ADSP-21990: Generation of Three-phase Sine-wave PWM patterns

AN21990-03

Table of Contents

Summary

This application note demonstrates the use of the Pulse Width Modulation (PWM) block of the ADSP-21990 to generate three-phase sine-wave pattern. It gives an example of configuring the PWM registers and setting the PWM interrupts, i.e. the PWMSYNC and PWMTRIP interrupts.

1 Implementation of the PWM library routine

1.1 Usage of PWM routine

In this library, one routine is defined for the PWM initialisation. It is developed as an easy-to-use library, which has to be linked to the user's application. The library consists of two files. The file "pwm.dsp" contains the assembly code of the subroutines. The block has to be compiled and then linked to an application. The user has to include the header file "pwm.h", which provides the function-like call to the subroutine. The following table shows the macro that is defined in this library.

Operation	*Usage*	*Explanation*
Initialisation	**PWM_Init(PWMTRIP_int_label, PWMSYNC_int_label)**	**Initialise the PWM block, including configuring the interrupts**

Table 1 Implementation Routine

This routine requires some configuration constants from the main include-file "main.h" that comes with every application note. Section 2 shows an example of usage of this library. For more information regarding this routine, see the comments in the "pwm.h" and the "pwm.dsp" files.

1.2 Usage of DSP registers

The macro listed in the Table 1 is based on the subroutine listed in Table 2. This subroutine will be discussed in the following section. The following table gives an overview of what DSP registers are used in this macro:

Subroutine	*Inputs*	*Outputs*	*Modified registers*
PWM_Init_	Configuration constants: i.e. PWM period, dead time, PWMSYNC pulse width	None	AR, AY0, I4, MR0, SR, AX0, AF, PX

Table 2 Input and output format, modified registers in the PWM initialization routine

1.3 Configuring the PWM block

1.3.1 pwm.h

The library may be accessed by including the header file "pwm.h" into the application code. A macro, PWM_Init, is defined in this file. It sets the interrupt vector for the PWMTRIP and the PWMSYNC interrupts. (Another macro called the Set_InterruptVector is used; which is defined in the general-purpose file "macro.h"). In the ADSP-21990, there are 12 user-defined interrupt lines, from USR0 (highest priority) to USR11 (lowest priority). In this macro, the PWMTRIP interrupt is assigned to USR0 and PWMSYNC interrupt is assigned to USR1. It then provides a call to the PWM_Init_ routine, which is defined in the pwm.dsp file. For more information on the "pwm.h" file, please refer to the comments in the file.

1.3.2 pwm.dsp

In this file, the PWM_Init_ routine is defined, which initialises the PWM block. It obtains the system parameters and PWM configuration values defined in "main.h"(refer to Section 2.2), then calculates and loads the correct values into the PWM period, dead time and PWMSYNC pulse width registers. The calculations are as shown in equations (1), (2) and (3). It also initialises the duty cycles of all channels to 50% and then enables the PWM outputs. For more information on the "pwm.dsp" file, please refer to the comments in the file.

$$\text{PWM period register (PWM0\_TM)} = \frac{H_{clk}}{2 \text{ x } f_{pwm}} \quad (1)$$

$$\text{PWM dead time register (PWM0\_DT)} = \frac{\text{Dead time}[s]}{2} \text{ x } H_{clk} \quad (2)$$

$$\text{PWM sync pulse register (PWM0\_SYNCWT)} = \text{Sync pulse width}[s] \text{ x } H_{clk} - 1 \quad (3)$$

Where H_{clk} is the peripheral clock [Hz], f_{pwm} is the PWM switching frequency [Hz]

2 Software Example: Generation of Three-Phase Sine-Wave PWM

In this example, three-phase sine-wave PWM patterns are generated. The user can specify the fundamental frequency and the amplitude of the waveform in the "main.dsp" program.

2.1 Main.dsp

It contains the initialisation and PWMSYNC and PWMTRIP interrupt service routines. Besides the "pwm.h" and "main.h", since sinusoidal calculation is also required, the "trigono.dsp" file is also included. Some constants and variables are defined. In this example, the fundamental frequency is 60Hz.

At the start of the program, initialisations such as setting the data memory page registers to accessing the internal memory and PWM initialisation are accomplished (see the "pwm.h' and "pwm.dsp" files for details). Note that the duty cycle register range is from: (-PWM0_TM/2 – PWM0_DT) to (PWM0_TM/2 + PWM0_DT). Hence, the maximum duty cycle value is |PWM0_TM/2 + PWM0_DT| and a value of 0 represents a 50% PWM duty cycle. The amplitude of the waveform is then scaled by the amplitude factor specified by the user. Taking the dead time into account, to avoid over-modulation[1], the maximum duty cycle should be in the range of (-PWM0_TM/2 + PWM0_DT) to (PWM0_TM/2 – PWM0_DT). Hence, the maximum amplitude factor without over modulation is:

$$\text{Amplitude factor (maximum without over-modulation)} = \frac{(\text{PWM0\_TM}/2 - \text{PWM0\_DT})}{(\text{PWM0\_TM}/2 + \text{PWM0\_DT})} \quad (4)$$

The PWMSYNC interrupt service routine is executed at the PWM switching rate (20kHz here). First, the PWMSYNC interrupt status in the PWM status register is cleared. The phase angle Theta is incremented and stored for the next interrupt. Sin(Theta) is computed for each phase such that phase A, B and C are 120^o apart. The duty cycle for each phase is computed, and then written to the duty cycle registers.

[1] . In the case of over-modulation, a smooth transition of PWM patterns into both full ON and full OFF conditions is ensured. For instance, if the commanded AH is increased until the corresponding signal transitions into either the fully ON state due to the particular values of duty cycle and dead time that are programmed, the corresponding low side signal may not have reached the full OFF state. When further increasing the duty cycle, AH will be held at 100%, while the AL signal smoothly decreases to 0%. Similar behaviour is valid for decreasing the duty cycle to 0% (transition into FULL OFF).

In the PWMTRIP interrupt service routine, the users can define what to implement in case of a PWM fault. In this example, the status bit of the PWMTRIP interrupt in the PWM status register is cleared.

For more details on the "main.dsp", please refer to the comments in the file.

2.2 Main.h

As discussed before, the configuration of the PWM block requires some constants, which are defined in the "main.h" file. The use of this file is to define general system parameters and constants, such as the crystal clock [kHz], core clock [kHz] and the peripheral clock [kHz] (also denoted as H_{clk}). Besides, any constants required by the library files will be defined here. In this case, the constants needed by the PWM library, such as the PWM switching frequency [Hz], the dead time [nsec] and the PWM sync pulse time [nsec] will be included in this file. Two files, the "ADSP-21990.h" and the "macro.h" are also included in the "main.h" file. The "ADSP-21990.h" defines the peripheral registers of the ADSP-21990 while the "macro.h" defines the most commonly used macro.

3 Experimental Results

The results below show the waveforms on the PWM channels while running this example. Here, the crystal clock is 25MHz, the peripheral clock (H_{clk}) is 12.5kHz. PWM switching frequency = 20kHz, PWM dead time = 2μs and PWM sync pulse width = 440ns. The fundamental frequency of the sine wave is 60Hz. Figure 1 to Figure 4 show the PWM outputs without over-modulation. In Figure 1, one can see that the fundamental frequency of 60Hz is achieved. Figure 2 shows the sine waves of the three phases, being 120^o apart. From Figure 4, it shows a dead time of 2μs is asserted. Figure 5 shows the outputs when over-modulation occurs.

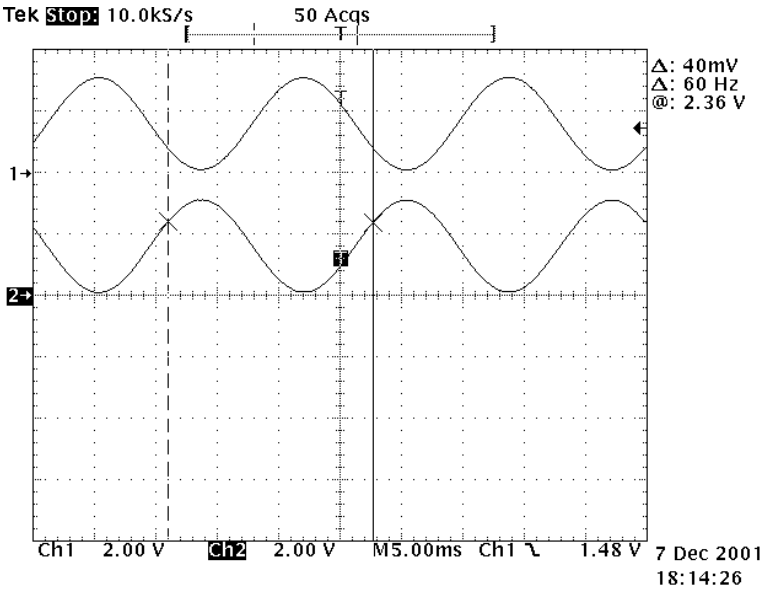

Figure 1 Filtered sine PWM outputs Ch1: AH, Ch2:AL

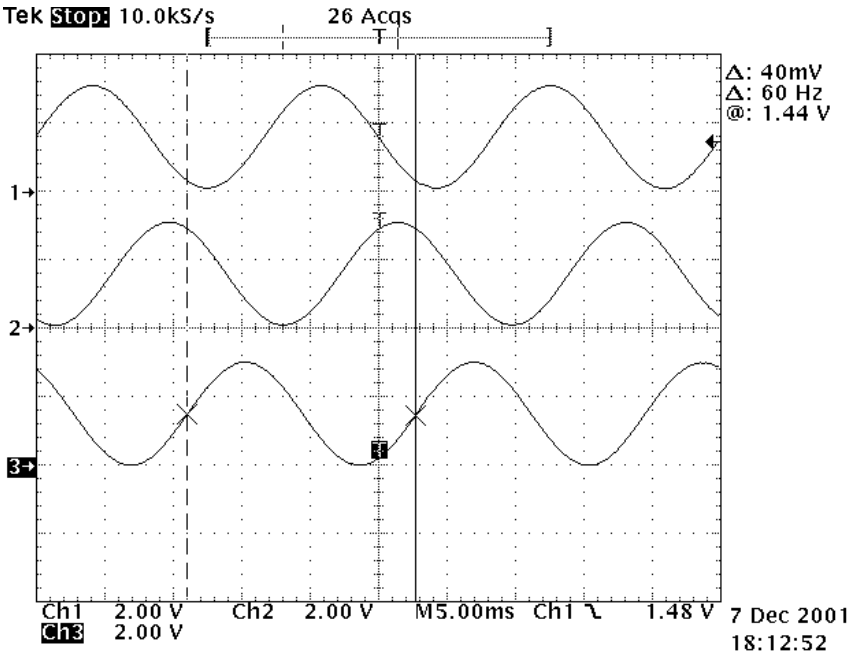

Figure 2 Filtered PWM outputs, Ch1: AH, Ch2: BH, Ch3: CH

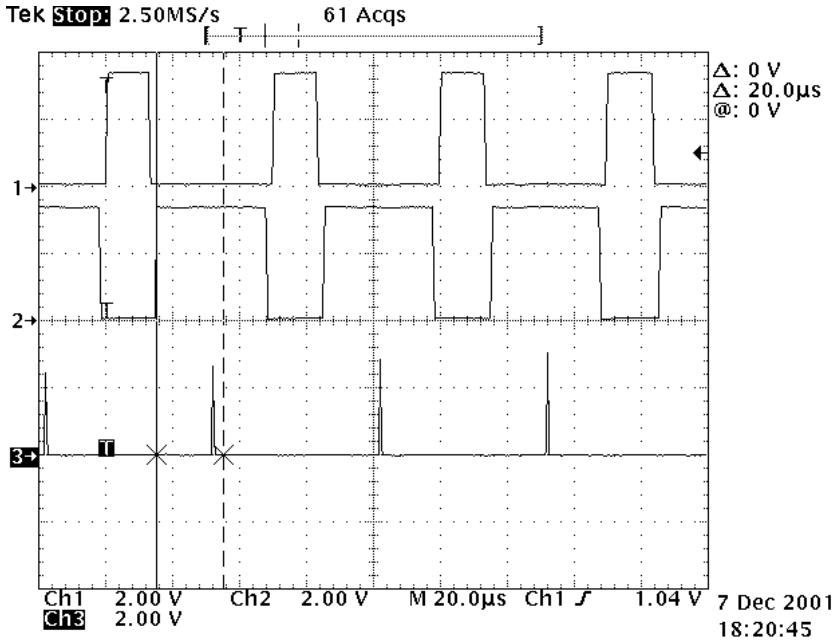

Figure 3 PWM outputs, Ch1: AH, Ch2: AL, Ch3: PWMSYNC

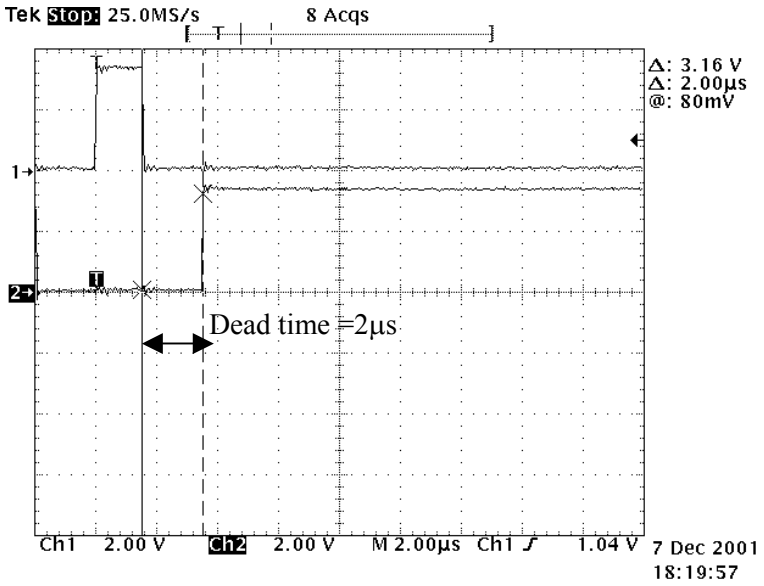

Figure 4 PWM outputs, Ch1: AH, Ch2:AL

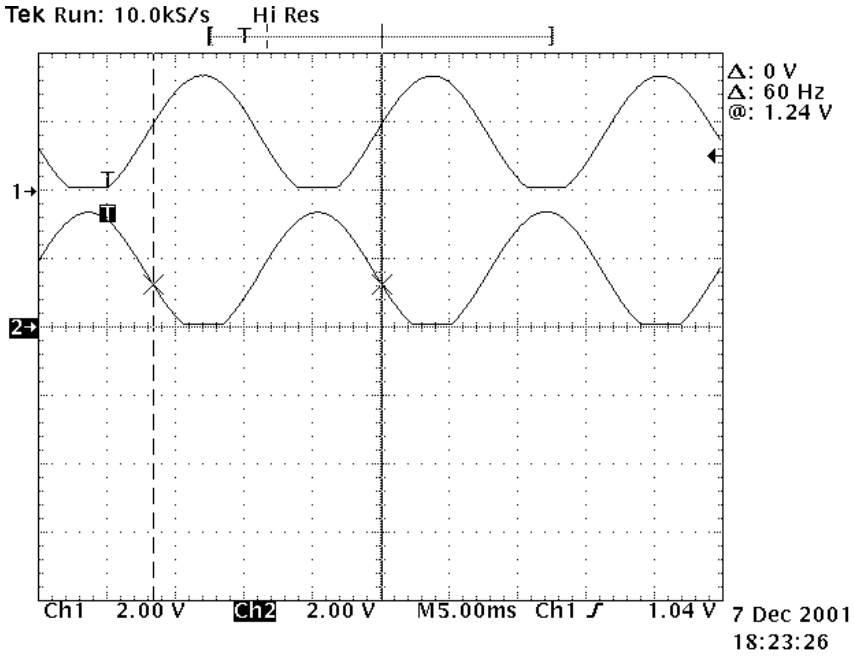

Figure 5 Filtered PWM outputs (over modulation) Ch1: AH, Ch2: AL

