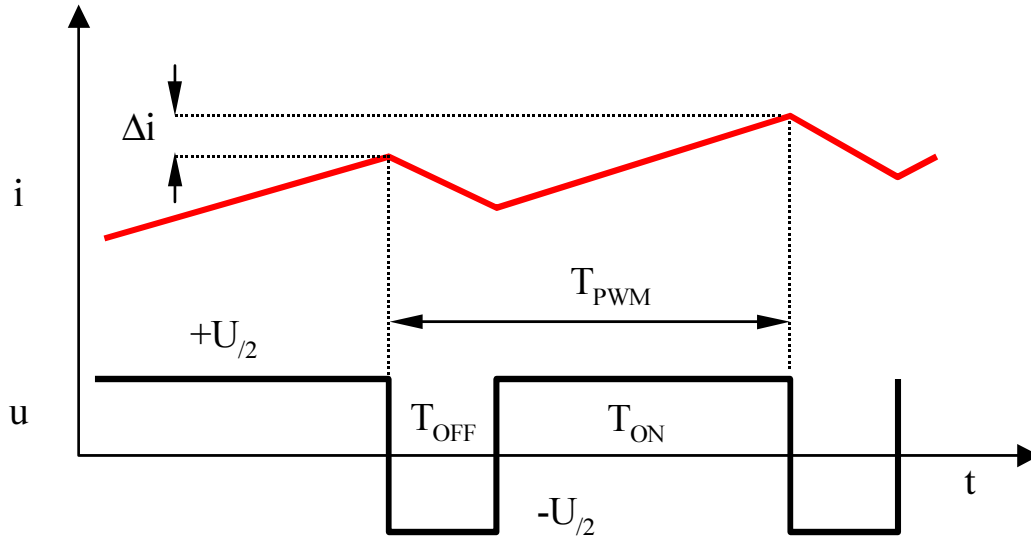


Figure 3-2. Applied PWM

When controlling transistors by PWM signal, the inductive load voltage and current waveforms can be as shown in [Figure 3-2](#). The following equations describe this case.

$$u = L \cdot \frac{di}{dt} \Rightarrow di = \frac{u}{L} \cdot dt \quad (\text{EQ 3-1})$$

$$T_{ON} = T_{PWM} \cdot d_c = T_{PWM} - T_{OFF} \quad (\text{EQ 3-2})$$

$$\Delta i = \frac{1}{2} \cdot \frac{U}{L} \cdot T_{ON} - \frac{1}{2} \cdot \frac{U}{L} \cdot T_{OFF} = \frac{1}{2} \cdot \frac{U}{L} (2T_{ON} - T_{PWM}) = \frac{T_{PWM}}{L} \cdot U \cdot \left(d_c - \frac{1}{2} \right) \quad (\text{EQ 3-3})$$

$$\Delta i = \frac{u^*}{L} \cdot T_{PWM} \quad (\text{EQ 3-4})$$

Where:

- u - is actual voltage applied to the inductive load
- u^* - is “mean” voltage applied to the inductive load during one PWM period
- i - is load current
- T_{PWM} - is PWM period
- T_{ON} - is PWM ON time, applied voltage is positive
- T_{OFF} - is PWM OFF time, applied voltage is negative
- d_c - is PWM duty cycle
- L - is the inductance of the load

by comparing (EQ 3-1) and (EQ 3-4) it can be seen that the PWM technique has the same impact as applying constant (“mean”) voltage during the PWM period. The PWM technique can be used to obtain the same load current (PWM current ripples are neglected) as if the variable voltage is applied to the inductive load. The PWM period (T_{PWM}) must be smaller than the voltage “changes” in order to allow the “mean” voltage to track the shape of the original voltage.

Let us modulate the PWM duty cycle (d_c) every PWM period using the function F_{PWM} as follows.

$$d_c = \frac{1}{2} \cdot F_{PWM} + \frac{1}{2} \text{ where: } -1 \leq F_{PWM} \leq 1 \quad (\text{EQ 3-5})$$

Then the load voltage u^* will be:

$$u^* = \frac{U}{2} \cdot F_{PWM} \quad (\text{EQ 3-6})$$

In a case where the F_{PWM} is a sinewave then the load voltage will be a sinewave, too. Thus the circuitry in [Figure 3-1](#) controlled by PWM technique provides a voltage source with variable voltage and frequency.

The PWM principle explanation is based on ideal inductance. In reality the winding resistance can be neglected only if the PWM period is significantly smaller than the electrical time constant of the RL load.

3.2 Quadrants Operation

Generally, any motor can be operated in motoring and generating mode of operation at forward and reverse speed. [Figure 3-3](#) shows the defined quadrants, the mode of motor and power stage operation. In order to control the motor in any quadrant the power stage, topology and PWM control technique must allow independent control of the voltage and current polarity.

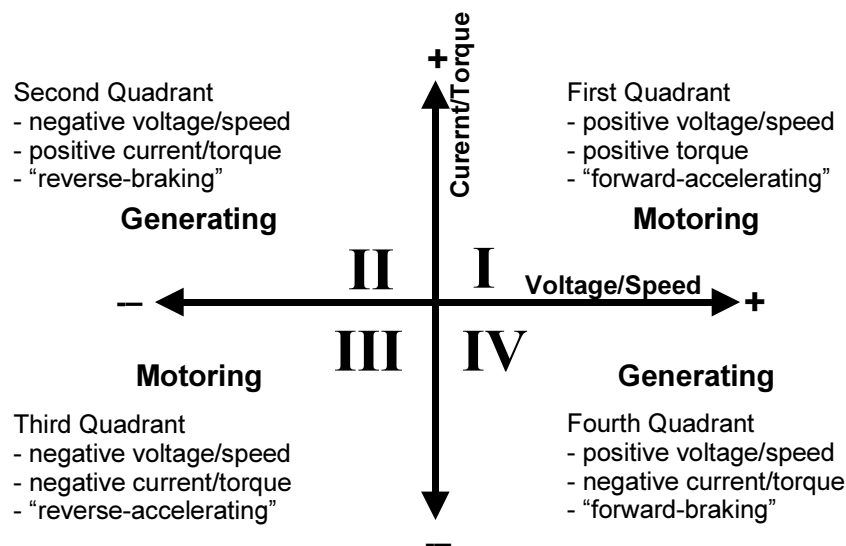


Figure 3-3. Quadrants of Operation

It must be noted that some motor control applications require limiting operation to just one or two quadrants. For example, a compressor works in the unidirectional motoring mode - Quadrant I; even if it is possible to start it in the opposite direction it works in motoring mode - Quadrant III. On the other side, a lift works in quadrant I (up) and II (down) when load is unchanged.

3.3 Bipolar and Unipolar Switching Modes

Generating both polarities of the load voltage is key for some applications as explained in the previous section. The power stage half-bridge shown in [Figure 3-1](#) can provide both polarities of the "mean" load voltage; however the actual voltage can only be positive or negative polarity - producing bipolar

PWM switching. The difference between bipolar and unipolar PWM switching modes can be better demonstrated using the full power stage bridge (see **Figure 3-4**). SW1-4 represent the power switches (e.g. IGBT transistors)..

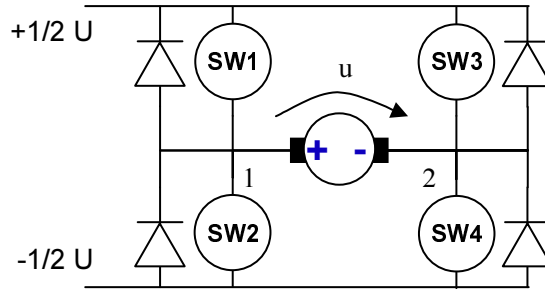


Figure 3-4. Power Stage Bridge

3.3.1 Bipolar Switching

The name of this PWM technique is derived from the fact that the actual voltage seen by the motor during one PWM period is +/-U (bipolar). The PWM control signals for the power switches SW1-4 allowing full control in all 4 quadrants, are as shown in **Figure 3-5**. The diagonal power switches are driven by the same PWM signal. Deadtime insertion is now required for PWM signal generation since the top and bottom power switches work in the complementary pair operation.

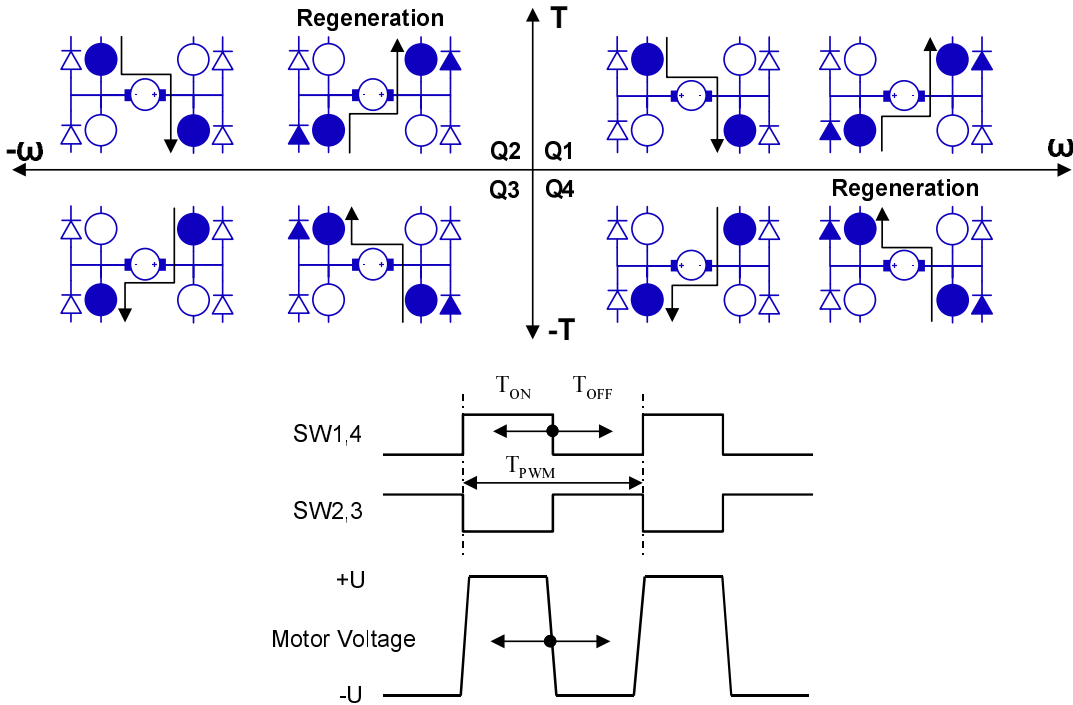


Figure 3-5. Bipolar Switching Modes

When controlling the duty cycle d_c by applying the (EQ 3-5) the load voltage u^* will be:

$$u^* = U \cdot F_{PWM} \tag{EQ 3-7}$$

There are two main disadvantages. The first one is that the rms value of the load voltage is not dependent on the actual duty cycle; it is always equal to maximum value U . This keeps the rms dependent portion of the power losses always at maximum. The second disadvantage is higher electromagnetic emission. This relates to the fact that all four switches act at the same time and $\Delta u = 2U$ during the power switching thus increasing du/dt and hence electromagnetic emission. On the other side, there is no current re-circulation to complicate current detection and limiting.

3.3.2 Unipolar Switching

The unipolar PWM technique, as can be understood from its name, is a combination of the PWM signals which results in single load voltage polarity during the PWM cycle (the load voltage can be $0/+U$ or $0/-U$).

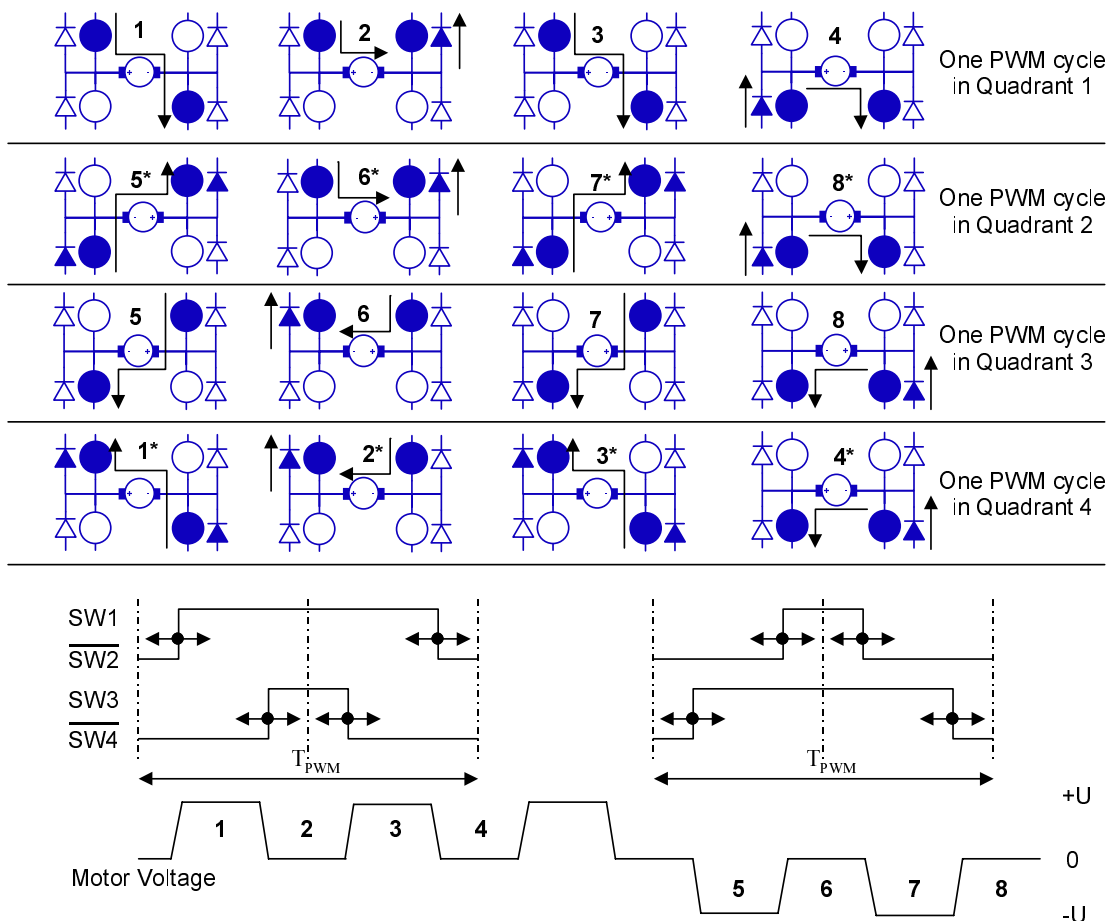


Figure 3-6. Unipolar Switching Modes

Using the PWM control signals as shown in **Figure 3-6** enables full control in all 4 quadrants just by changing the duty cycle of the presented signals. The asterisk “*” denotes the power switch states at the opposite current direction. The top and bottom power switches work in the complementary pair operation mode therefore the deadtime insertion is required.

Let us define d_{c1} and d_{c3} to be duty cycle of the power switches SW1 and SW3, respectively. Then the load voltage u^* will be:

$$u^* = U \cdot (d_{c1} - d_{c3}) \quad (\text{EQ 3-8})$$

The load voltage u^* only depends on the duty cycles difference and does not directly depend on values of the duty cycles themselves. This provides some freedom in generating unipolar PWM switching. We can then distinguish the following four main cases (the comparison is done for quadrant 1):

1. SW1 is always ON, $d_{c1} = 1$ and $d_{c3} = 1 - F_{PWM}$;
2. SW3 is always OFF, $d_{c1} = F_{PWM}$ and $d_{c3} = 0$;
3. symmetrical voltage operation, $d_{c1} = \frac{1}{2} + \frac{1}{2} \cdot F_{PWM}$, $d_{c3} = d_{c1} - F_{PWM} = \frac{1}{2} - \frac{1}{2} \cdot F_{PWM}$;
4. dc voltage u_0 injection, $d_{c1} = \frac{1}{2} \cdot F_{PWM} + u_0$, $d_{c3} = d_{c1} - F_{PWM} = u_0 - \frac{1}{2} \cdot F_{PWM}$.

In all four cases, the voltage u^* is the same as in the case of bipolar PWM technique; see (EQ 3-7). Case 4 is, in fact, the most general one. The other cases can be obtained by using appropriate u_0 .

When comparing this PWM technique with the bipolar PWM technique, electromagnetic emissions are lower due to just half du/dt . Single polarity of load voltage u also causes lower ripples of the inductive load current therefore the unipolar PWM technique is often preferred.

Unipolar switching is usually generated using the so-called “center-aligned PWM” although both, “center-aligned” and “edge-aligned” PWM can be used in principal. The terms “center-aligned” and “edge-aligned” are explained in [Section 3.4](#).

3.4 Center- and Edge-Aligned PWM

Two PWM signals with equal period can be basically aligned in two ways: center-aligned and edge-aligned.

Center-aligned PWM signals are symmetrical along the centers of ON and OFF time intervals (see [Figure 3-7](#)). Mostly an up/down counter is used in this case. The PWM output resolution is then two PWM counter counts (two PWM clock periods).

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

$$\text{PWM pulse width} = (\text{PWM value}) \times (\text{PWM clock period}) \times 2$$

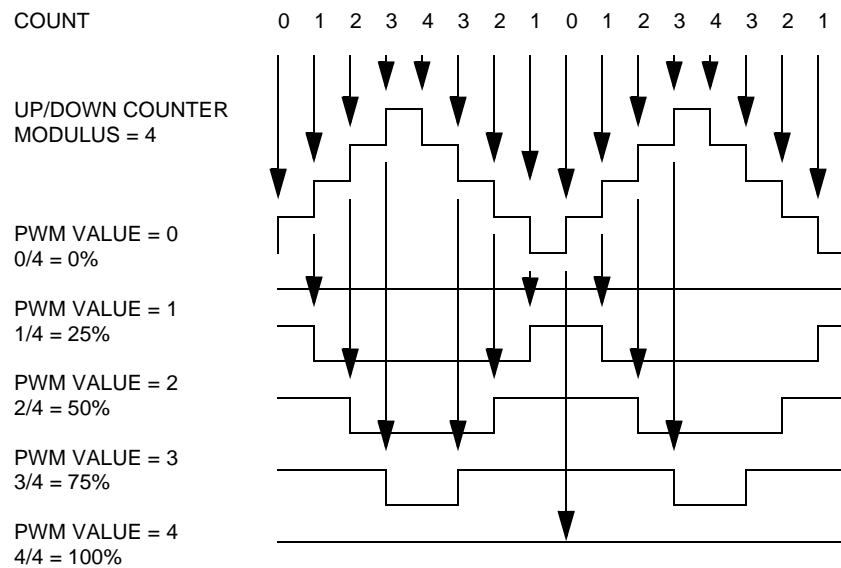


Figure 3-7. Center-Aligned PWM Output

Unlike the center-aligned PWM, the **edge-aligned PWM** ON times always start at the same time, employing an up counter output (see [Figure 3-8](#)). The PWM output resolution is one PWM counter count (one PWM clock period).

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period})$$

$$\text{PWM pulse width} = (\text{PWM value}) \times (\text{PWM clock period})$$

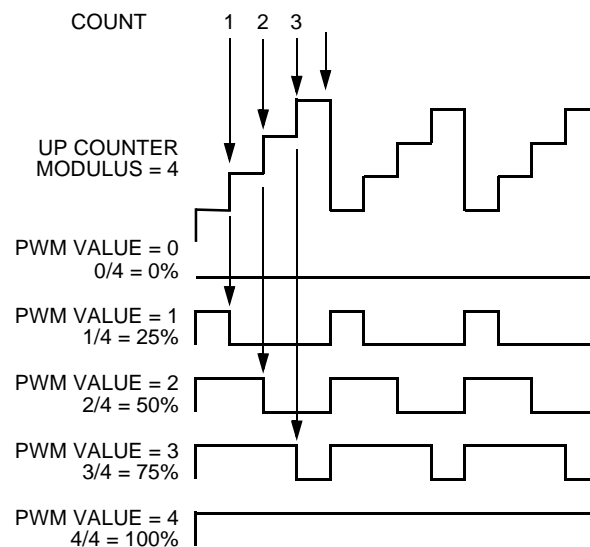


Figure 3-8. Edge-Aligned PWM Output

The PWM control frequency of the center-aligned PWM is half of the edge-aligned PWM at the same PWM clock period and PWM modulus. In order to obtain the same frequency, the PWM modulus must be cut by half. Lower pulse-width resolution could be thought as a disadvantage of the center-aligned PWM, but (as can be seen from [Figure 3-6](#)) the frequency of the load voltage is doubled. Then the PWM resolution of both PWMs, the center- and the edge-aligned, is the same if the PWM clock period and load voltage PWM frequency are constant.

The advantage of the center-aligned PWM is lower electromagnetic emission because it takes half the number of power switches to generate the same load voltage frequency. Thus, the center-aligned PWM is the preferred method of generating PWM signals.

4. MC PWM Module

4.1 Description

This application note is not intended to replace the user manual, where a high-level explanation of the motor control PWM module is provided. See [Section 7](#). [2.] for more details.

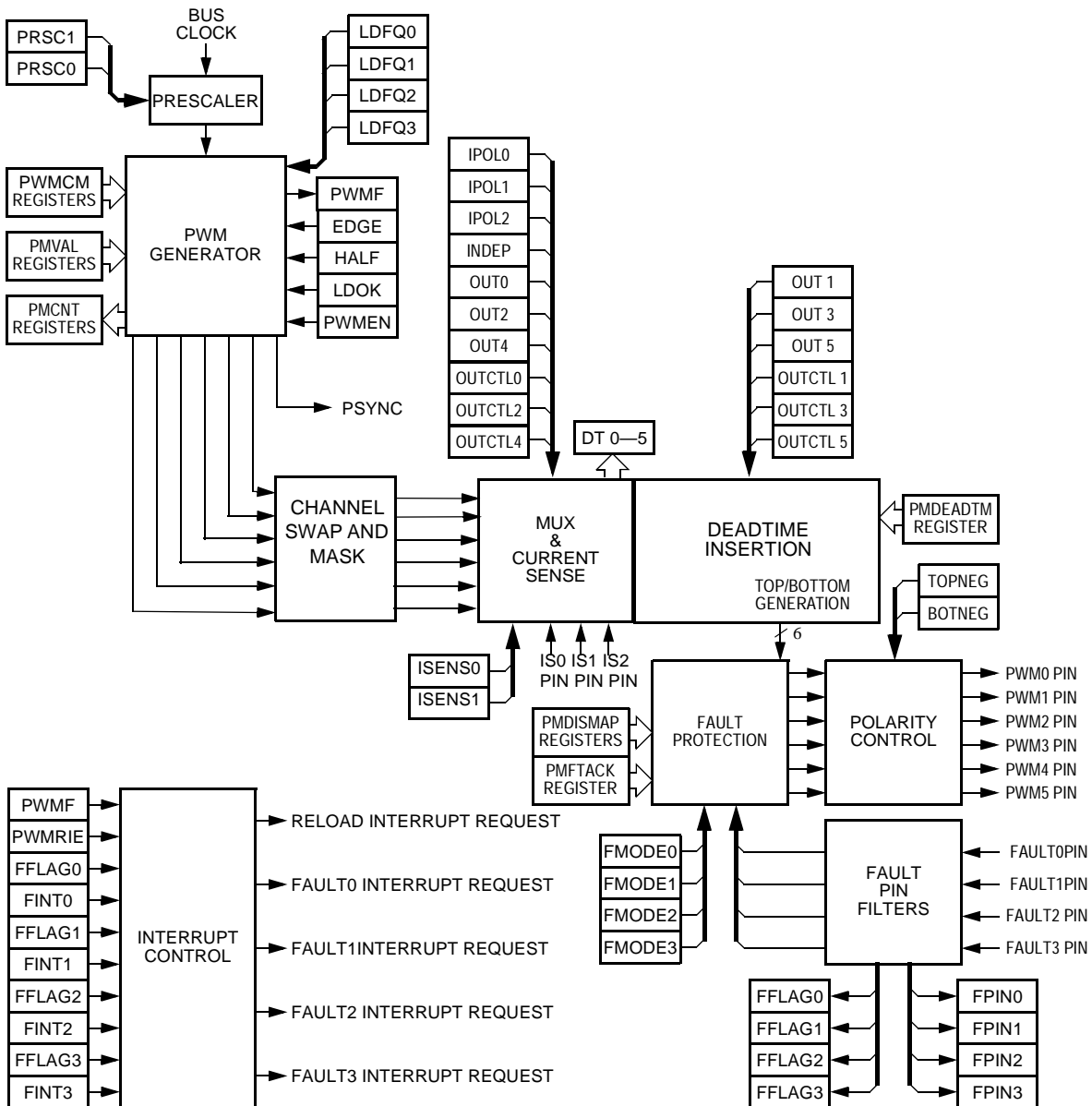


Figure 4-1. PWM Block Diagram

Preliminary Copy

4.1.1 Prescaler

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by one, two, four, and eight. The prescaler bits PRSC0 and PRSC1 in the PWM control register (PWMCTL), select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins.

4.1.2 PWM Generator

The PWM generator contains a 15-bit up/down PWM counter producing six output signals with software-selectables:

- **Alignment** - The logic state of the EDGE bit in the configuration register determines whether the PWM output is edge-aligned or center-aligned.
- **Period** - The value written to the PWM counter modulus register (PWMCM) is used to determine the PWM period. The period can also be varied by using the prescaler.
 - With edge-aligned output, the modulus is the period of the PWM output in clock cycles.
 - With center-aligned output, the modulus is one-half of the PWM output period in clock cycles.
- **Pulse Width** - The number written to the PWM value register (PWMVALx) determines the pulse width duty cycle of the PWM output in clock cycles.
 - With edge-aligned output, the pulse width is the value written to the PWM value register.
 - With center-aligned output, the pulse width is twice the value written to the PWM value register.

A PWM value less than or equal to zero deactivates the PWM output for the entire PWM period. A PWM value greater than or equal to the modulus activates the PWM output for the entire PWM period.

- **Load Enable** - The load okay bit, LDOK, enables loading the PWM generator with:
 - A prescaler divisor - from the PRSC1 and PRSC0 bits in PWM control register.
 - A PWM period - from the PWM counter modulus registers.
 - A PWM pulse width - from the PWM value registers.

LDOK prevents reloading of these PWM parameters before software is finished calculating them. Setting LDOK allows the prescaler bits and the PWMCM and PWMVALx registers to be loaded into a set of buffers. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle. Set LDOK by reading it when it is a logic zero and then writing a logic one to it. After loading, LDOK is automatically cleared.

- **Load Frequency** - The LDFQ [3:0] bits in the PWM control register (PMCTL) select an integral loading frequency of one to 16-PWM reload opportunities. The HALF bit in the PMCTL register controls half-cycle reloads for center-aligned PWMs. Note that reload opportunities can only occur at the beginning of a PWM cycle in edge-aligned mode.
- **Synchronization Output** - The PWM generates a synchronization pulse connected as an input to the synchronization module, Timer1. A high-true pulse occurs for each reload of the PWM regardless of the state of the LDOK bit. When half-cycle reloads are enabled (HALF = 1 in the PMCTL register), the pulse can occur on the half cycle.

- **Hardware Acceleration** - VLMODE[1:0] bits determine the way the Value Registers are being loaded. The write protectable ENHA bit enables the HW acceleration feature.
 - 00 = Each Value Register is accessed independently.
 - 01 = Writing to Value Register 0 to also writes to Value Registers 1 to 5.
 - 10 = Writing to Value Register 0 to also writes to Value Registers 1 to 3.

4.1.3 Channel Mask and Swap

The outputs of this block are individually masked (0 = Unmasked, 1 = Masked - channel set to a value of 0% duty cycle) and/or swapped PWM logical channels (the PWM generator outputs 0-1; 2-3; 4-5).

4.1.4 MUX and Current Sense

This block consists of the MUX logic which selects the input to the deadtime generation block from the odd-numbered/even-numbered PWM register pair and the corresponding output control bit (OUTx). The internal MUX logic is controlled by the status of the output control enable bits, the independent/complementary mode of operation bits and the selected deadtime correction method.

4.1.4.1 Software Output Control

Setting output control enable bit, OUTCTLx, enables software to drive the PWM outputs rather than the PWM generator. The OUTCTLx and OUTx bits are in the PWM output control register.

Table 4-1. Software Output Control

OUTx bit	Complementary Channel Operation	Independent Channel Operation
OUT0	1—PWM0 is active + deadtime insertion 0—PWM0 is inactive	1—PWM0 is active 0—PWM0 is inactive
OUT1	1—PWM1 is complement of PWM 0 0—PWM1 is inactive	1—PWM1 is active 0—PWM1 is inactive
OUT2	1—PWM2 is active + deadtime insertion 0—PWM2 is inactive	1—PWM2 is active 0—PWM2 is inactive
OUT3	1—PWM3 is complement of PWM 2 0—PWM3 is inactive	1—PWM3 is active 0—PWM3 is inactive
OUT4	1—PWM4 is active + deadtime insertion 0—PWM4 is inactive	1—PWM4 is active 0—PWM4 is inactive
OUT5	1—PWM5 is complement of PWM 4 0—PWM5 is inactive	1—PWM5 is active 0—PWM5 is inactive

Notes: During software output control, TOPNEG and BOTNEG still control output polarity.

In complementary channel operation, the OUT0, OUT2, OUT4 bits replace the PWM generator outputs as inputs to the deadtime generators. **Complementary channel pairs still cannot be active simultaneously**, and the deadtime generators continue to insert deadtime whenever an OUT0/2/4 bit toggles. Deadtime is not inserted when the OUT1/3/5 bit toggles. The OUT0, OUT2, OUT4 bits control the top PWM signals while the OUT1, OUT3, OUT5 bits control the bottom PWM signals.

Setting the OUTCTLx bits does not disable the PWM generators and current status sensing circuitry. They continue to run, but no longer control the output pins. When the OUTCTLx bits are cleared, the outputs of the PWM generator become the inputs to the deadtime generators at the beginning of the next PWM cycle. Software can drive the PWM outputs even when PWM enable bit (PWMEN) is set to zero.

4.1.4.2 Deadtime Correction

In the complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage (see [Figure 4-2](#); refer also to [Section 7](#). [2.] and [3.]).

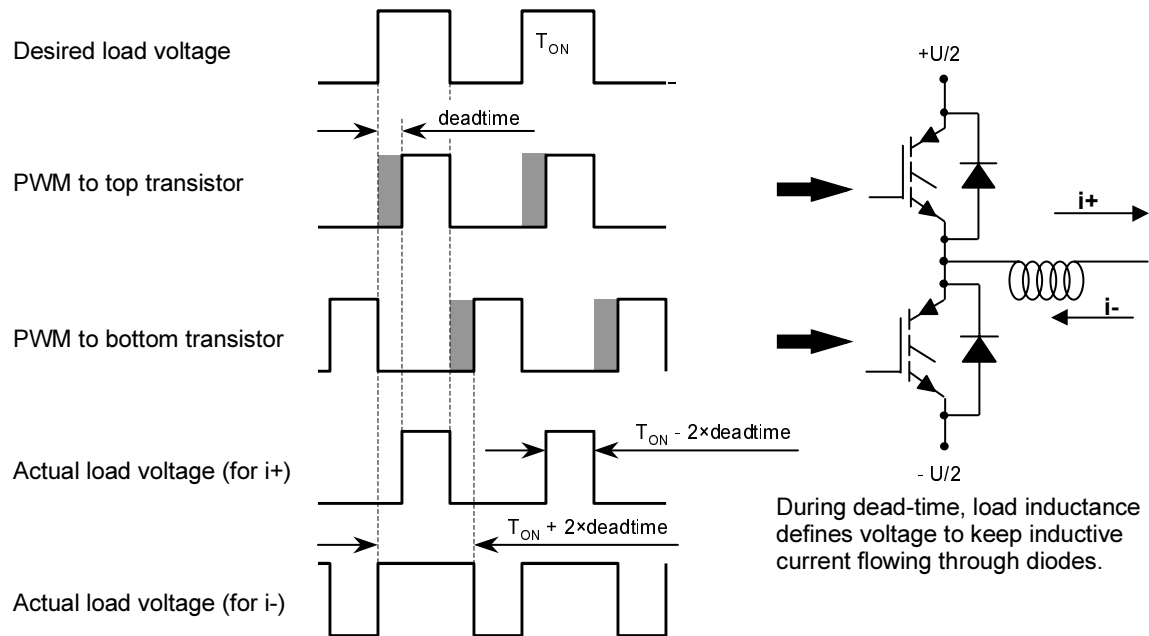


Figure 4-2. Distortion Caused by Inductance

On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque (current) ripple and rough operation (see [Figure 4-3](#)).

Preliminary Copy

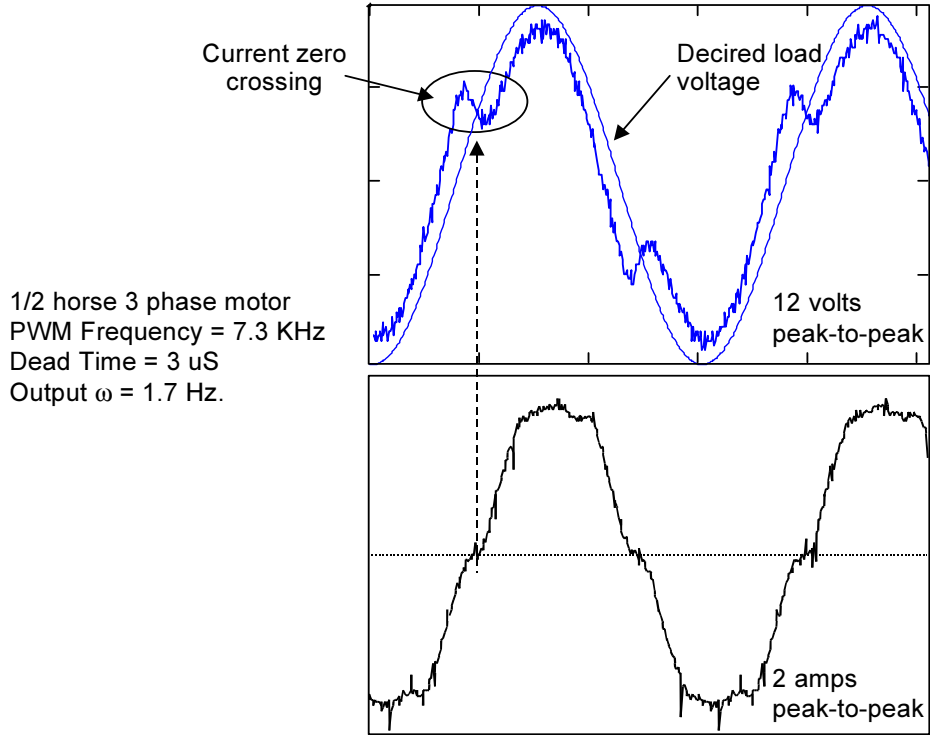


Figure 4-3. Distortion on Motor Voltage and Current Waveforms

Fortunately the deadtime distortion can be successfully corrected (compensated) by giving the PWM module information on which transistor is controlling at a given time.

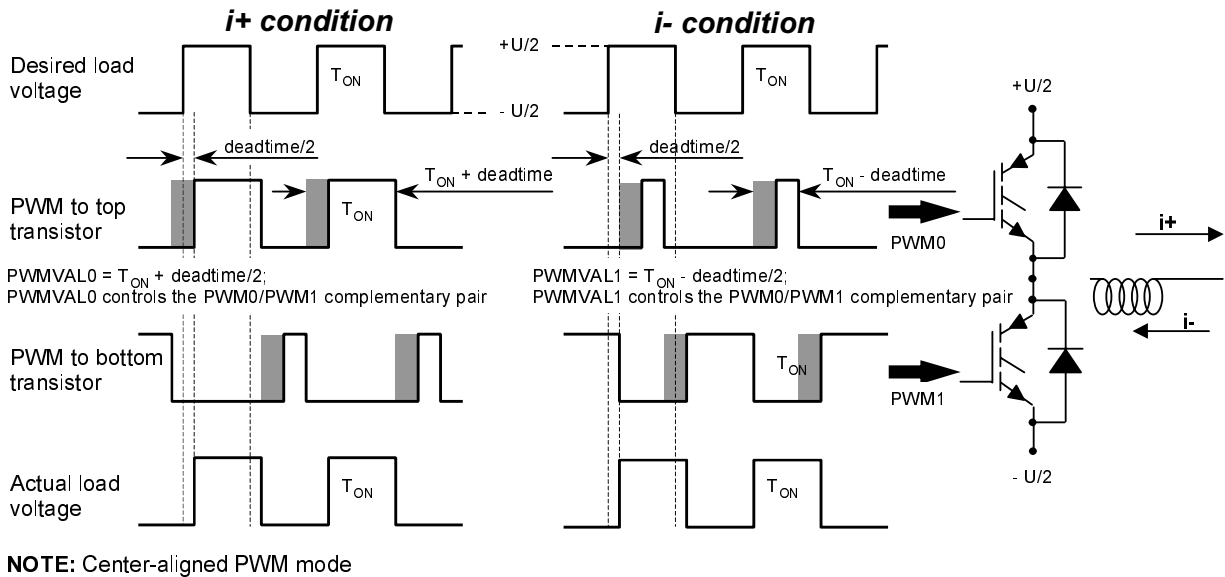


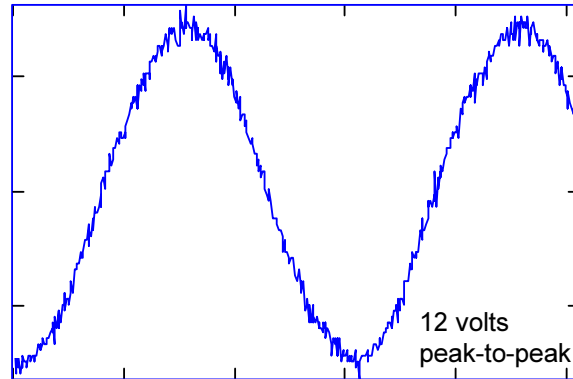
Figure 4-4. Distortion Correction

Internal odd-numbered/even-numbered PWM MUX logic provides a hardware support for “contra-modulating” the output voltage (deeply explained in [Section 7](#). [3.]). The user software is still

responsible for calculating both compensated PWM values (for i+ and i- conditions) prior to placing them in an odd-numbered/even-numbered PWM register pairs.

Voltage with Correction Enabled

By using the voltage information obtained during the dead-time, the software can “counter-modulate” the PWM waveforms to cancel the distortion.



Current with Correction Enabled

1/2 horse 3 phase motor
PWM Frequency = 7.3 KHz
Dead Time = 3 uS
Output ω = 1.7 Hz.

Output waveforms obtained with voltage sensor employing hysteresis.

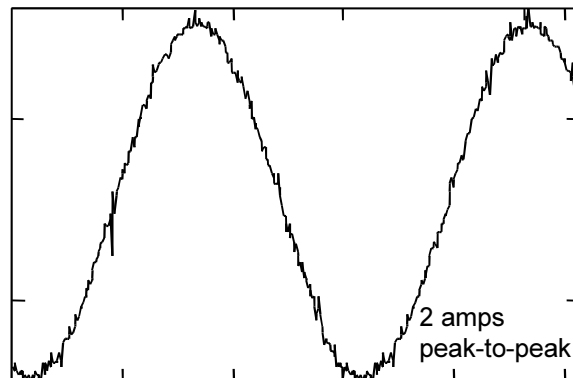


Figure 4-5. Results of Distortion Correction

Three methods are supported by the MC PWM module to determine whether the odd or the even PWMVAL register controls the pulse width at any given time.

- Manual correction
- Automatic current status correction
 - during deadtime
 - when the PWM counter value equals the value in the PWM counter modulus registers

Table 4-2. Correction Method Selection

ISENS[1:0]	Correction method
0X	Manual correction or no correction
10	Current status sample correction on pins IS1, IS2, and IS3 during deadtime ¹
11	Current status sample on pins IS1, IS2, and IS3 ² At the half cycle in center-aligned operation At the end of the cycle in edge-aligned operation

¹ The polarity of the ISx pin is latched when both the top and bottom PWMs are off. At the 0% and 100% duty cycle boundaries, there is no deadtime, so no new current value is sensed.

² Current is sensed even with 0% or 100% duty cycle.

Note: Assume the user will provide current status sensing circuitry causing the voltage at the corresponding input pin to be low for positive current and high for negative current. In addition, it assumes the top PWMs are PWM 0, 2, and 4 while the bottom PWMs are PWM 1, 3, and 5.

In the **manual correction mode**, writing to the IPOL0–IPOL2 bits selects either the odd or the even PWM value registers to use in the next PWM cycle.

Table 4-3. Top/Bottom Manual Correction

Bit	Logic state	Output Control
IPOL0	0	PWMVAL0 controls PWM0/PWM1 pair
	1	PWMVAL1 controls PWM0/PWM1 pair
IPOL1	0	PWMVAL2 controls PWM2/PWM3 pair
	1	PWMVAL3 controls PWM2/PWM3 pair
IPOL2	0	PWMVA4 controls PWM4/PWM5 pair
	1	PWMVAL5 controls PWM4/PWM5 pair

Notes: IPOLx bits are buffered so only one PWMVALx register is used per PWM cycle. If an IPOLx bit changes during a PWM period, the new value does not take effect until the next PWM period regardless of the state of the load okay bit (LDOK).

In the **automatic current status correction mode** a current sense pin, ISx, for a PWM pair selects either the odd or the even PWM value registers to use in the next PWM cycle. The selection is based on user-provided current sense circuitry driving the ISx pin high for negative current and low for positive current

Table 4-4. Top/Bottom Current-Sense Correction

Pin	Logic state	Output control
IS0	0	PMVAL0 controls PWM0/PWM1 pair
	1	PMVAL1 controls PWM0/PWM1 pair
IS1	0	PMVAL2 controls PWM2/PWM3 pair
	1	PMVAL3 controls PWM2/PWM3 pair
IS2	0	PMVAL4 controls PWM4/PWM5 pair
	1	PMVAL5 controls PWM4/PWM5 pair

Notes: Values latched on the ISx pins are buffered so only one PWM register is used per PWM cycle. If a current status changes during a PWM period, the new value does not take effect until the next PWM period.

4.1.5 Deadtime Insertion - Top/Bottom generation

The write-protectable Independent or Complimentary Pair Operation (INDEPxx) Bits determine whether the motor control PWM channels will be independent PWMs or complementary PWM pairs. While in the complementary mode, each PWM pair can be used to drive top/bottom transistors. Ideally, the PWM pairs are an inversion of each other. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Notes: *To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period.*

Deadtime generators automatically insert software-selectable activation delays into each pair of PWM outputs. The Deadtime Register (PMDEADTM) specifies the number of PWM clock cycles to use for deadtime delay. Every time the deadtime generator inputs changes state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state. [Section 4.1.4.1](#) describe deadtime insertion during the software control of the PWM outputs.

4.1.6 Fault Protection

Fault protection can disable any combination of PWM pins. Faults are generated by a logic one on any of the FAULT pins. Each FAULT pin can be mapped arbitrarily to any of the PWM pins.

When fault protection hardware disables PWM pins, the PWM generator continues to run, only the output pins are deactivated. The fault decoder disables PWM pins selected by the fault logic and the disable mapping registers (write protectable). Each bank of four bits, allocated in the disable mapping registers, control the mapping for a single PWM pin. Refer to [Table 4-5](#) for fault mapping matrix.

The fault protection is enabled even when the PWM is not enabled; therefore, a fault will be latched in and will be cleared in order to prevent an interrupt when the PWM is enabled.

Table 4-5. PWM Pin - Fault Mapping Matrix

PWM Pin to be disabled	Fault0	Fault1	Fault2	Fault3	PWM Disable Mapping Register
	to assign Fault and PWM pin set DISMAP bit #				
PWM0	0	1	2	3	PMDISMAP1 [0:3]
PWM1	4	5	6	7	PMDISMAP1 [4:7]
PWM2	8	9	10	11	PMDISMAP1 [8:11]
PWM3	12	13	14	15	PMDISMAP1 [12:15]
PWM4	16	17	18	19	PMDISMAP2 [0:3]
PWM5	20	21	22	23	PMDISMAP2 [4:7]

4.1.7 Fault Pin Filters

Each fault pin has a filter to test for fault conditions. After every IPBus cycle sets the FAULTx pin at logic zero, the filter synchronously samples the pin once in each of the next two cycles. If both samples are logic ones, the corresponding FAULTx pin bit, FPINx, and FAULTx pin flag, FFLAGx, are set. The FPINx bit remains set until the pin returns to logic zero and the filter samples a logic zero synchronously once in the following IPBus cycle. Clear FFLAGx by writing a logic one to the corresponding fault acknowledge bit, FTACKx.

If the FIEEx, FAULTx pin interrupt enable bit is set, the FFLAGx flag generates a CPU interrupt request. The interrupt request latch remains set until:

- software clears the FFLAGx flag by writing a logic one to the FTACKx bit
- software clears the FIEEx bit by writing a logic zero to it

- a reset occurs

The two methods can be used to clear the fault flags:

- **Automatic Fault Clearing** - when FMODEx is set, disabled PWM pins are enabled when the FAULTx pin returns to logic zero and a new PWM half cycle begins. Clearing the FFLAGx flag does not affect disabled PWM pins.
- **Manual Fault Clearing** - clearing a Fault Mode bit, FMODEx, configures faults from the FAULTx pin for manual clearing.
 - PWM pins disabled by the FAULT0 pin or the FAULT2 pin are enabled when software clears the corresponding FFLAGx flag. The PWM pins are enabled when the next PWM half cycle begins regardless of the logic level detected by the filter at the fault pin.
 - PWM pins disabled by the FAULT1 pin or the FAULT3 pin are enabled when software clears the corresponding FFLAGx flag and the filter detects a logic zero on the fault pin at the start of the next PWM half cycle boundary.

Notes: Fault protection also applies during software output control when the OUTCTLx bits are set. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, PWMEN = 1. But the OUTx bits can control the PWM pins while the PWM generator is off, PWMEN = 0. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

4.1.8 Polarity Control

Output polarity of the PWMs are determined by two write-protectable options: TOPNEG and BOTNEG bits. The top polarity option, TOPNEG, controls the polarity of PWM0, PWM2 and PWM4. The bottom polarity option, BOTNEG, controls the polarity of PWM1, PWM3 and PWM5. *Positive* polarity means when the PWM is active its output is high. Conversely, *negative* polarity means when the PWM is active its output is low.

This feature is very useful when electrical isolation is needed between the PWM control signals and the power switch drivers. Because of nature of their internal circuit functionality, most of the ultra-fast optocouplers reverse the logic polarity. It can be compensated for by internal PWM module polarity setting. In particular, the ability to set different polarity for top and bottom switches permits low cost design where low side power switch drivers are driven directly and the top ones use the optocouplers.

Notes: The optocoupler can be directly connected (with resistor limiting current) to the PWM pins thanks to the high sink current capability of the PWM pins.

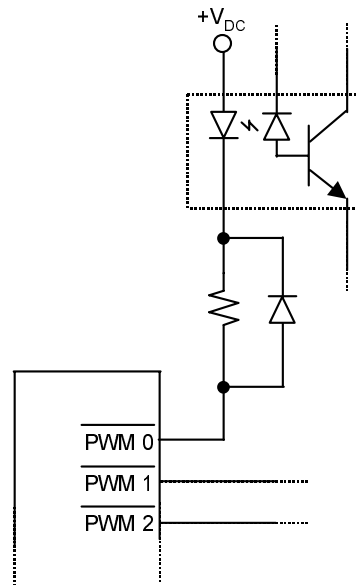


Figure 4-6. Direct PWM Pin-optocoupler Connection

4.1.9 Interrupt Control

Five PWM sources can generate CPU interrupt requests:

- Reload flag (PWMF)—PWMF is set at the beginning of every reload cycle. The reload interrupt enable bit, PWMRIE, enables PWMF to generate CPU interrupt requests. PWMF and PWMRIE are in PWM control register (PMCTL).
- Fault flags (FFLAG0–FFLAG3)—The FFLAG_x bit is set when a logic 1 occurs on the FAULT_x pin. The fault pin interrupt enable bits, FIE0–FIE3, enable the FFLAG_x flags to generate CPU interrupt requests. FFLAG0–FFLAG3 are in the PWM Fault Status & Acknowledge Register (PMFSA). FIE0–FIE3 are in the PWM Fault Control Register (PMFCTL).

5. Motor Control PWM Generation

This section provides an explanation of MC PWM module settings as well as service of the module needed during normal operation. Since the motor control field covers a very wide spectrum of applications there are a large number of possible PWM generation strategies. Because the PWM module implemented on Motorola DSP controllers offers a high degree of flexibility, it exceeds the scope of this application note to cover all possible setting combinations. Therefore this explanation is based on examples of the most common strategies of driving the following popular electric motors:

- AC Induction motor (ACIM)
- Permanent Magnet Synchronous motor (PMSM)
- Brushless DC motor (BLDC motor)
- Switched Reluctance motor (SRM)
- DC Brushed motor (DC motor)

The first two motor types will be grouped together because power stage topology and the PWM control are common for both types.

5.1 AC Induction Motor and PM Synchronous Motor

Both motors are normally driven from a variable frequency and voltage source. Such source can be implemented by proper PWM control of the 3-phase power stage bridge as shown in [Figure 5-1](#)

The full four-quadrant PWM control (current and voltage change polarity) with complementary mode and deadtime insertion is a must; other settings, like center- or edge-aligned PWM, are optional. Usually the center-aligned (therefore unipolar) PWM is chosen to lower electromagnetic emissions. The PWM frequency selection is always a compromise between audible noise, electromagnetic emissions, current ripples and power switching losses. It will be different for industrial, appliance or automotive environments.

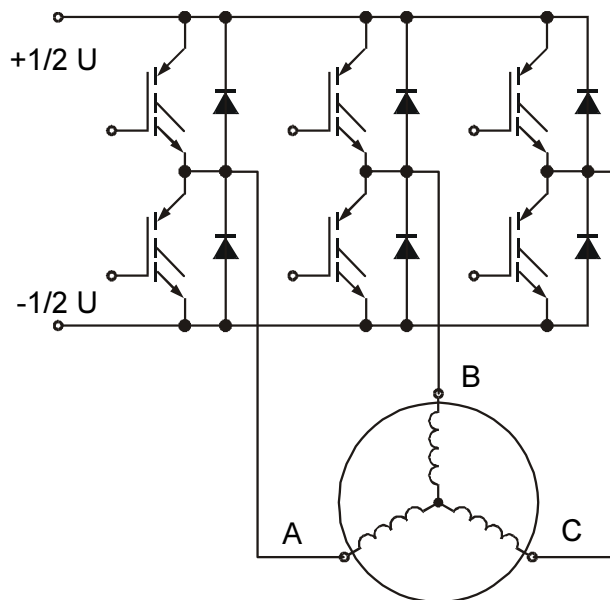


Figure 5-1. ACIM, PMSM 3-ph Power Stage

5.1.1 3-ph PWM Static Setting

The MC PWM module provides an excellent feature set to drive ACIM and PMSM.

The following setting is used to initialize the module for given motor types:

- **Complementary Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to zero (reset state) in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - based on used hardware the polarity for top and bottom power switches needs to be set in the PMCFG accordingly.
- **Center-aligned PWM** (EDG) - bit is cleared (reset state) in the PMCFG.
- **Deadtime** (PMDEADTM) - it needs to match the switching characteristics of used power switches. The usual value fits between 1.5-3 usec => PWMDT = 60-120 dec @ 25 nsec PWM clock period.

Note that the deadtime is defined as a number of PWM clock cycles and is affected by changes of the prescaler value.

- **PWM Clock prescaler** (PRSC [1:0]) - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency** (LDFQ[0:3]) + **Half Cycle reload** (HALF) - this setting strongly depends on the application requirements. For fast servo application, a half cycle reload feature is often used, while standard industrial control uses reload every PWM opportunity or every 2 PWM opportunities, resulting in a 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.
- **Deadtime Correction Method**
 - Current Status Bits (ISENS[1:0]) - select correction method using [Table 4-2](#)
 - Current Polarity Bits (IPOLx) - if in manual deadtime correction mode set the Current Polarity Bits accordingly or leave cleared (the reset state) for other correction modes.
- **PWM Frequency, Modulus** (PWMCM) - setting must reflect the required PWM frequency. Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.
- **PWM Pins SW Output Control** - Output Pad Enable (PAD_EN = 1). No output control is needed so the OUTCTLx and OUTx pins in the PMOUT register can be left at their reset state.
- **PWM Channel Control** - no channel control needed (HW acceleration disabled, no masked PWM generator channels; no swapped channel; each Value Register is accessed independently); the PMCCR register can be left in its reset state.
- **Faults** -
 - **PWM Disable Mapping** (PMDISMAP1-2) - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#) The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.
 There is no one “magic” setting satisfying all needs. Mostly the Fault 0,1 and 2 are connected to two phase over-current and one DC-bus over-current detection circuitries, respectively. The Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is that any fault disables all PWMs. The PWM disable mapping bits DISMAP0-23 = 1 which is the reset state.
 - **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an application must perform some more sophisticated checking before enabling PWM again the manual fault clearing should be selected.
 - **FAULT Interrupt Enable** (FIEx) - setting these bits in PMFCTL enables CPU interrupt requests generated by the FAULTx pin. Usually the fault interrupt is enabled when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.
- **Write-protection** - If required the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:

- PMCFG - the PWM polarity, the complementary PWM pair operation mode and center-aligned PWM channels
- PMDEADTM - the deadtime,
- PMDISMAP1-2 - the PWM fault disable mapping matrix
- PMCCR - ENHA = 0 - the disabled HW acceleration features (Value Register Load Mode - VLMODE, PWM generator channels swapping)

The functions which are still available are: PWM Value setting, PWM Frequency setting, PWM Clock prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting, odd/even PWMVALx selection for Deadtime Correction, Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control and PWM generator channels masking.

- **PWM Value Registers** (PWMVALx) - usually all duty cycles are initially set the same - producing no current (zero phase-phase voltage). 50% duty cycle would produce “natural zero” level (refer to [Figure 3-1](#) and (EQ 3-5)), while it helps to keep charging the bootstrap circuitry in the power switch drivers. The PWM Value calculation is as follows:

PWM value x = (duty cycle x)/100% × (PWM modulus)

PWMVALx = 50% / 100% × 1250 dec = 625 dec for previously selected PWM modulus.

The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.

- **PWM Reload Interrupt Enable** (PWMRIE) - this bit enables the PWMF flag to generate CPU interrupt requests. See also the note below.
- **PWM Enable** (PWMEN) - this bit enables the PWM generator and the PWM pins. It should be set at the end of whole initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set. In complementary channel operation with current-status correction selected, PWM value registers one, three, and five control the outputs for the first PWM cycle.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state; therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.1.2 3-ph Waveforms Generation

Now the MC PWM module is initialized as explained in previous section and is ready for cycle-by-cycle generation of the 3-phase waveforms and safe operation even when fault occurs.

Since the user can separately control each PWM complementary pair, almost any waveform can be generated. For harmonic motor types such as ACIM and PMSM the generation of sinusoidal current needs to be achieved. Although there are many different sinusoidal current PWM control strategies, the common feature of all of them is the sinusoidal shape of the phase-phase voltage. It is because motor terminals “see” only the voltage difference - phase-phase voltage; therefore the PWM voltage waveform, generated by each branch (half-bridge) of the 3-ph power stage, does not necessarily need to be sinusoidal (refer to section 3.3.2 and (EQ 3-8)).

Most common 3-phase PWM waveform generation techniques can be divided into two main groups:

- **3-phase PWM generation based on the waveform table** (sinusoidal; sinusoidal + 3rd harmonic injection; sinusoidal + sine cap injection; etc.),

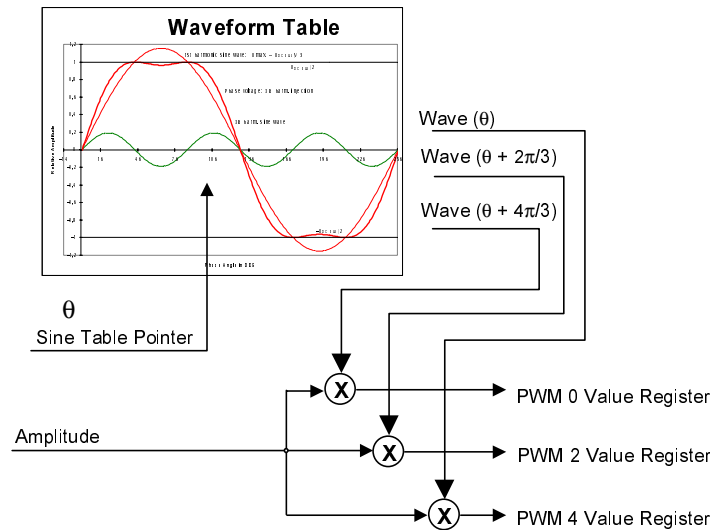


Figure 5-2. Generation using Waveform Table

- **3-phase PWM generation based on Space Vector Modulation (SVM) and its variants** (standard SVM; NULL = O₀₀₀; NULL = O₁₁₁; Null = O₁₁₁ in sectors 1,3,5 + Null = O₀₀₀ in sectors 2,4,6; etc.).

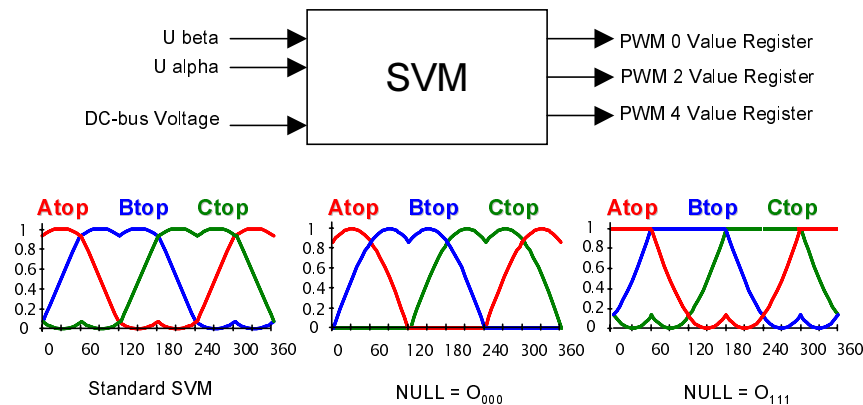


Figure 5-3. Generation using SVM

Both groups of 3-phase PWM waveform generation techniques can provide the same result in term of switching losses, harmonic content, phase current ripples, etc.

The first group, which is utilizing waveform table, better fits an application where the table pointer is naturally used by control application - for example V/Hz or natural Vector Control of the ACIM control - while the second group is suitable for Field Oriented Vector Control method of ACIM and PMSM.

The detailed description of the different PWM waveform generation strategies, explanation of principle and necessary theoretical background will be given in separate application note.

Notes: The Embedded SDK provides 3-phase PWM waveform generation algorithms with seamless interfaces to PWM module drivers. For more details see Embedded SDK documentation [4.] and supplied application examples.

Preliminary Copy

5.1.3 Cycle-by-cycle Modification of PWM Waveform

The MC PWM Module is initialized to generate complementary, center-aligned PWM waveforms with given duty cycle and PWM frequency.

Performing the following steps on regular basis (each reload) leads to step-by-step modification of the PWM waveform shape and hence modulation of the motor load voltage.

1. Upon a reload opportunity (defined by the PWM Load Frequency + Half Cycle reload setting), regardless of whether an actual reload occurs as determined by LDOK bit, the PWMF reload flag is set.
2. If the PWM reload interrupt enable bit, PWMRIE is set, the PWMF flag generates CPU interrupt requests allowing software to calculate new PWM Values in real time based on one of the previously described 3-phase PWM waveform generation techniques.

Notes: When PWMRIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

3. Then the following steps must be performed (usually in the PWM reload ISR):
 - the PWMVALx registers (buffers) must be updated with new values previously calculated by a 3-phase PWM waveform generation algorithm
 - the PWMF reload flag must be acknowledged by reading PWM Control Register with PWMF set and then writing a logic zero to the PWMF bit
 - the load okay bit, LDOK, needs to be set by reading it when it is a logic zero and then writing a logic one to it. LDOK prevents reloading of these PWM parameters before software is finished calculating them
4. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle.
5. After loading, LDOK is automatically cleared.

Figure 5-4 is a graphical representation of above explained process.

You may have already noticed that the load okay bit, LDOK, enables loading the PWM generator with more new parameters. They are:

- a prescaler divisor - from the PRSC1 and PRSC0 bits in PWM Control Register
- a PWM period - from the PWM counter modulus registers (PWMCM)
- a PWM pulse width - from the PWM value registers (PWMVALx)

This feature enables an on-fly variation of the PWM frequency - a quite common acoustic noise elimination technique used when setting the PWM frequency above audible frequency range is not possible for any reason (e.g. too high switching losses). Spreading acoustic noise to wider audible frequency spectrum suppresses sharp acoustic noise normally produced by single PWM frequency. Although it does not limit total transmitted acoustic power, human ear receives spread frequency (“soft”) noise much more positively.

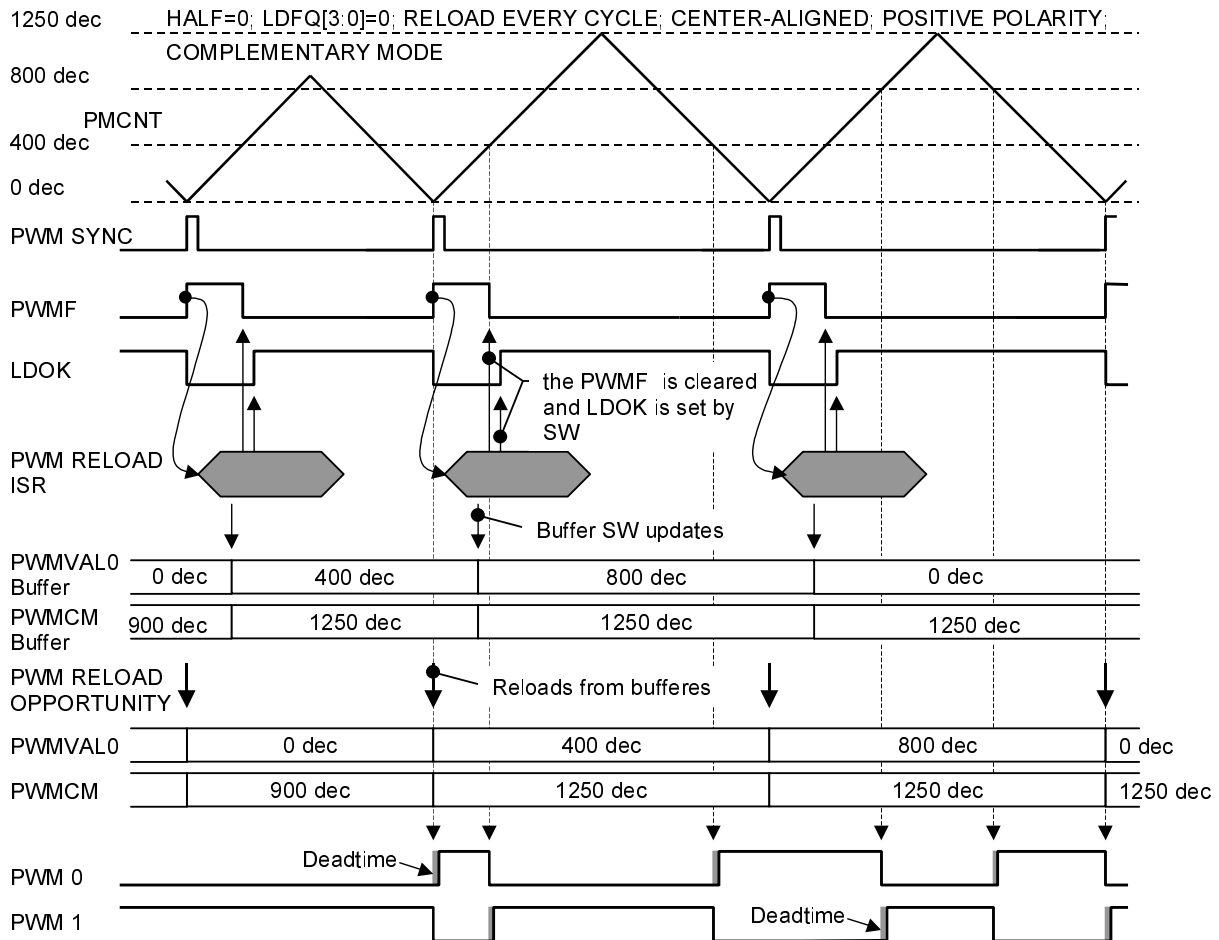


Figure 5-4. Cycle-by-cycle Modification of PWM Waveform

One should not neglect the deadtime correction. If the manual deadtime correction method is performed, the IPOLx bits can be updated in the PWM reload ISR to define whether odd or even numbered PWMVALx registers control the PWM complementary pairs.

Notes: IPOLx bits are buffered so only one PWM register is used per PWM cycle. If an IPOLx bit changes during a PWM period, the new value does not take effect until the next PWM period regardless of the state of the load okay bit (LDOK).

5.2 BLDC Motor

The BLDC motor power stage topology can be as in case of ACIM and PMSM, but the PWM control is different. Therefore it is covered in this separate section.

The BLDC motor is driven by the three phase rectangular shape (“DC”) voltages which create a rotational field. Usually one phase is left unpowered for commutation period. This easy created shape ensures the simplicity of control and drive. But the rotor position must be known at certain angles.

The PWM control of the BLDC motor can be split into two independent processes:

Preliminary Copy

- **commutation** - so called six-step control, which sequentially deactivates one motor phases as indicated in [Figure 5-5](#) The power stage topology then changes to “full bridge” shown in [Figure 3-4](#)
- **voltage control** - DC voltage applied onto the two other motor phases is controlled cycle-by-cycle using a speed, current or torque controller.

The phase commutation process is usually serviced in the timer interrupt service routine. The time of initiated interrupts then reflects the position signal or the elapsed time (proportional to the angular phase-shift) from position signal.

The voltage control is similar as for DC brushed motors. The control loop is calculated regularly with given period. In most cases, this period matches the PWM reload period. The motor behaves as a DC motor and can easily work on all four quadrants.

Preliminary Copy

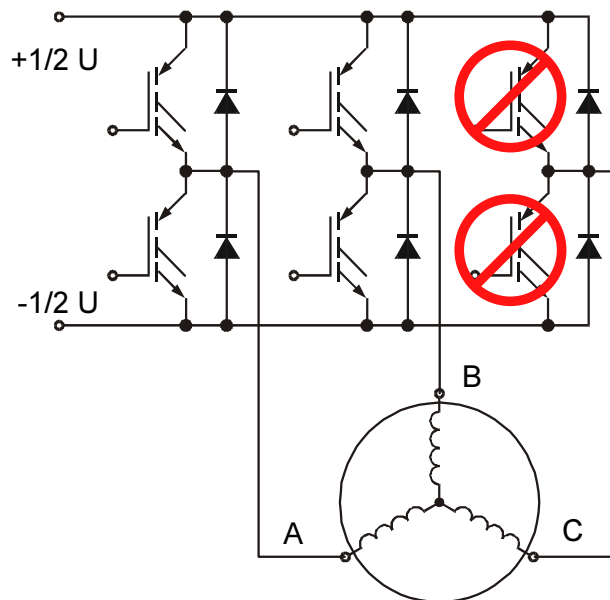


Figure 5-5. BLDC Motor 3-ph Power Stage

5.2.1 PWM for Classical BLDC Motor Control

The complementary mode and the deadtime insertion are needed when full four-quadrant PWM control (current and voltage change polarity) is required. Other settings, like center- or edge-aligned PWM, are optional. Usually the center-aligned (therefore unipolar; refer to [Figure 3-6](#) for PWM signal shapes) PWM is chosen to lower electromagnetic emissions. If the required DC voltage, to be created by PWM, corresponds to $U \times F_{PWM}$ then the PWM duty cycle of first power stage “branch” voltage is:

$$d_{c1b} = \frac{1}{2} + \frac{1}{2} \cdot F_{PWM} \tag{EQ 5-1}$$

and the PWM duty cycle of second power stage “branch” voltage is:

$$d_{c2b} = \frac{1}{2} - \frac{1}{2} \cdot F_{PWM} \quad (\text{EQ 5-2})$$

This is the so-called symmetrical voltage operation, as described in section [Section 3.3.2](#).

The hardware feature, dedicated for BLDC motor PWM control, is swapping of odd and even PWM generator outputs supporting the symmetrical voltage operation (see [Figure 5-6](#)). The even and odd PWMVALx are loaded in accordance with the equations (EQ 5-1) and (EQ 5-2), respectively.

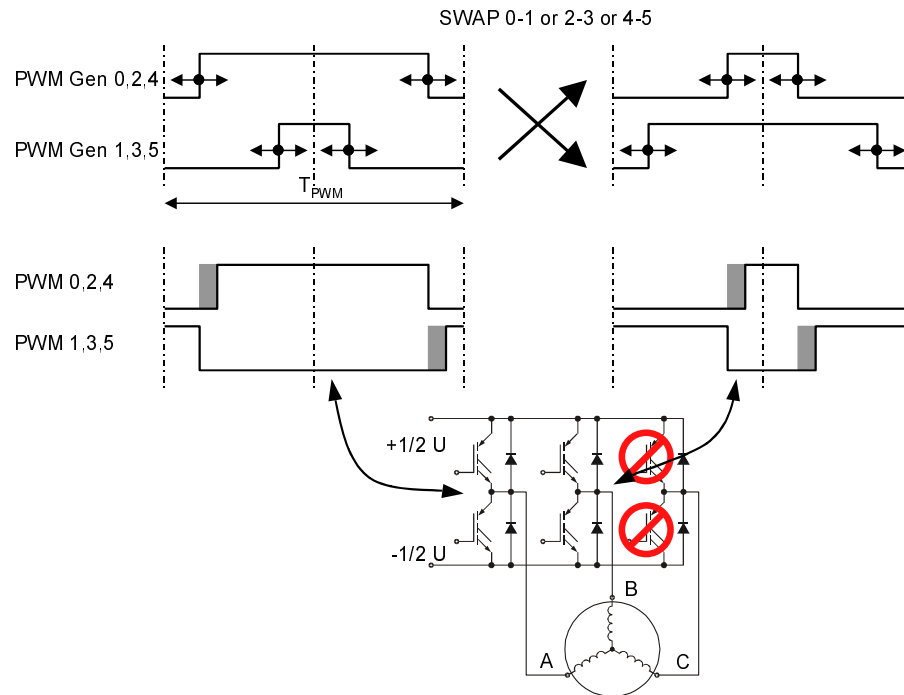


Figure 5-6. PWM Swapping for BLDC Motor

The BLDC motor, unlike the PMSM, has always one phase unpowered (deactivated PWM signal). The commutation control benefits from the PWM SW Output control of individual PWM outputs. Thus the commutation of the PWM control signals can be performed immediately, without changing the content of the PWM value registers (changes made to PWMVALx registers take effect at reload opportunity - defined by LDFQ[0:3], HALF). These changes are then asynchronous to the PWM reload period.

Choosing the PWM frequency is always a compromise between audible noise, electromagnetic emissions, current ripples and power switching losses. It will be different for industrial, appliance or automotive environments.

5.2.1.1 BLDC PWM Static Setting

The following setting is used to initialize the module for classical BLDC motor control:

- **Complementary Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to zero (reset state) in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - based on used hardware the polarity for top and bottom power switches needs to be set in the PMCFG accordingly.
- **Center-aligned PWM** (EDG) - bit is cleared (reset state) in the PMCFG.

- **Deadtime (PMDEADTM)** - it needs to match the switching characteristics of used power switches. The usual value fits between 1.5-3 usec => PWMDT = 60-120 dec @ 25 nsec PWM clock period.

Note that the deadtime is defined as a number of PWM clock cycles and is affected by changes of the prescaler value.

- **PWM Clock prescaler (PRSC [1:0])** - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency (LDFQ[0:3]) + Half Cycle reload (HALF)** - this setting strongly depends on the application requirements. For fast servo application the half cycle reload feature is often used while standard industrial control uses reload every PWM opportunity or every 2 PWM opportunities. This results in 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.
- **Deadtime Correction Method**

- Current Status Bits (ISENS[1:0]) - must be set to 0 for no correction. Ssee [Table 4-2](#)

- Current Polarity Bits (IPOLx) - must be set to 0 or the reset state can be left.

- **PWM Frequency, Modulus (PWMCM)** - setting must reflect the required PWM frequency.

Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.

- **PWM Pins SW Output Control (PMCCR)** - Output Pad Enable (PAD_EN = 1). Output control is needed to provide commutation means. Set the OUTx = 0 for PWM output pins deactivation. Set the OUTCTLx to one of the three vectors - [0,0,0,0,1,1]; [0,0,1,1,0,0]; [1,1,0,0,0,0] - to deactivate one of the three phases reflecting the initial commutation stage (rotor position).

- **PWM Channel Control** - PWM channel control is needed, ENHA = 1,

- MSKx = 0 no masked PWM generator channels,

- channel swapped - SWPx, set one of the three vectors - [1,0,0]; [0,0,1]; [0,1,0] - in accordance with the initial commutation stage (rotor position),

- VLMODE [1:0] = 0 each PWMVALx is accessed independently.

- **Faults** -

- **PWM Disable Mapping (PMDISMAP1-2)** - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#). The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.

There is no one “magic” setting satisfying all needs. Mostly the Fault 2 is connected to the DC-bus over-current detection circuitry and the Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is such that both faults disable all PWMs and the Fault 0,1 are inactive. The PWM disable mapping bits:

DISMAP2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22 and 23 = 1 other DISMAPx bits = 0.

- **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an

application must perform some more sophisticated checking before enabling PWM again the manual fault clearing should be selected.

- **FAULT Interrupt Enable** (FIEx) - setting bits 2 and 3 in PMFCTL enables CPU interrupt requests generated by the FAULT2,3 pins. Usually the fault interrupt is enabled when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.
- **Write-protection** - If required, the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:
 - PMCFG - the PWM polarity, the complementary PWM pair operation mode and center aligned PWM channels,
 - PMDEADTM - the deadtime
 - PMDISMAP1-2 - the PWM fault disable mapping matrix
 - PMCCR - ENHA = 1 - the enabled HW acceleration features (Value Register Load Mode - VLMODE and PWM generator channels swapping can be modified)

The functions which are still available follow: PWM Value setting, PWM Frequency setting, PWM Clock prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting (not suitable), odd/even PWMVALx selection (not suitable), Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control, PWM generator channels masking and swapping.

- **PWM Value Registers** (PWMVALx) - usually all duty cycles are initially set the same - producing no current (zero phase-phase voltage). 50% duty cycle would produce “natural zero” level (refer to [Figure 3-1](#) and (EQ 3-5)), while it helps to keep charging the bootstrap circuitry in the power switch drivers. The PWM Value calculation is as follows:

$$\text{PWM value } x = (\text{duty cycle } x) / 100\% \times (\text{PWM modulus})$$

$$\text{PWMVAL}_x = 50\% / 100\% \times 1250 \text{ dec} = 625 \text{ dec for previously selected PWM modulus.}$$
 The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.
- **PWM Reload Interrupt Enable** (PWMRIE) - this bit enables the PWMF flag to generate CPU interrupt requests. See also the notes below.
- **PWM Enable** (PWMEN) - this bit enables the PWM generator and the PWM pins. It should be set at the end of whole initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state; therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.2.1.2 Voltage Control - Cycle-by-cycle Modification of PWM

The MC PWM Module is initialized to generate complementary, center-aligned PWM waveforms with given duty cycle and PWM frequency.

As is described in previous sections, the PWM duty cycle is periodically updated according to the speed (or torque, or current) controller output. Such controller calculates the new values of the PWM duty cycles (PWMVALx).

Performing the following steps on regular basis (each reload) leads to modification of the PWM duty cycle and hence modulation of the motor load DC voltage.

1. Upon a reload opportunity (defined by the PWM Load Frequency + Half Cycle reload setting), regardless of whether an actual reload occurs as determined by LDOK bit, the PWMF reload flag is set.
2. If the PWM reload interrupt enable bit, PWMRIE is set, the PWMF flag generates CPU interrupt requests allowing software to calculate new PWM Values in real time reflecting the application speed (or torque, or current) controller output.

Notes: When PWMRIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

3. Then the following steps must be performed (usually in the PWM reload ISR):
 - the PWMVALx registers (buffers) must be updated with new values previously calculated by SW (the PWMVAL0,2,4 reflecting the (EQ 5-1) and the PWMVAL1,3,5 reflecting the (EQ 5-2))
 - the PWMF reload flag must be acknowledged by reading PWM Control Register with PWMF set and then writing a logic zero to the PWMF bit
 - the load okay bit, LDOK, needs to be set by reading it when it is a logic zero and then writing a logic one to it. LDOK prevents reloading of these PWM parameters before software is finished calculating them.
4. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle.
5. After loading, LDOK is automatically cleared.

See [Figure 5-7](#) for signal waveforms.

The load okay bit, LDOK, enables loading the PWM generator with more new parameters such as:

- a prescaler divisor - from the PRSC1 and PRSC0 bits in PWM Control Register
- a PWM period - from the PWM counter modulus registers (PWMCM)
- a PWM pulse width - from the PWM value registers (PWMVALx)

This feature enables an on-fly variation of the PWM frequency - acoustic noise elimination technique already mentioned in section [Section 5.1.3](#).

5.2.1.3 Commutation

As explained, the “six step” commutation, corresponding to the actual rotor position, must be performed to create the rotational magnetic field driving the BLDC motors. The commutation event is critical for its angular (time) accuracy. Any deviation causes the torque ripples and hence the speed variation. While at low speed (low commutation frequency) the commutation can be performed by controlling the duty cycle - synchronously to the PWM cycle, at the high speed an asynchronous control of WM outputs is must.

The Motorola DSP56F80x family offers features for effective control of the commutation process by:

- PWM SW Output control of individual PWM outputs and
- PWM generator channel swapping

The commutation of the control signals can be performed immediately, without changing the content of the PWM value registers. If complementary mode is set the deadtime is properly inserted at PWM signal edges (see [Figure 5-7](#)).

Notes: Setting the OUTCTLx bits does not disable the PWM generators and current status sensing circuitry. They continue to run, but no longer control the output pins. When the OUTCTLx bits are cleared, the outputs of the PWM generator become the inputs to the deadtime generators at the beginning of the next PWM cycle. Software can drive the PWM outputs even when PWM enable bit (PWMEN) is set to zero.

Notes: During software output control, TOPNEG and BOTNEG still control output polarity.

The whole commutation process is independent from the voltage (speed) control of the BLDC application. It consists of the following steps.

1. **Obtaining the rotor position.** Usually three Hall effect sensors are used. The commutation algorithm detects the change of Hall sensors state. These sensors put together six states, where each state determines which motor phases are powered by 3-phase power stage.
2. **Preparing the commutation.** The state of the three Hall effect sensors is used as a pointer in the commutation look up table. This pre-prepared table associates the sensors states with the values of the SW output control bits OUTCTLx (PMOUT) and the values of the PWM generator swapping bits SWPxx (PMCCR).
3. **Advancing-delaying the commutation event.** The commutation does not always need to be performed when the edge from position sensors is detected. For example: advancing the commutation event enables to drive BLDC motor at higher speed, but sacrifices optimal efficiency and low torque ripples.
4. **Performing the commutation.** New values of the SW output control bits OUTCTLx and the PWM generator swapping bits SWPxx are loaded to corresponding PWM module registers (see [Figure 5-7](#)). Performing the above actions periodically generates the commutation sequence, hence the rotational magnetic field. More details about commutation table and driving the BLDC motor using the Hall effect sensors can be found in [Section 7](#).

Preliminary Copy

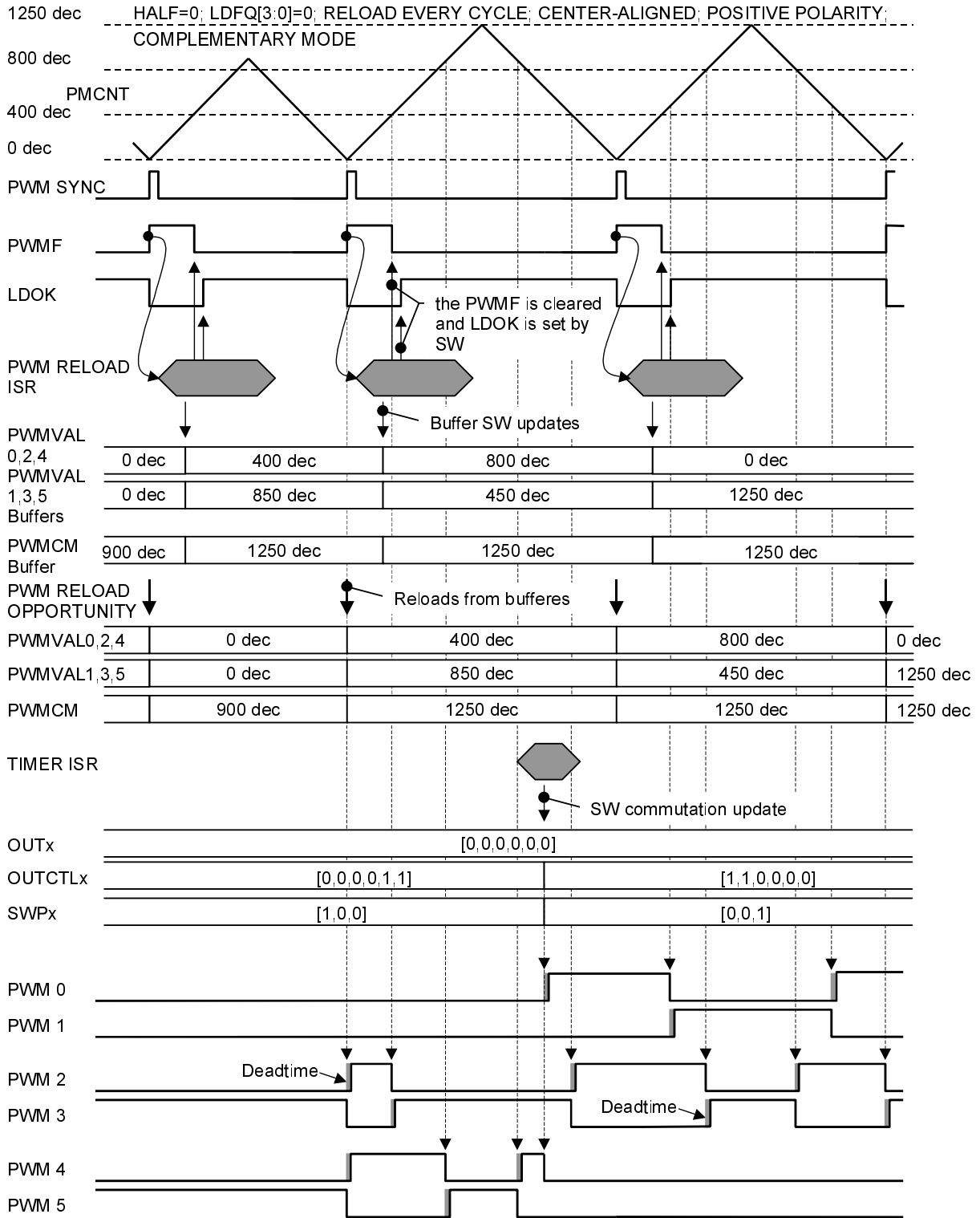


Figure 5-7. Cycle-by-cycle Modification of PWM Waveform

5.2.2 PWM for Back-EMF Detection BLDC Motor Control

The BLDC motor theory (provided in [Section 7](#). [8.] and [9.]) shows that the Back-EMF voltage (induced into the stator windings by movement of the permanent magnets mounted on the rotor) can be used for rotor position recognition.

The Back-EMF sensing technique is based on the fact that only two phases of a BLDC motor are connected at a time, so the third phase can be used to sense the Back-EMF voltage. There are two conditions which have to be met:

- the bipolar PWM must be produced (see [Figure 3-5](#)) to keep motor load voltage always $\pm U$
- no current is going through the phase used to sense the Back-EMF

The Back-EMF sensing technique has some limits. This is caused by the fact that the amplitude of the induced voltage is proportional to the motor speed. Hence, the Back-EMF cannot be sensed at a very low speed. Such technique is suitable for unidirectional drives (compressors, fans, blowers etc.) working just in one of four quadrants.

The needed bipolar PWM can be generated by driving the top and bottom switch (in diagonal) by the same PWM signal (independent operation) - the motor phase inductance then opens the anti parallel diodes in other diagonal during the turn OFF time. The required DC voltage to be created by PWM then corresponds to $U \times F_{PWM}$ when the PWM duty cycle is controlled using the (EQ 3-5).

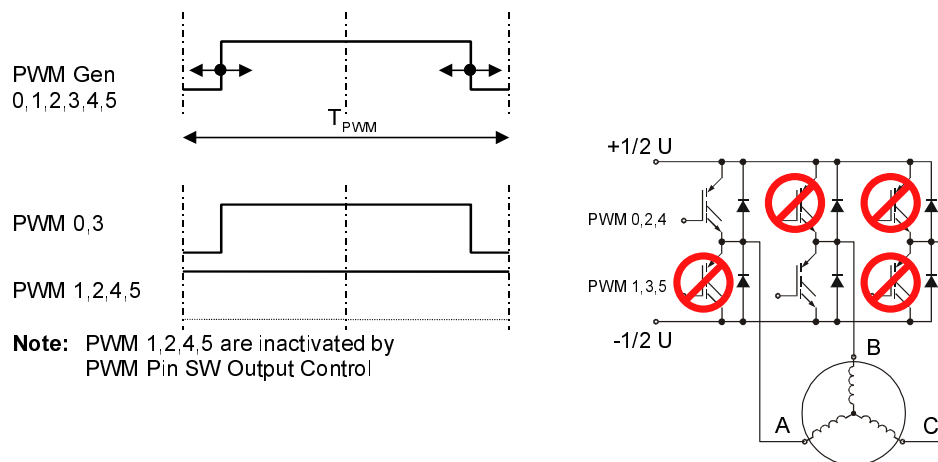


Figure 5-8. Bipolar PWM for Unidirectional Control

Notes: Driving the diagonal power switches by the same PWM signal allows to reverse applied voltage but not the motor current (torque).

The center or edge aligned PWM settings are optional. Usually the center-aligned PWM is the choice for easier ADC synchronization to center of PWM pulse (PWM reload).

The feature, accelerating the duty cycle setting, is the hardware acceleration, where writing to PWM value register 0 to also writes to PWM value registers 1 to 5.

Thus the commutation of the PWM control signals can be performed immediately without changing the content of the PWM value registers using the PWM SW Output control of individual PWM outputs (changes made to PWMVALx registers take effect at reload opportunity - defined by LDFQ[0:3], HALF). The commutation is then asynchronous to the PWM reload period.

The PWM frequency must be set high enough to prevent discontinuous current operation (at low currents). And it is always a compromise between audible noise, electromagnetic emissions and power switching losses.

5.2.2.1 BLDC Back-EMF PWM Static Setting

The following setting is used to initialize the module for classical BLDC motor control:

- **Independent Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to one in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - based on used hardware the polarity for top and bottom power switches needs to be set in the PMCFG accordingly.
- **Center-aligned PWM** (EDG) - bit is cleared (reset state) in the PMCFG.
- **Deadtime** (PMDEADTM) - no deadtime generation is needed because of the independent mode.
- **PWM Clock Prescaler** (PRSC [1:0]) - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency** (LDFQ[0:3]) + **Half Cycle reload** (HALF) - this setting strongly depends on the application requirements. For standard control the reload every PWM opportunity or every 2 PWM opportunities is often used, this results in 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.
- **Deadtime Correction Method**
 - Current Status Bits (ISENS[1:0]) - must be set to 0 for no correction, see Table 4-2.
 - Current Polarity Bits (IPOLx) - must be set to 0 or the reset state can be left.
- **PWM Frequency, Modulus** (PWMCM) - setting must reflect the required PWM frequency. Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.
- **PWM Pins SW Output Control** (PMCCR) - Output Pad Enable (PAD_EN = 1). Output control is needed to provide commutation means. Set the OUTx = 0 for PWM output pins deactivation. Set the OUTCTLx to one of the six vectors - [0,1,1,0,1,1]; [1,0,1,1,0,1]; [1,1,0,1,1,0]; [1,0,0,1,1,1]; [0,1,1,1,1,0]; [1,1,1,0,0,1] - to drive the diagonal power switches and to deactivate one of the three phases reflecting the initial commutation stage (rotor position).
- **PWM Channel Control** - PWM channel control is needed, ENHA = 1,
 - MSKx = 0 no masked PWM generator channels,
 - no PWM generator channel swapping - SWPx = [0,0,0] (reset state),
 - VLMODE [1:0] = 01, writing to PWM Value Register 0 also writes to PWMVAL 1 to 5.
- **Faults** -
 - **PWM Disable Mapping** (PMDISMAP1-2) - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#) The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.

There is no one “magic” setting satisfying all needs. Mostly the Fault 2 is connected to the DC-bus over-current detection circuitry, the Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is that both faults disable all PWMs and the Fault 0,1 are inactive. The PWM disable mapping bits:
 DISMAP2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22 and 23 = 1 other DISMAPx bits = 0.

- **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an application must perform some more sophisticated checking before enabling PWM again, the manual fault clearing should be selected.
- **FAULT Interrupt Enable (FIEx)** - setting bits 2 and 3 in PMFCTL enables CPU interrupt requests generated by the FAULT2,3 pins. Usually the fault interrupt is enabled when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.
- **Write-protection** - If required the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:
 - PMCFG - the PWM polarity, the independent PWM pair operation mode and center aligned PWM channels
 - PMDEADTM - the deadtime - not used
 - PMDISMAP1-2 - the PWM fault disable mapping matrix
 - PMCCR - ENHA = 1 - the enabled HW acceleration features (Value Register Load Mode - VLMODE can be modified)

The functions which are still available follow: PWM Value setting, PWM Frequency setting, PWM Clock prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting (not suitable), odd/even PWMVALx selection (not suitable), Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control, PWM generator channels masking and swapping (not suitable).

- **PWM Value Registers (PWMVALx)** - usually all duty cycles are initially set to 0% - producing no current. 50% duty cycle would produce “natural zero” level (refer to [Figure 3-5](#)), while it helps to keep charging the bootstrap circuitry in the power switch drivers. The PWM Value calculation is as follows:

$$\text{PWM value } x = (\text{duty cycle } x) / 100\% \times (\text{PWM modulus})$$

$$\text{PWMVAL}_x = 50\% / 100\% \times 1250 \text{ dec} = 625 \text{ dec for previously selected PWM modulus.}$$

The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.

- **PWM Reload Interrupt Enable (PWMRIE)** - this bit enables the PWMF flag to generate CPU interrupt requests. See also the notes below.
- **PWM Enable (PWMEN)** - this bit enables the PWM generator and the PWM pins. It should be set at the end of whole initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state, therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.2.2.2 Voltage Control - Cycle-by-cycle Modification of PWM

The MC PWM Module is initialized to generate independent, center-aligned PWM waveforms with given duty cycle and PWM frequency.

As described in previous sections, the PWM duty cycle is periodically updated according to the speed (or torque, or current) controller output. Such a controller calculates new values of the PWM duty cycles (PWMVALx). The hardware acceleration feature enables concurrent update of the all PWMVAL registers just by writing to the PWMVAL0.

Performing the following steps on regular basis (each reload) leads to modification of the PWM duty cycle and hence modulation of the motor load DC voltage.

1. Upon a reload opportunity (defined by the PWM Load Frequency + Half Cycle reload setting), regardless whether an actual reload occurs as determined by LDOK bit, the PWMF reload flag is set.
2. If the PWM reload interrupt enable bit, PWMRIE is set, the PWMF flag generates CPU interrupt requests allowing software to calculate new PWM Values in real time reflecting the application speed (or torque, or current) controller output.

Notes: When PWMRIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

3. Then the following steps must be performed (usually in the PWM reload ISR):
 - the PWMVAL0 register (buffer) must be updated with new values previously calculated by SW (PWMVAL1-5 buffers are updated automatically)
 - the PWMF reload flag must be acknowledged by reading PWM Control Register with PWMF set and then writing a logic zero to the PWMF bit
 - the load okay bit, LDOK, needs to be set by reading it when it is a logic zero and then writing a logic one to it. LDOK prevents reloading of these PWM parameters before software is finished calculating them.
4. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle.
5. After loading, LDOK is automatically cleared.

See [Figure 5-9](#) for signal waveforms.

The load okay bit, LDOK, enables loading the PWM generator with more new parameters, such as:

- a prescaler divisor - from the PRSC1 and PRSC0 bits in PWM Control Register
- a PWM period - from the PWM counter modulus registers (PWMCM)
- a PWM pulse width - from the PWM value registers (PWMVALx)

This feature enables an on-fly variation of the PWM frequency - acoustic noise elimination technique already mentioned in section [Section 5.1.3](#).

5.2.2.3 Commutation

The “six step” commutation of the PWM control signals for the Back-EMF voltage sensing techniques is implemented similarly as in case of the classical BLDC motor control. The PWM SW Output control of individual PWM outputs, feature offered by Motorola DSP56F80x family, is employed, so changing the content of the OUTCTLx bits provides immediate PWM signals commutation.

Notes: Setting the OUTCTLx bits do not disable the PWM generators and current status sensing circuitry. They continue to run, but no longer control the output pins. When the OUTCTLx bits are cleared, the outputs of the PWM generator become the inputs to the deadtime generators at the beginning of the next PWM cycle. Software can drive the PWM outputs even when PWM enable bit (PWMEN) is set to zero.

Notes: During software output control, TOPNEG and BOTNEG still control output polarity.

The whole commutation process is independent from the voltage (speed) control of the BLDC application. It consists of the following steps.

1. **Obtaining the rotor position.** The position recognition algorithms detects the rotor position based on the sensed Back-EMF. The rotor position determines which motor phases are powered by 3-phase power stage.
2. **Preparing the commutation.** When the right position of the rotor is determined. The pointer in the commutation look up table is updated (incremented) to read new values of the SW output control bits OUTCTLx (PMOUT).
3. **Advancing-delaying the commutation event.** The commutation does not always need to be performed when the edge from position sensors is detected. For example: advancing the commutation event enables to drive BLDC motor at higher speed, but sacrifices the optimal efficiency and low torque ripples.
4. **Performing the commutation.** New values of the SW output control bits OUTCTLx are loaded to corresponding PWM module registers (see [Figure 5-9](#)).

Performing the above actions periodically generates the commutation sequence, hence the rotational magnetic field. More details about driving the BLDC motor using the Back-EMF zero crossing detection can be found in [Section 7](#). [8.] and [9.].

Preliminary Copy

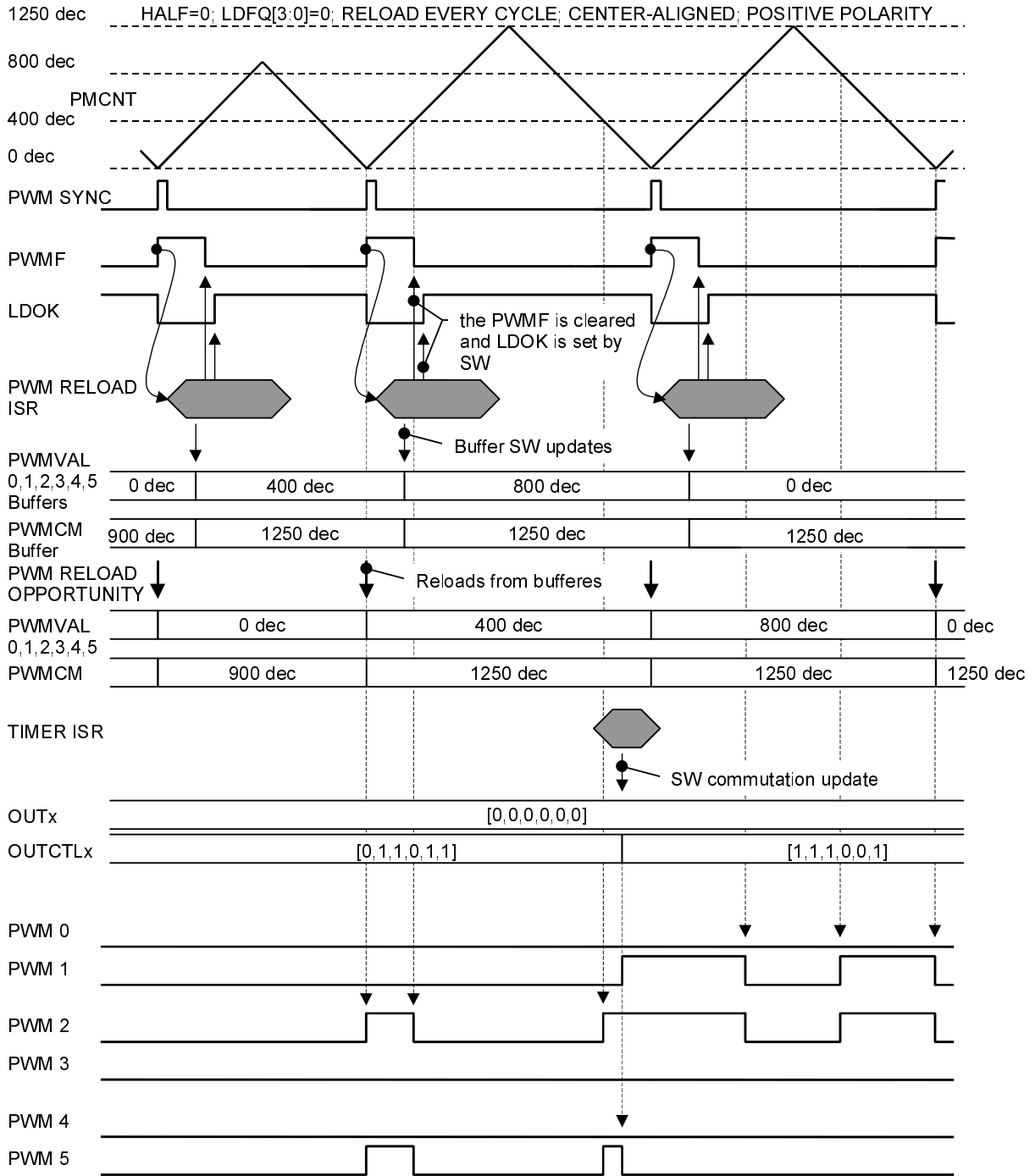


Figure 5-9. Cycle-by-Cycle Modification of PWM Waveform

5.3 SR Motor

The SR motor is controlled by non-sinusoidal, unipolar phase voltage strokes coupled with the rotor position. A profile of the phase current defines the generated torque and thus the speed of the motor. Due to this fact the motor requires control electronic for the operation. Several power stage topologies are being implemented. The SR power stage structure defines the freedom of controlling of the individual phases.

The widest freedom of control gives the power stage with two independent power switches per phase. It enables to control the individual phases fully independently on each other. **Figure 5-10** illustrates a 3-phase SRM topology. Other possible topologies share some of the power devices for more phases, saving the power stage cost but decreasing control freedom. The unique topology of the SR power stage, that differs from the AC ones, makes it fault tolerant because it eliminates the possibility of rail-to-rail short circuit, thus eliminating the need for the deadtime insertion logic.

The other advantage is based on the fact that the torque produced by SRM does not depend on the phase current polarity. Therefore only unipolar voltage and current operation is needed, which leads to one quadrant PWM generation. This should not be understood incorrectly, the full four quadrant operation (motoring or generating mode in both directions) of the SRM drive is still possible by applying the voltage strokes at different rotor positions. For more details on how to drive the SRM see **Figure 7**. [10.].

The control of the SRM can be split onto two independent processes:

- commutation - sequentially activates one or more motor phases
- voltage control - DC voltage applied onto motor phase is controlled by speed, current or torque controller

The phase commutation process is usually serviced in the timer interrupt service routine. The time of initiated interrupts then reflects the position signal or the elapsed time (angular phase-shift) from position signal.

The voltage controller (control loop) is calculated regularly with given period. In most cases, this period matches the PWM reload period.

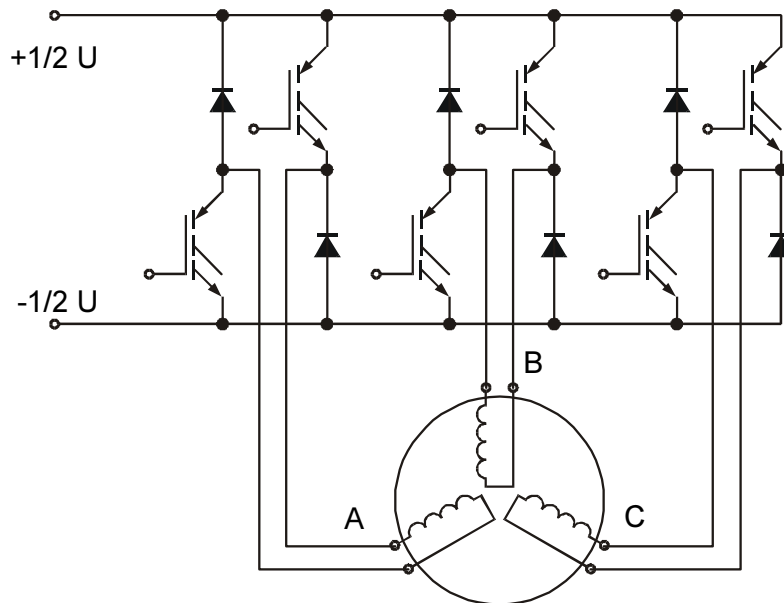


Figure 5-10. SRM 3-ph Power Stage

5.3.1 PWM for Classical SRM

The SR motors differ in many aspects. The number of phases wound on the stator can be seen as a main one. Each number of phases have a suitable combination of stator and rotor poles numbers. The presented PWM operation is based on the most popular 3-phase SRM with configuration 6/4 of stator/rotor poles.

Two general SRM phase PWM control method can be distinguished.

- The **individual phase PWM operation** - is based on the power stage topology shown in [Figure 5-10](#) allowing the highest flexibility of the SRM phases control. The control of motor phases can overlap since the phase PWM turn ON times, the phase PWM turn OFF times and the PWM duty cycles are individually controlled.

Most of the advanced SRM control techniques which minimize torque ripples require individual phase PWM operation. Each phase current is separately modulated by a special shape leading to “clean” total mechanical torque, while the torques produced by individual phases do not necessarily need to be ripple free.

- The **combined phase PWM operation** - the control of PWM of different phases never overlaps. After one phase is switched OFF the other one is turned ON, thus the PWM duty cycle can be just “multiplexed” between the motor phases. This operation is similar to one described for BLDC motor control and it is used for lower-end SRM control.

Note that the combined PWM operation limit the control freedom. The advantage is that some of power switches can be shared for more phases, saving the power stage cost.

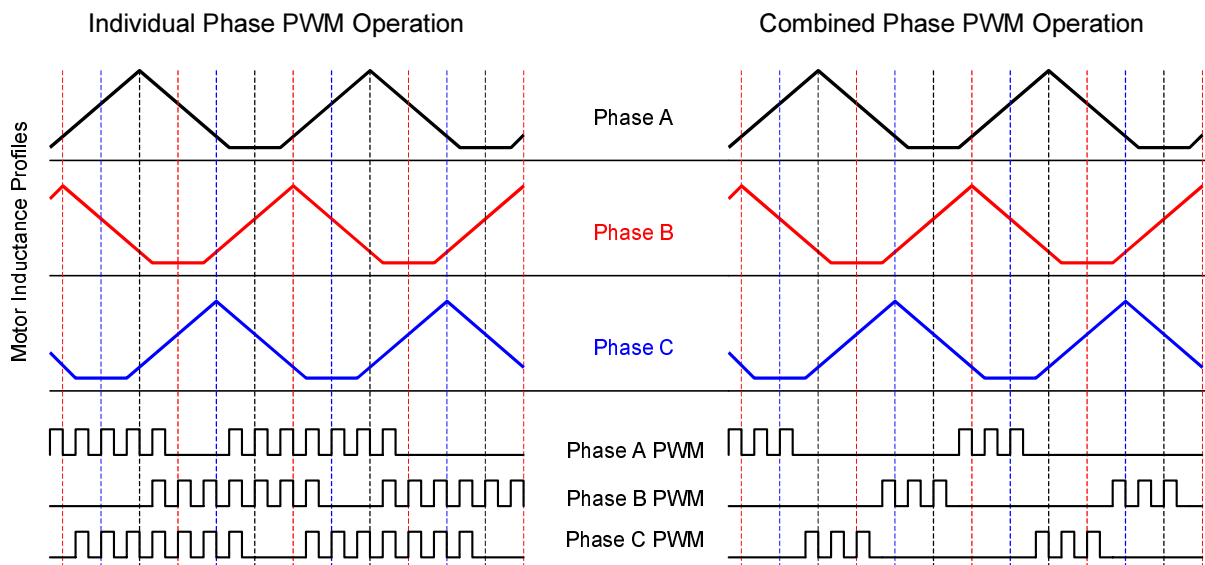


Figure 5-11. SRM Phase PWM Operation Methods

The MC PWM module supports both methods of phased PWM operation. In the independent operation mode each PWM can be individually controlled, while the PWM SW Output control of individual PWM outputs allows easy activation and deactivation of each PWM signal. Thus, the commutation of the PWM control signals can be performed immediately (asynchronously to the PWM reload), without changing the content of the PWM value registers.

Notes: During software output control, TOPNEG and BOTNEG still control output polarity.

One of the main SRM advantages is the simple “single quadrant” PWM operation for any of the four quadrants of the SRM drive operation (refer to [Section 7](#). [10.]). Therefore the deadtime insertion is not needed.

Even the unipolar PWM switching mode is a natural choice, the bipolar PWM switching could be used, in principal. But the higher rms value (losses) and higher electromagnetic emissions usually disqualify it (refer to [Section 3.3.1](#)).

Similarly, the center-aligned PWM mode is used rather than the edge-aligned one, although both modes can generate the unipolar PWM switching. The center-aligned PWM mode simply offers better electromagnetic emission figures (as explained in [Section 3.3.2](#)).

Figure 5-12 demonstrates two possible PWM patterns, both generating unipolar PWM switching.

- In **case A**, the top power switch is driven by PWM signal while the bottom power switch is kept ON for whole phase operation. As can be easily understood, the same motor phase voltage will be generated if the PWM signal drives the bottom and the ON signal drives the top switch. Because the bootstrap technique, usually used by top power switch drivers, there is a need for “recharging” the bootstrap circuit - switching ON and OFF the top switch. Therefore the PWM signal drives mostly the top switch.

The motor load voltage is always positive (operation in quadrant I). Its mean value can be calculated as described in [Section 3.3.2](#) - cases 1. or 2. If one switch (e.g. upper one) is driven by PWM signal with duty cycle $d_c = F_{PWM}$ and the other switch (e.g. lower one) is always ON then the mean motor load voltage u^* is:

$$u^* = U \cdot F_{PWM} > 0 \quad (\text{EQ 5-3})$$

The advantage, very simple PWM signal generation, is also the main disadvantage, which is an uneven distribution of the power losses of the power devices during the PWM cycle (e.g. the top diode is not conducting any current).

- **Case B** addresses the even distribution of the power losses. Both switches are driven by PWM signal.

This unipolar PWM type allows “two quadrant” operation (QI. and QII.). The motor current is always positive while motor phase load voltage can be controlled from full-negative to full-positive range. Consequently, the $\Delta i/T_{PWM}$ of motor phase current is then also controlled from full-negative to full-positive range. Thus, better phase current shape control is possible in advanced applications such as torque ripple elimination.

If the power switch PWM duty cycles d_{ctop} and $d_{cbottom}$ are:

$$d_{ctop} = d_{cbottom} = \frac{1}{2} \cdot F_{PWM} + \frac{1}{2} \quad \text{where: } -1 \leq F_{PWM} \leq 1 \quad (\text{EQ 5-4})$$

then the mean motor load voltage u^* is:

$$u^* = U \cdot (d_{ctop} - 1 + d_{cbottom}) = U \cdot F_{PWM} \quad (\text{EQ 5-5})$$

The motor phase load voltage PWM frequency is double that of the PWM control signal frequency which decreases the PWM phase current ripples. Both switches and both diodes share the load current during one PWM cycle which balances the switching and conductive losses.

Notes: The individual phase PWM operation differs from combined one just in individual setting of the PWMVALx registers and in different strategy of the PWM Pins SW Output Control.

Case B PWM generation can be implemented on the MC PWM module by setting different PWM polarity for top and bottom PWM pins.

Choosing the PWM frequency is always a compromise between audible noise, electromagnetic emissions, current ripples and power switching losses.

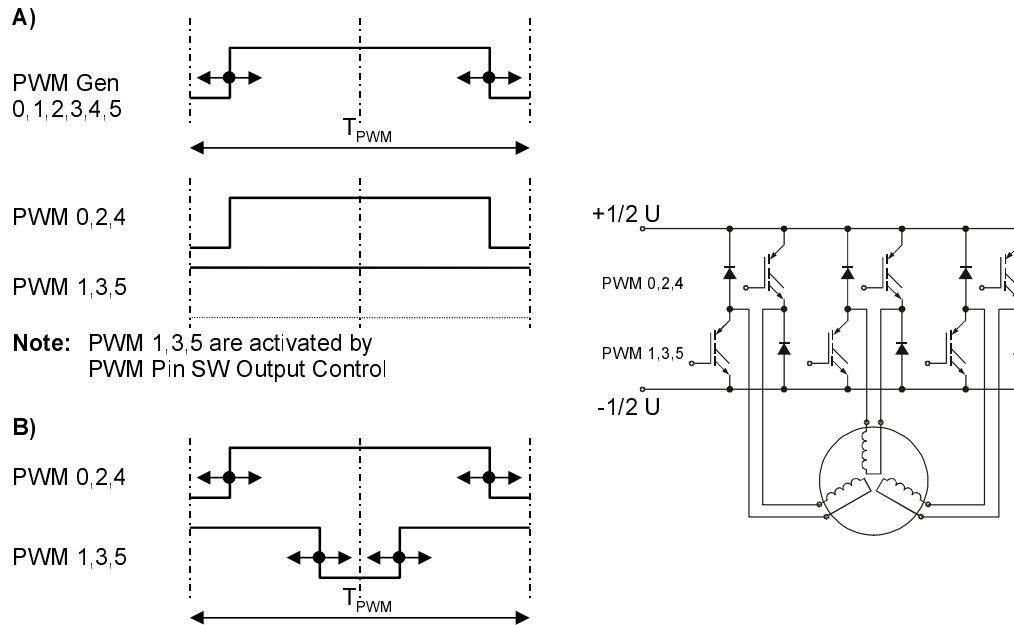


Figure 5-12. PWMs for SRM

5.3.1.1 SRM PWM Static Setting - Case A

The following setting is used to initialize the module for SRM control. A different setting for combined phase PWM operation and the independent phase PWM operation is also provided.

- **Independent Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to one in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - based on used hardware the polarity for top and bottom power switches needs to be set in the PMCFG accordingly.
- **Center-aligned PWM** (EDG) - bit is cleared (reset state) in the PMCFG.
- **Deadtime** (PMDEADTM) - no deadtime generation is needed because of the independent mode.
- **PWM Clock prescaler** (PRSC [1:0]) - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency** (LDFQ[0:3]) + **Half Cycle reload** (HALF) - this setting strongly depends on the application requirements. For high speed applications half cycle reload feature can be used while standard control often uses reload every PWM opportunity or every 2 PWM opportunities, this results in 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.
- **Deadtime Correction Method**
 - Current Status Bits (ISENS[1:0]) - must be set to 0 for no correction, see [Table 4-2](#)
 - Current Polarity Bits (IPOLx) - must be set to 0 or the reset state can be left.

- **PWM Frequency, Modulus (PWMCM)** - setting must reflect the required PWM frequency. Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$
 16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.
- **PWM Pins SW Output Control (PMCCR)** - Output Pad Enable (PAD_EN = 1). Output control is needed to provide commutation means.
 Set the OUTx and OUTCTLx to one of the three vector pairs - [0,0,0,0,1,0],[1,1,1,1,1,0]; [0,0,1,0,0,0],[1,1,1,0,1,1]; [1,0,0,0,0,0],[1,0,1,1,1,1] - to drive one phase and to deactivate two other phases (combined phase PWM operation) while reflecting the initial commutation stage (rotor position).
 The individual phase PWM operation can be easily achieved by individual SW output control of PWM pins.
 Note that setting the OUTx bit to 1 activates and setting it to 0 deactivates the PWM output pin if the corresponding OUTCTLx is set to one. Otherwise the PWM control signal is output from the PWM pin.
 During software output control, TOPNEG and BOTNEG still control output polarity.
- **PWM Channel Control** - PWM channel control is needed, ENHA = 1,
 - MSKx = 0 no masked PWM generator channels,
 - no channel swapping - SWPx = [0,0,0] (reset state),
 - VLMODE [1:0] = 01, writing to PWM Value Register 0 also writes to PWMVAL 1 to 5 - combined phase PWM operation. The PWM generator outputs six PWM channel with the same PWM duty cycle. The commutation control is then provided through the PWM Pins SW Output Control function.
 VLMODE [1:0] = 00, each PWM Value Register is accessed independently - individual phase PWM operation.
- **Faults** -
 - **PWM Disable Mapping (PMDISMAP1-2)** - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#) The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.
 There is no one “magic” setting satisfying all needs. Mostly the Fault 2 is connected to the DC-bus over-current detection circuitry, the Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is that both faults disable all PWMs and the Fault 0,1 are inactive. The PWM disable mapping bits:
 DISMAP2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22 and 23 = 1 other DISMAPx bits = 0.
 - **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an application must perform some more sophisticated checking before enabling PWM again the manual fault clearing should be selected.
 - **FAULT Interrupt Enable (FIEx)** - setting bits 2 and 3 in PMFCTL enables CPU interrupt requests generated by the FAULT2,3 pins. Usually the fault interrupt is enabled when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.

- **Write-protection** - If required the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:
 - PMCFG - the PWM polarity, the independent PWM pair operation mode and center aligned PWM channels
 - PMDEADTM - the deadtime - not used
 - PMDISMAP1-2 - the PWM fault disable mapping matrix
 - PMCCR - ENHA = 1 - the enabled HW acceleration features (Value Register Load Mode - VLMODE can be modified)

The functions which are still available follow: PWM Value setting, PWM Frequency setting, PWM Clock prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting (not suitable), odd/even PWMVALx selection (not suitable), Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control, PWM generator channels masking and swapping (not suitable).

- **PWM Value Registers** (PWMVALx) - usually all duty cycles are set to 0% - producing no current. 0% duty cycle produces “natural zero” level.

The PWM Value calculation is as follows:

$$\text{PWM value } x = (\text{duty cycle } x) / 100\% \times (\text{PWM modulus})$$

$$\text{PWMVAL}_x = 50\% / 100\% \times 1250 \text{ dec} = 625 \text{ dec for previously selected PWM modulus.}$$

The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.

- **PWM Reload Interrupt Enable** (PWMRIE) - this bit enables the PWMF flag to generate CPU interrupt requests. See also the note sbelow.
- **PWM Enable** (PWMEN) - this bit enables the PWM generator and the PWM pins. It should be set at the end of whole initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state, therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.3.1.2 SRM PWM Static Setting - Case B

The following setting is used to initialize the module for SRM control. A different setting for combined phase PWM operation and the independent phase PWM operation is described in the text.

- **Independent Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to one in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - different polarity for top and bottom power switches needs to be set (in the PMCFG) based on used hardware, e.g. TOPNEG[2:0]=[0,0,0], BOTNEG[2:0]=[1,1,1].
- **Center-aligned PWM** (EDG) - bit is cleared (reset state) in the PMCFG.

- **Deadtime (PMDEADTM)** - no deadtime generation is needed because of the independent mode.
- **PWM Clock prescaler (PRSC [1:0])** - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency (LDFQ[0:3]) + Half Cycle reload (HALF)** - this setting strongly depends on the application requirements. For high speed applications half cycle reload feature can be used while standard control often uses reload every PWM opportunity or every 2 PWM opportunities, this results in 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.
- **Deadtime Correction Method**
 - Current Status Bits (ISENS[1:0]) - must be set to 0 for no correction, see [Table 4-2](#)
 - Current Polarity Bits (IPOLx) - must be set to 0 or the reset state can be left.
- **PWM Frequency, Modulus (PWMCM)** - setting must reflect the required PWM frequency. Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.
- **PWM Pins SW Output Control (PMCCR)** - Output Pad Enable (PAD_EN = 1). Output control is needed to provide commutation means.

Example:

set the OUTx to [1,0,1,0,1,0] and OUTCTLx to one of the three vectors - [1,1,1,1,0,0], [1,1,0,0,1,1], [0,0,1,1,1,1] - to drive one phase and to deactivate two other phases (combined phase PWM operation) while reflecting the initial commutation stage (rotor position).

Individual phase PWM operation can be easily achieved by individual SW output control of PWM pins.

Note that setting the OUTx bit to 1 activates and setting it to 0 deactivates the PWM output pin if the corresponding OUTCTLx is set to one. Otherwise the PWM control signal is output from the PWM pin.

During software output control, TOPNEG and BOTNEG still control output polarity.

- **PWM Channel Control** - no channel control needed (HW acceleration disabled, no masked PWM generator channels, no swapped channel, each PWM Value Register is accessed independently), the PMCCR register can be left in its reset state.

Note that the commutation control is provided through the PWM Pins SW Output Control function.

- **Faults -**

- **PWM Disable Mapping (PMDISMAP1-2)** - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#) The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.

There is no one “magic” setting satisfying all needs. Mostly the Fault 2 is connected to the DC-bus over-current detection circuitry, the Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is that both faults disable all PWMs and

the Fault 0,1 are inactive. The PWM disable mapping bits:

DISMAP2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22 and 23 = 1 other DISMAPx bits = 0.

- **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an application must perform some more sophisticated checking before enabling PWM again the manual fault clearing should be selected.
- **FAULT Interrupt Enable (FIEx)** - setting bits 2 and 3 in PMFCTL enables CPU interrupt requests generated by the FAULT2,3 pins. Usually the fault interrupt is enabled when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.
- **Write-protection** - If required, the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:
 - PMCFG - the PWM polarity, the independent PWM pair operation mode and center aligned PWM channels,
 - PMDEADTM - the deadtime - not used,
 - PMDISMAP1-2 - the PWM fault disable mapping matrix,
 - PMCCR - ENHA = 0 - the disabled HW acceleration features (Value Register Load Mode - VLMODE, PWM generator channels swapping).

The functions which are still available follow: PWM Value setting, PWM Frequency setting, PWM Clock prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting (not suitable), odd/even PWMVALx selection (not suitable), Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control, PWM generator channels masking.

- **PWM Value Registers (PWMVALx)** - usually all duty cycles are initially set to 0% - producing no current. 50% duty cycle produces “natural zero” level.

The PWM Value calculation is as follows:

$$\text{PWM value } x = (\text{duty cycle } x) / 100\% \times (\text{PWM modulus})$$

$$\text{PWMVAL}_x = 50\% / 100\% \times 1250 \text{ dec} = 625 \text{ dec for previously selected PWM modulus.}$$

The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.

- **PWM Reload Interrupt Enable (PWMRIE)** - this bit enables the PWMF flag to generate CPU interrupt requests. See also the note below.
- **PWM Enable (PWMEN)** - this bit enables the PWM generator and the PWM pins. It should be set at the end of whole initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state, therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.3.1.3 Voltage Control - Cycle-by-cycle Modification of PWM

The MC PWM Module is initialized to generate independent, center-aligned PWM waveforms with given duty cycle and PWM frequency.

As it is described in previous sections the PWM duty cycle is periodically updated according to the speed (or torque, or current) controller output. Such a controller calculates the new values of the PWM duty cycles (PWMVALx). There are the following possibilities:

- In **case A - combined phase PWM operation** utilizing the hardware acceleration feature enables concurrent update of the all PWMVALx registers just by writing to the PWMVAL0.
- In **case A - individual phase PWM operation** the PWMVALx registers are individually accessed.
- In **case B** the PWMVALx registers are individually accessed for both the **individual** and the **combined** phase PWM operation.

The implementation of the PWM generation is, in this case, based on the different polarity setting for top and bottom PWM pins. Further explanation of duty cycle calculation assumes the following polarity settings: TOPNEGxx = 0 positive top-side polarity, BOTNEGxx = 1 negative bottom-side polarity.

Equations (EQ 5-4) and (EQ 5-5) describe the PWM control signal duty cycle calculation for the case B PWM generation, the actual duty cycle setting for PWMVALx register must take into consideration also the polarity setting. Therefore the top PWM Valx duty cycle $d_{ctopVal}$ is:

$$d_{ctopVal} = \frac{1}{2} + \frac{1}{2} \cdot F_{PWM} \text{ where: } -1 \leq F_{PWM} \leq 1 \quad (\text{EQ 5-6})$$

and the bottom switch PWM duty cycle $d_{cbottomVal}$ is:

$$d_{cbottomVal} = \frac{1}{2} - \frac{1}{2} \cdot F_{PWM} \quad (\text{EQ 5-7})$$

Then the mean motor load voltage u^* is:

$$u^* = U \cdot F_{PWM} \quad (\text{EQ 5-8})$$

The u^* can be negative or positive, but it is always unipolar within one PWM cycle.

Performing the following steps on regular basis (each reload) leads to modification of the PWM duty cycle and hence modulation of the motor load DC voltage.

1. Upon a reload opportunity (defined by the PWM Load Frequency + Half Cycle reload setting), regardless whether an actual reload occurs as determined by LDOK bit, the PWMF reload flag is set.
2. If the PWM reload interrupt enable bit, PWMRIE is set, the PWMF flag generates CPU interrupt requests allowing software to calculate new PWM Values in real time reflecting the application speed (or torque, or current) controller output.

Notes: When PWMRIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

3. Then the following steps must be performed (usually in the PWM reload ISR):

- the PWMVALx register (buffers) update with new values previously calculated by SW:
 - in **case A - combined phase PWM operation** the PWMVAL0 register (buffer) must be updated reflecting the (EQ 5-3) (PWMVAL1-5 buffers are updated automatically)
 - in **case A - individual phase PWM operation** the PWMVALx registers (buffers) must be updated individually reflecting the (EQ 5-3)
 - in **case B** the PWMVALx registers (buffers) must be updated such that the PWMVAL0,2,4 reflect the (EQ 5-6) and the PWMVAL1,3,5 reflect the (EQ 5-7)
 - the PWMF reload flag must be acknowledged by reading PWM Control Register with PWMF set and then writing a logic zero to the PWMF bit,
 - the load okay bit, LDOK, needs to be set by reading it when it is a logic zero and then writing a logic one to it. LDOK prevents reloading of these PWM parameters before software is finished calculating them.
4. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle.
 5. After loading, LDOK is automatically cleared.

Figure 5-13 and **Figure 5-14** show the control of the PWM signal waveforms for the case A - combined phase PWM operation and for the case B - combined phase PWM, respectively.

The load okay bit, LDOK, enables loading the PWM generator with more new parameters, such as:

- a prescaler divisor - from the PRSC1 and PRSC0 bits in PWM Control Register,
- a PWM period - from the PWM counter modulus registers (PWMCM),
- a PWM pulse width - from the PWM value registers (PWMVALx).

This feature enables an on-fly variation of the PWM frequency - acoustic noise elimination technique already mentioned in [Section 5.1.3](#).

5.3.1.4 Commutation

The commutation of the PWM signals over the motor phases is implemented similarly as in case of the BLDC motor control. The PWM SW Output control of individual PWM outputs, a feature offered by Motorola DSP56F80x family, is employed, so changing the content of the PWM value registers is not needed; hence the commutation happens instantly.

Notes: Setting the OUTCTLx bits do not disable the PWM generators and current status sensing circuitry. They continue to run, but no longer control the output pins. When the OUTCTLx bits are cleared, the outputs of the PWM generator become the inputs to the deadtime generators at the beginning of the next PWM cycle. Software can drive the PWM outputs even when PWM enable bit (PWMEN) is set to zero.

Notes: During software output control, TOPNEG and BOTNEG still control output polarity.

The whole commutation process consists of the following steps.

1. **Obtaining the rotor position.** The position recognition algorithms detects the rotor position based on the position sensor output or the estimator output (sensorless application). The rotor position determines which phases are powered by 3-phase power stage.
2. **Preparing the commutation.** When the right position of the rotor is determined, the pointer in the commutation look up table is updated (incremented) to read new values of the SW output control bits OUTCTLx (PMOUT).

3. **Advancing-delaying the commutation event.** The commutation is not always needed to be performed at the position detected by sensor or estimator. For example: advancing the commutation event enables to build more current in the SRM phase so the motor can run at higher speed, but produces high torque ripples.
4. **Performing the commutation.** New values of the SW output control bits OUTCTLx are loaded to corresponding PWM module registers (see [Figure 5-13](#) and [Figure 5-14](#)).

Performing the above actions periodically, generates the commutation sequence, hence the rotational magnetic field. More details about driving the SRM can be found in [Section 7](#). [10].

Preliminary Copy

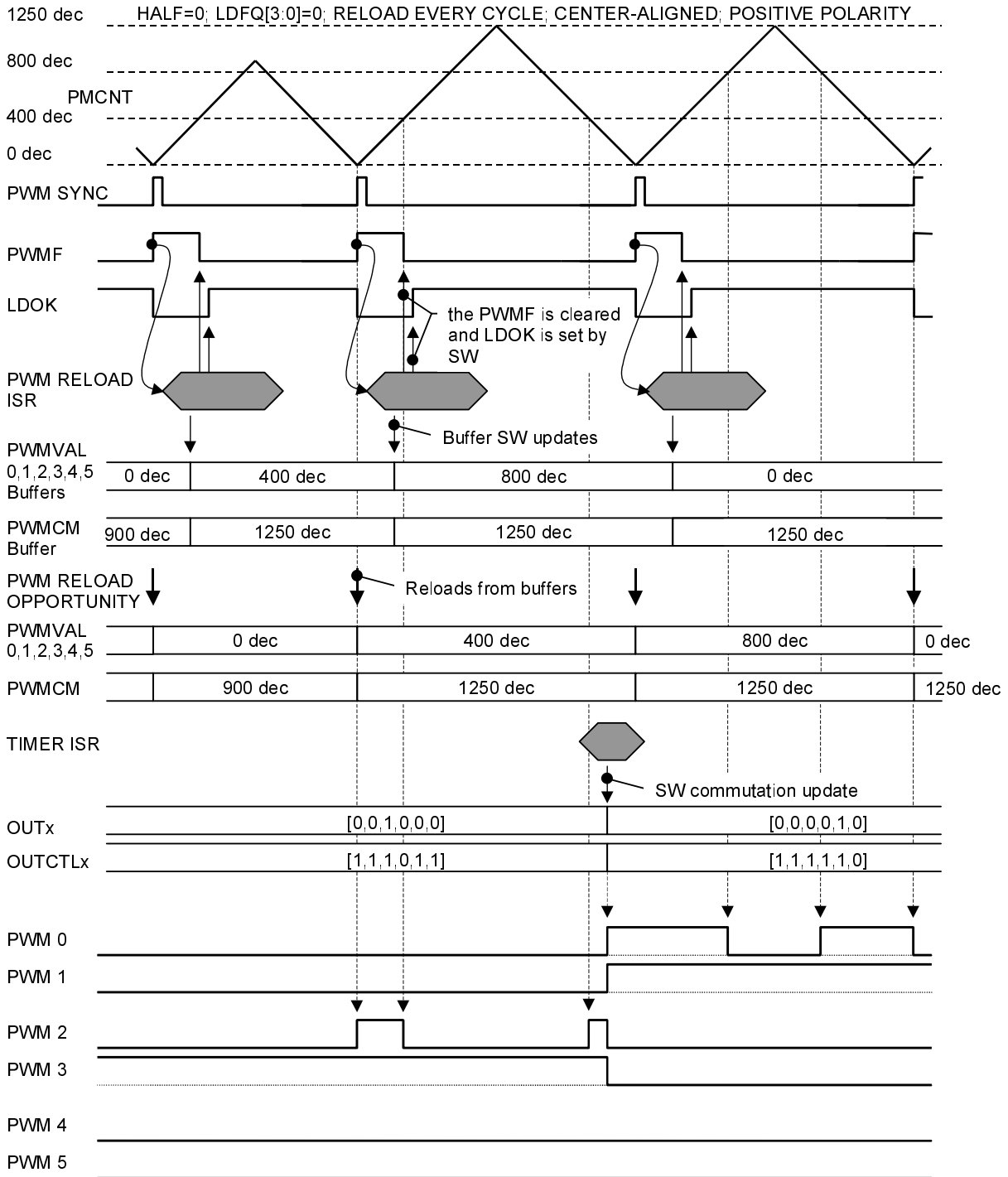


Figure 5-13. SRM - Case A - PWM Waveforms

Preliminary Copy

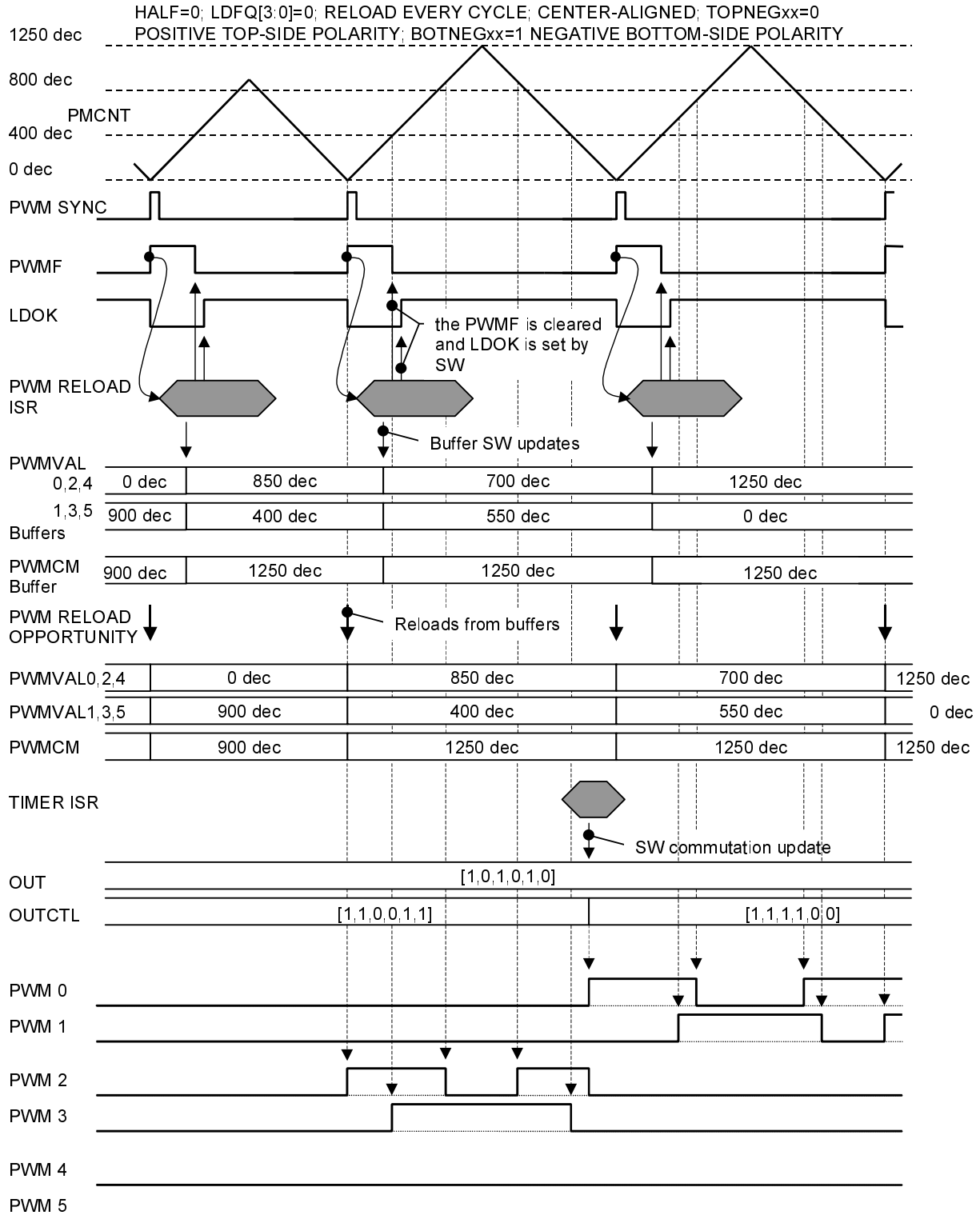


Figure 5-14. SRM - Case B - PWM Waveforms

Preliminary Copy

5.4 DC Brushed Motor

The DC Brushed motor power stage topology is a classical “full bridge” as shown in [Figure 5-15](#). The theoretical background for this power stage topology is given in [Section 3.2](#) and [Section 3.3](#).

The DC Brushed motor is driven by the “DC” voltage source. A rotational field is created by means of commutator and brushes on the motor. Although the “end of life” of DC Brushed motor drives was forecast several times, in reality these drives are still very popular and vital. This is mainly because control is very simple, and no sophisticated calculations and algorithms such as commutation, waveform generation, space vector modulation, etc. are required.

Usually the controller consists of the speed control loop with inner current (torque) control loop. The inner loop controls DC voltage applied onto the motor winding. The control loop is calculated regularly with given period. In most cases, this period matches the PWM reload period.

Driving the DC motor from DC voltage source (power stage shown in [Figure 5-15](#)), the motor can easily work in all four quadrants. The complementary mode of operation with deadtime insertion is needed for smooth reversal of the motor current (motor torque), hence smooth full four quadrant control. Other settings, like center or edge aligned PWM, are optional. Usually the center-aligned (therefore unipolar) PWM is chosen to lower electromagnetic emissions.

The PWM frequency selection is always a compromise between audible noise, electromagnetic emissions, current ripples and power switching losses. It will be different for industrial, appliance or automotive environment.

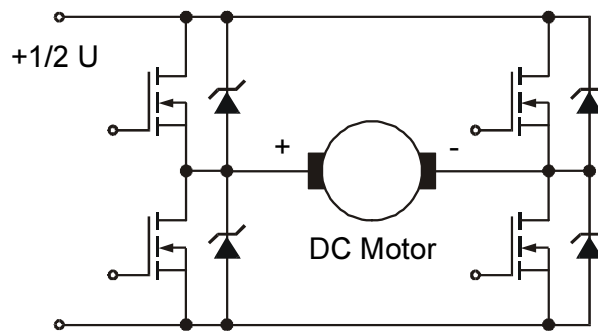


Figure 5-15. DC Brushed Motor Power Stage

5.4.1 PWM for DC Brushed Motor

As it is described in [Section 3.3](#), there are a number of possible PWM types which differ in the switching sequence of power switches and all can produce comparable results. Detailed requirements need to be known in order to choose suitable PWM for given DC motor drive application.

This section focuses on the most-used type of PWM generation which allows DC motor control in motoring and generating modes at any speed direction with smooth torque and speed reversal. The further explanation assumes that the complementary mode with the deadtime insertion and the center-aligned (therefore unipolar; refer to [Section 3-6](#) for PWM signal shapes) PWM is chosen.

The shape of the PWM signal, as shown in [Section 5-16](#), can be easily implemented using the MC PWM module. The PWMVAL0 and PWMVAL2 needs to be loaded in accordance to the equations (EQ 5-9) and (EQ 5-10), respectively.

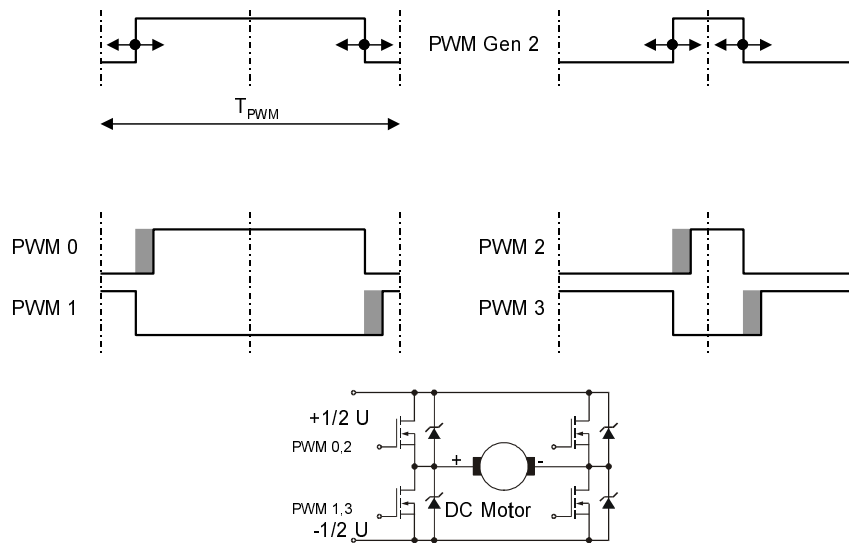


Figure 5-16. PWM for DC Motor

The PWM duty cycle of the top switch 0 (PWMVAL0) defines voltage at the first power stage “branch”:

$$d_{cSW0} = \frac{1}{2} + \frac{1}{2} \cdot F_{PWM} \quad \text{where: } -1 \leq F_{PWM} \leq 1 \quad (\text{EQ 5-9})$$

The PWM duty cycle of top switch 2 (PWMVAL2) defines voltage at the second power stage “branch”:

$$d_{cSW2} = \frac{1}{2} - \frac{1}{2} \cdot F_{PWM} \quad (\text{EQ 5-10})$$

This is, so called, symmetrical voltage operation, as described in [Section 3.3.2](#).

The mean motor load voltage u^* is:

$$u^* = U \cdot F_{PWM} \quad (\text{EQ 5-11})$$

The u^* can be negative or positive, but it is always unipolar within one PWM cycle.

Because the PWM channels work in the complementary, mode the pre-programmed deadtime is automatically inserted whenever top or bottom switch acts. The deadtime has to be inserted to provide enough switch OFF relaxation time to avoid overlap of conducting interval between the top and bottom switch. Thus it provides protection of the power stage from short circuit.

5.4.1.1 DC Brushed Motor PWM Static Setting

The following setting is used to initialize the module for classical DC motor control:

- **Complementary Operation Mode** (INDEP[2:0]) - these write-protectable bits must be set to zero (reset state) in the PWM Configure Register (PMCFG).
- **PWM Polarity** (TOPNEG[2:0], BOTNEG[2:0]) - based on used hardware the polarity for top and bottom power switches needs to be set in the PMCFG register accordingly.

- **Center-aligned PWM (EDG)** - bit is cleared (reset state) in the PMCFG.
- **Deadtime (PMDEADTM)** - it needs to match the switching characteristics of used power switches. Usual value fits between 1.5-3 usec => PWMDT = 60-120 dec @ 25 nsec PWM clock period.

Note that the deadtime is defined as a number of PWM clock cycles and is affected by changes of the prescaler value.

- **PWM Clock prescaler (PRSC [1:0])** - usually the prescaler is set (in PMCTL) to divide by 1 to obtain the best PWM resolution.
- **PWM Load Frequency (LDFQ[0:3]) + Half Cycle reload (HALF)** - this setting strongly depends on the application requirements. For fast servo application half cycle reload feature is often used, while standard control uses reload every PWM opportunity or every 2 PWM opportunities, this results in 62.5 usec or 125 usec PWM reload period @ 16 kHz PWM frequency.

- **Deadtime Correction Method**

- Current Status Bits (ISENS[1:0]) - must be set to 0 for no correction, see [Table 4-2](#)
- Current Polarity Bits (IPOLx) - must be set to 0 or the reset state can be left.

- **PWM Frequency, Modulus (PWMCM)** - setting must reflect the required PWM frequency.

Note the following formula defines the PWM period for center-aligned PWM mode.

$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$

16 kHz PWM frequency (62.5 usec PWM period) can be achieved setting the PWM modulus = 1250 dec @ 25nsec PWM clock period.

- **PWM Pins SW Output Control** - Output Pad Enable (PAD_EN = 1), no output control is needed so the OUTCTLx and OUTx pins in the PMOUT register can be left at their reset state.
- **PWM Channel Control** - no channel control needed (HW acceleration disabled, no masked PWM generator channels, no swapped channel, each Value Register is accessed independently), the PMCCR register can be left in its reset state.
- **Faults** -

- **PWM Disable Mapping (PMDISMAP1-2)** - These write-protected registers determine disabled PWM pins by the fault protection inputs, illustrated in [Table 4-5](#) The settings must reflect the application specific requirements. Reset sets all of the bits used in the PWM disable mapping registers.

There is no one “magic” setting satisfying all needs. Mostly the Fault 2 is connected to the DC-bus over-current detection circuitry, the Fault 3 is then connected to the DC-bus over-voltage detection circuitry. Further selection is that both faults disable all PWMs and the Fault 0,1 are inactive. The PWM disable mapping bits: DISMAP2, 3, 6, 7, 10, 11, 14, and 15 = 1 other DISMAPx bits = 0.

- **Fault Clearing Mode** - automatic or manual clearing mode can be individually set for each fault. Usually the automatic fault clearing is used for automatic current limiting. If an application must perform some more sophisticated checking before enabling PWM again, the manual fault clearing should be selected.
- **FAULT Interrupt Enable (FIE_x)** - setting bits 2 and 3 in PMFCTL enables CPU interrupt requests generated by the FAULT2,3 pins. Usually the fault interrupt is enabled

when in the manual clearing mode. Then the corresponding ISR serves the over-current or over-voltage exception in accordance with the application requirements.

- **Write-protection** - If required the safety critical setting can be protected by the write-protect (WP) bit in PMCFG. Once set it prevents any further writes to write-protected registers or bits. WP can be cleared only by a reset. The following is a list of registers (their bits) and functions which are write-protected:
 - PMCFG - the PWM polarity, the complementary PWM pair operation mode and center aligned PWM channels
 - PMDEADTM - the deadtime
 - PMDISMAP1-2 - the PWM fault disable mapping matrix
 - PMCCR - ENHA = 0 - the disabled HW acceleration features (Value Register Load Mode - VLMODE, PWM generator channels swapping)

The functions which are still available follow: PWM Value setting, PWM Frequency setting, PWM Clock prescaler/prescaler setting, PWM Load Frequency + Half Cycle reload setting, Deadtime Correction Method setting (not suitable), odd/even PWMVALx selection (not suitable), Fault Interrupts Enable/Disable, Fault Clearing Mode selection, Fault acknowledging, PWM Pins SW Output Control (not suitable), PWM generator channels masking and swapping (not suitable).

- **PWM Value Registers (PWMVALx)** - usually all duty cycles are initially set the same - producing no current (zero phase-phase voltage). 50% duty cycle would produce “natural zero” level (refer to [Figure 3-1](#) and (EQ 3-5)), while it helps to keep charging the bootstrap circuitry in the power switch drivers. The PWM Value calculation is as follows:

$$\text{PWM value } x = (\text{duty cycle } x) / 100\% \times (\text{PWM modulus})$$

$$\text{PWMVAL}_x = 50\% / 100\% \times 1250 \text{ dec} = 625 \text{ dec for previously selected PWM modulus.}$$

The PWM value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins.

- **PWM Reload Interrupt Enable (PWMRIE)** - this bit enables the PWMF flag to generate CPU interrupt requests. See also the notes below.
- **PWM Enable (PWMEN)** - this bit enables the PWM generator and the PWM pins. It should be set at the end of initialization. See also the notes below.

Notes: Initialize all registers and set the LDOK bit before setting the PWMEN bit. With LDOK set, setting PWMEN for the first time after reset immediately loads the PWM generator, thereby setting the PWMF flag. PWMF generates a CPU interrupt request if the PWMRIE bit is set. In complementary channel operation with current-status correction selected, PWM value registers one, three, and five control the outputs for the first PWM cycle.

Notes: The Embedded SDK (Software Development Kit) supports initialization and control of the MC PWM module through its drivers and commands. The Embedded SDK default setting might not match the MC PWM reset state, therefore it is highly recommended to perform complete settings of all functions. This is also good programmer practice, because it simplifies porting the code to future platforms.

5.4.1.2 Voltage Control - Cycle-by-cycle Modification of PWM

The MC PWM Module is initialized to generate complementary, center-aligned PWM waveforms with given duty cycle and PWM frequency.

As it is described in previous sections, the PWM duty cycle is periodically updated according to the speed (or torque, or current) controller output. Such a controller calculates the new values of the PWM duty cycles (PWMVALx).

Performing the following steps on regular basis (each reload) leads to modification of the PWM duty cycle and hence modulation of the motor load DC voltage.

1. Upon a reload opportunity (defined by the PWM Load Frequency + Half Cycle reload setting), regardless of whether an actual reload occurs as determined by LDOK bit, the PWMF reload flag is set.
2. If the PWM reload interrupt enable bit, PWMRIE is set, the PWMF flag generates CPU interrupt requests, allowing software to calculate new PWM Values in real time reflecting the application speed (or torque, or current) controller output.

Notes: When PWMRIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

3. Then the following steps must be performed (usually in the PWM reload ISR):
 - the PWMVAL0 and PWMVAL2 registers (buffers) must be updated with new values previously calculated by SW (the PWMVAL0 reflecting the (EQ 5-9) and the PWMVAL2 reflecting the (EQ 5-10))
 - the PWMF reload flag must be acknowledged by reading PWM Control Register with PWMF set and then writing a logic zero to the PWMF bit
 - the load okay bit, LDOK, needs to be set by reading it when it is a logic zero and then writing a logic one to it. LDOK prevents reloading of these PWM parameters before software is finished calculating them.
4. The loaded buffers use the PWM generator at the beginning of the next PWM reload cycle.
5. After loading, LDOK is automatically cleared.

See [Figure 5-17](#) for signal waveforms.

The load okay bit, LDOK, enables loading the PWM generator with more new parameters, such as:

- a prescaler divisor - from the PRSC1 and PRSC0 bits in PWM Control Register,
- a PWM period - from the PWM counter modulus registers (PWMCM),
- a PWM pulse width - from the PWM value registers (PWMVALx).

This feature enables an on-the-fly variation of the PWM frequency - acoustic noise elimination technique already mentioned in [Section 5.1.3](#).

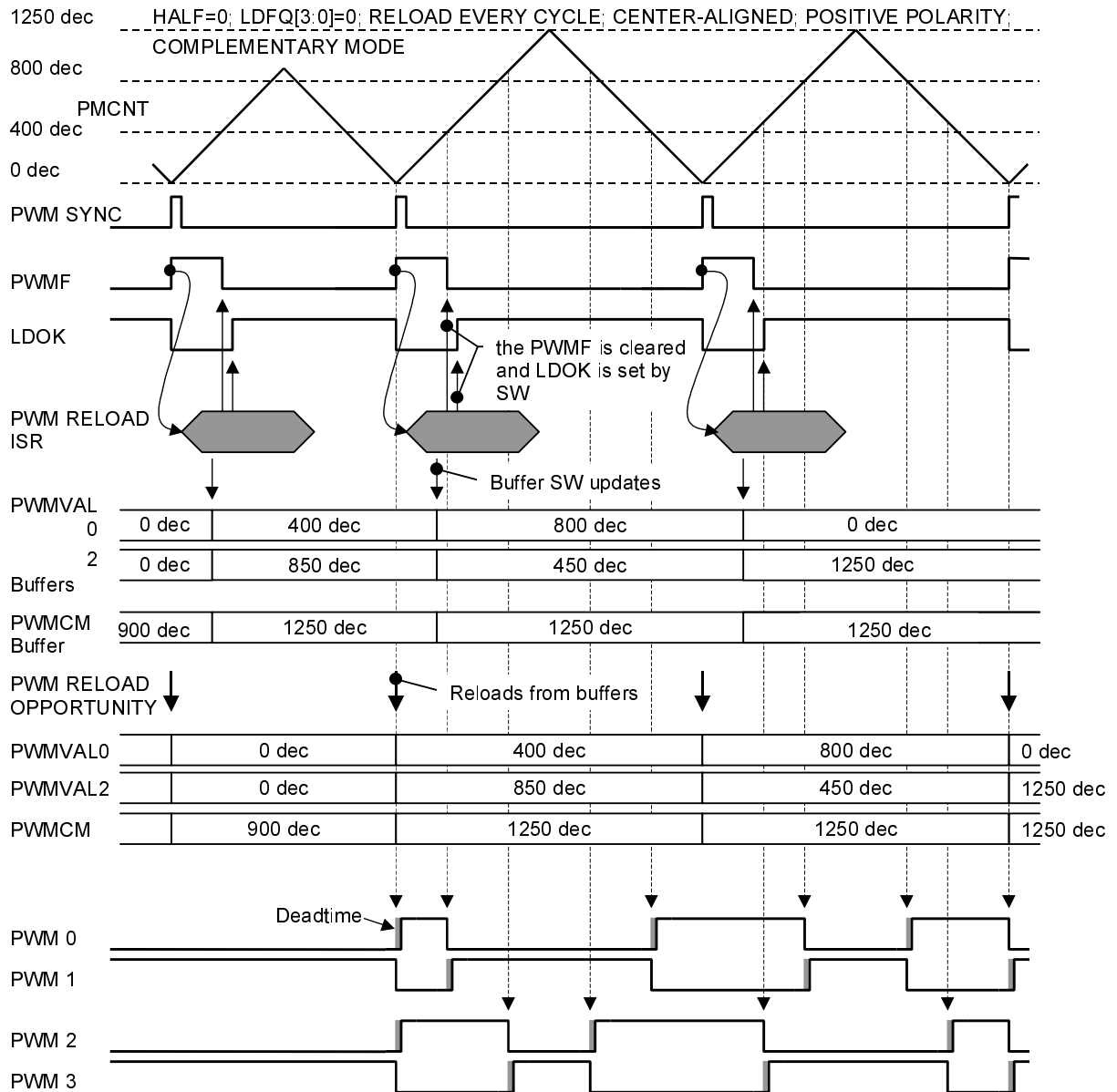


Figure 5-17. DC Brushed Motor - PWM Waveforms

6. Conclusion

New dedicated Motorola advanced motor control chips - DSP56F80x - offer exceptional motor control on-chip Pulse Width Modulator with high degree of flexibility, as a result of implementing what our customers have asked for through their feedback, and by learning from our long history of practical experience.

Preliminary Copy


This motor control PWM module is an effective solution for large number of applications, starting from the servo and motor control of a variety of types, the power inverter and converter applications up to the actuator control or the advanced D class audio amplifier.

This application note describes basics of the motor control PWM techniques, the key functions of the motor control PWM module and its usage. Four classic motor control applications - ACIM, PMSM, BLDC motor, SRM and DC Brushed motor - were used to demonstrate the key features, the step-by-step setting and on-fly modification of the motor control PWM module parameters. All this should serve not only as a “template” for user application but also to shorten the learning time.

7. References

- [1.] DSP56F800 16-bit Digital Signal Processor, Family Manual, DSP56F800FM/D, Motorola
- [2.] DSP56F80x 16-bit Digital Signal Processor, User’s Manual, DSP56F801-7UM/D, Motorola
- [3.] Making Low-Distortion Motor Waveforms with the MC68HC708MP16 - AN 1728, AN1728/D
- [4.] Embedded SDK (Software Development Kit), Motor Control Library
- [5.] Embedded SDK (Software Development Kit), Targeting Motorola DSP5680x Platforms
- [6.] 3-Phase AC Motor Control with V/Hz Speed Open Loop using DSP56F80X, AN1911/D
- [7.] 3-Phase BLDC Motor Control with Hall Sensors using DSP56F80x, AN1916/D
- [8.] 3-Phase BLDC Motor Control with Sensorless BEMF Zero Crossing Detection using DSP56F80x, AN1914/D
- [9.] 3-phase BLDC Motor Control with Sensorless Back-EMF ADC Zero Crossing Detection using DSP 56F80x, AN1913/D
- [10.] 3-Phase SR Motor Control with Hall Sensors using DSP56F80x, AN1912/D
- [11.] Motorola motor control web page: *e-www.motorola.com/motor/*
- [12.] Motorola Embedded Software Development Kit web page:
e-www.motorola.com/motor/devtools/sdk.html

Preliminary Copy

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 which may be provided in Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2001.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu. Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

Technical Information Center: 1-800-521-6274

HOME PAGE: <http://www.motorola.com/semiconductors/>

