

Space-Vector PWM With TMS320C24x/F24x Using Hardware and Software Determined Switching Patterns

Zhenyu Yu

Digital Signal Processing Solutions

Abstract

Space-vector (SV) pulse width modulation (PWM) technique has become a popular PWM technique for three-phase voltage-source inverters (VSI) in applications such as control of AC induction and permanent-magnet synchronous motors. This document gives an in-depth discussion of the theory and implementation of the SV PWM technique.

Two different SV PWM waveform patterns, one using the regular compare function on the Texas Instruments (TI™) TMS320C24x/F24x digital signal processors (DSPs) and another implemented with the SV PWM hardware module on the TI TMS320C24x/F24x DSPs are presented, with complete code examples for the TMS320F243/1. At the end, a complete AC induction motor control application is discussed to show the effectiveness of both approaches.

PWM waveforms of the presented implementations and experimental data in the form of motor currents are shown and discussed. A full TMS320F243/1 program example is attached. The observation of dead band imbalance for the hardware-implemented SVPWM pattern in this report has not been seen in other publications.

Contents

Introduction	2
Background	3
Theory of SV PWM Technique.....	3
SV PWM Waveform Patterns.....	9
Application in Three-Phase AC Induction Motor Control	20
Experimental Results	22
Conclusions	22
References.....	24
Appendix A. Program for Open-Loop Three-Phase AC Induction Motor Control With SV PWM Technique and Constant V/Hz Principle	25

Figures

Figure 1. Symmetric and Asymmetric PWM Signals	2
Figure 2. Three-Phase VSI Diagram	3
Figure 3. The Basic Space Vectors (Normalized w.r.t. V_{dc}) and Switching States	5
Figure 4. Software Determined SV PWM Waveform Pattern	10
Figure 5. Switching Direction for Software Determined SV PWM Pattern	11
Figure 6. SV PWM Outputs With Carrier Filtered Out	13
Figure 7. SV PWM Outputs With Carrier Filtered Out and Dead Band Enabled	14
Figure 8. Hardware-Implemented SV PWM Waveform Pattern	15
Figure 9. SV PWM Outputs With Carrier Filtered Out	19
Figure 10. SV PWM Outputs With Carrier Filtered Out and Dead Band Enabled	19
Figure 11. Program Flow Chart	20
Figure 12. Block Diagram of an Open-Loop AC Induction Motor Control System	22
Figure 13. Motor Current and Spectrum Obtained With the Software Approach	23
Figure 14. Motor Current and Spectrum Obtained With the Hardware Approach	23

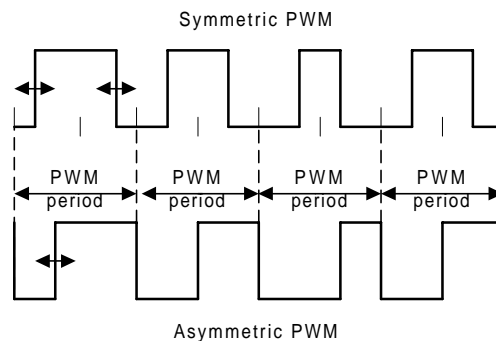
Tables

Table 1. Device On/Off States and Corresponding Outputs of a Three-Phase VSI	4
Table 2. Determination of the Sector of U_{out} Based on N	8
Table 3. Hardware and Software Determined SV PWM Switching Pattern Comparison	9

Introduction

Because of advances in solid state power devices and microprocessors, PWM inverters are becoming more and more popular in today's motor drives. PWM inverters make it possible to control both the frequency and magnitude of the voltage and current applied to a motor. As a result, PWM inverter-powered motor drives offer better efficiency and higher performance compared to fixed frequency motor drives. The energy that a PWM inverter delivers to a motor is controlled by PWM signals applied to the gates of the power transistors, as shown in Figure 1.

Figure 1. Symmetric and Asymmetric PWM Signals



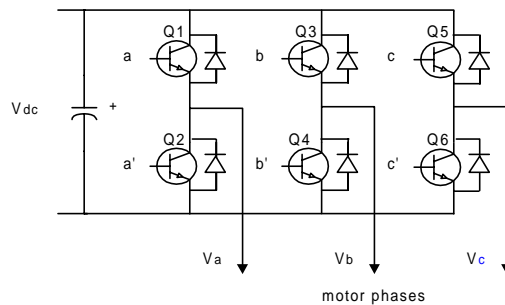
Different PWM techniques (ways of determining the modulating signal and the switch-on/switch-off instants from the modulating signal) exist. Popular examples are sinusoidal PWM, hysteric PWM and the relatively new space-vector (SV) PWM. These techniques are commonly used for the control of AC induction, BLDC and Switched Reluctance (SR) motors. The SV PWM technique for three-phase voltage-source inverter (VSI) is addressed in this application.

Background

Theory of SV PWM Technique

The structure of a typical three-phase VSI is shown in Figure 2. As shown below, V_a , V_b and V_c are the output voltages of the inverter. Q_1 through Q_6 are the six power transistors that shape the output, which are controlled by a , a' , b , b' , c and c' . When an upper transistor is switched on (i.e., when a , b or c is 1), the corresponding lower transistor is switched off (i.e., the corresponding a' , b' or c' is 0). The on and off states of the upper transistors, Q_1 , Q_3 and Q_5 , or equivalently, the state of a , b and c , are sufficient to evaluate the output voltage for the purpose of this discussion.

Figure 2. Three-Phase VSI Diagram



The relationship between the switching variable vector $[a, b, c]^t$ and the line-to-line output voltage vector $[V_{ab} \ V_{bc} \ V_{ca}]^t$ and the phase (line-to-neutral) output voltage vector $[V_a \ V_b \ V_c]^t$ is given by equation 1 and equation 2 below.

$$\begin{bmatrix} V_{ab} \\ V_{bc} \\ V_{ca} \end{bmatrix} = V_{dc} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (\text{equation 1})$$

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \frac{1}{3} V_{dc} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (\text{equation 2})$$

where V_{dc} is the DC supply voltage, or bus voltage.

As shown in Figure 2, there are eight possible combinations of on and off states for the three upper power transistors. The eight combinations and the derived output line-to-line and phase voltages in terms of DC supply voltage V_{dc} , according to equations 1 and 2, are shown in Table 1.

SV PWM refers to a special way of determining the switching sequence of the upper three power transistors of a three-phase VSI. It has been shown to generate less harmonic distortion in the output voltages and or currents in the windings of the motor load and provides more efficient use of DC supply voltage, in comparison to direct sinusoidal modulation technique.



Table 1. Device On/Off States and Corresponding Outputs of a Three-Phase VSI

a	b	c	v _a	v _b	v _c	v _{ab}	v _{bc}	v _{ca}
0	0	0	0	0	0	0	0	0
1	0	0	2/3	-1/3	-1/3	1	0	-1
1	1	0	1/3	1/3	-2/3	0	1	-1
0	1	0	-1/3	2/3	-1/3	-1	1	0
0	1	1	-2/3	1/3	1/3	-1	0	1
0	0	1	-1/3	-1/3	2/3	0	-1	1
1	0	1	1/3	-2/3	1/3	1	-1	0
1	1	1	0	0	0	0	0	0

Assume d and q are the fixed horizontal and vertical axes in the plane of the three motor phases. The vector representations of the phase voltages corresponding to the eight combinations can be obtained by applying the following so-called d - q transformation to the phase voltages:

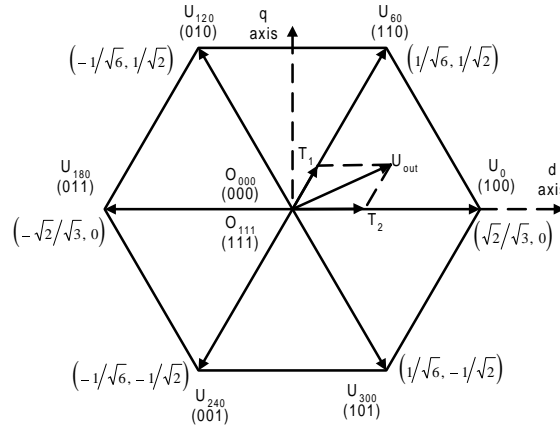
$$T_{abc-dq} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \quad (\text{equation 3})$$

This transformation is equivalent to an orthogonal projection of $[a, b, c]^T$ onto the two dimensional plane perpendicular to the vector $[1, 1, 1]^T$ in a three-dimensional coordinate system, the results of which are six non-zero vectors and two zero vectors as shown in Figure 3. The nonzero vectors form the axes of a hexagonal. The angle between any adjacent two non-zero vectors is 60 degrees. The zero vectors are at the origin and apply zero voltage to a three-phase load. The eight vectors are called the Basic Space Vectors and are denoted here by $U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000}$ and O_{111} .

The same d - q transformation can be applied to a desired three-phase voltage output to obtain a desired reference voltage vector U_{out} in the d - q plane as shown in Figure 3. Note that the magnitude of U_{out} is the rms value of the corresponding line-to-line voltage with the defined d-q transform.

The objective of SV PWM technique is to approximate the reference voltage U_{out} instantaneously by combination of the switching states corresponding the basic space vectors. One way to achieve this is to require, for any small period of time T , the average inverter output be the same as the average reference voltage U_{out} as shown in equation 4. Note, T_1 and T_2 in equation 4 are the respective durations for which switching states corresponding to U_x and U_{x+60} (or U_{x-60}) are applied. U_x and U_{x+60} (or U_{x-60}) are the basic space vectors that form the sector containing U_{out} . However, if we assume that the change in reference voltage U_{out} is tiny within T , then equation 4 becomes equation 5, where $T_1 + T_2 \leq T$. Therefore, it is critical that T be small with respect to the speed of change of U_{out} . In practice the approximation is done for every PWM period, T_{pwm} . Therefore it is critical that the PWM period be small with respect to the speed of change of U_{out} .

Figure 3. The Basic Space Vectors (Normalized w.r.t. V_{dc}) and Switching States



$$\frac{1}{T} \int_{nT}^{(n+1)T} U_{out}(t) dt = \frac{1}{T} (T_1 U_x + T_2 U_{x\pm 60}) \quad (\text{equation 4})$$

$$U_{out}(nT) = \frac{1}{T} (T_1 U_x + T_2 U_{x\pm 60}) \quad (\text{equation 5})$$

Equation 5 means that for every PWM period, U_{out} can be approximated by having the inverter in switching states U_x and U_{x+60} (or U_{x-60}) for T_1 and T_2 duration of time respectively. Since the sum of T_1 and T_2 should be less than or equal to T_{pwm} , the inverter needs to be in O_{000} or O_{111} state for the rest of the period. Therefore, equation 5 becomes equation 6 in the following, where $T_1 + T_2 + T_o = T_{pwm} = T$.

$$T_{pwm} U_{out} = T_1 U_x + T_2 U_{x\pm 60} + T_o (O_{000} \text{ or } O_{111}) \quad (\text{equation 6})$$

From equation 6, we get equation 7 for T_1 and T_2 .

$$\begin{bmatrix} T_1 & T_2 \end{bmatrix}^T = T_{pwm} \begin{bmatrix} U_x & U_{x\pm 60} \end{bmatrix}^{-1} U_{out} \quad (\text{equation 7})$$

where $\begin{bmatrix} U_x & U_{x\pm 60} \end{bmatrix}^{-1}$ is the normalized decomposition matrix for the sector.

Assume the angle between U_{out} and U_x is α . From Figure 3, we can also obtain equation 8 in the following for T_1 and T_2 .

$$\begin{aligned} T_1 &= \sqrt{2} T_{pwm} \|U_{out}\| \cos(\alpha + 30^\circ) \\ T_2 &= \sqrt{2} T_{pwm} \|U_{out}\| \sin(\alpha) \end{aligned} \quad (\text{equation 8})$$

Depending on specific application, calculation of T_1 and T_2 can be done either with equation 7 or equation 8. Equation 7 is sector dependent. However, the matrix inverse can be calculated off-line for each sector and obtained via a look-up table during on-line calculation. This approach is useful when U_{out} is given in the form of vector $[U_d, U_q]^T$. Equation 8 is independent of sector and is useful when U_{out} is given in the form of magnitude and phase angle.



U_x can be the closest basic space vector on either side of U_{out} . U_{x+60} (or U_{x-60}) is then the basic space vector on the opposite side. In either case, T_1 represents the component on U_x , T_2 represents the component on the other basic space vector.

The following is a code example to calculate T_1 and T_2 (as compare values) using equation 7.

Example 1. Code Example for Calculation of T_1 and T_2 Using Equation 7

```

                .data
*****
** Decomposition matrices indexed by the sector, s, Uout is in      **
*****
decomp_        .WORD 20066      ; D1-scaled by 2 to the 14th power
                .WORD -11585
                .WORD 0
                .WORD 23170

                .WORD -20066
                .WORD 11585
                .WORD 20066
                .WORD 11585

                .WORD 0
                .WORD 23170
                .WORD -20066
                .WORD -11585

                .WORD 0
                .WORD -23170
                .WORD -20066
                .WORD 11585

                .WORD -20066
                .WORD -11585
                .WORD 20066
                .WORD -11585

                .WORD 20066
                .WORD 11585
                .WORD 0
                .WORD -23170
;
;
                .bss decomp,24    ; decomposition matrices
                .bss temp,1      ; temporary storage

                .txt
*****
** Initialize the decomposition matrices                          **
*****
                LAR  AR0,#decomp    ; Point to 1st destination
                LAR  AR1,#(24-1)    ; 24 entries
                LACC #decomp_      ; Point to 1st data item
                MAR  *,AR0          ; Point to AR0

init_table

```



```

        TBLR  *,1          ; Move data&pnt to nextdesti.
        ADD   #1          ; Point to next data item
        BANZ  init_table,0 ; Continue if there is more
;
;
;-----
; Calculate  $T_1$ & $T_2$  as compare values based on:  $T_{pwm} U_{out}=V_1*T_1+V_2*T_2$ 
;
; i.e.  $[T_1 T_2]=T_{pwm}*inverse[V_1 V_2]*U_{out}$ 
; i.e.  $[0.5*T_1 0.5*T_2]=T_p*inverse[V_1 V_2]*U_{out}$ 
; i.e.  $[0.5*C_1 0.5*C_2]=inverse[V_1 V_2]*U_{out}=M(sector)*U_{out}$ 
; where  $C_1=T_1/T_p$ ,  $C_2=T_2/T_p$ , are normalized  $T_1$ & $T_2$  wrt  $T_p$ 
;  $M(sector)=inverse$  of  $[V_1 V_2]$  = decomposition matrix
; obtained through table lookup
;  $U_{out}=\text{Transpose of } [U_d U_q]$ 
;  $T_p=\text{Timer 1 period} = 0.5*T_{pwm}$ 
;  $T_{pwm}=\text{PWM period}$ 
; Input S: sector of  $U_{out}$  (0-5)
;  $U_d$ : d compo. of  $U_{out}(0-1/\sqrt{2})$ , D2(Scaled by  $2^{**13}$ )
;  $U_q$ : q compo. of  $U_{out}(0-1/\sqrt{2})$ , D2(Scaled by  $2^{**13}$ )
; t1_period_: Timer period (for PWM freq)
; t1_periods: Timer period in D10 (Scaled by  $2^{**5}$ )
; Output cmp_0:  $0.5(1-0.5C_1-0.5C_2)T_p$  cmp value for 1st-to-tog ch
; cmp_1:  $cmp_0+0.5C_1T_p$  cmp value for 2nd-to-tog ch
; cmp_2:  $cmp_1+0.5C_2T_p$  cmp value for 3rd-to-tog ch
;-----
        LACC  #decomp ;
        ADD   S,2     ;
        SACL  temp    ; get the pointer
        LAR   AR0,temp ; point to parameter table

; Calculate  $0.5C_1$  based on  $0.5C_1=U_d*M(1,1)+U_q*M(1,2)$ 
        LT    Ud      ; D2
        MPY  **      ; M(1,1) Ud:  $D2*D1=D(3+1)$ 
        PAC          ; D4
        LT    Uq      ; D4
        MPY  **      ; M(1,2) Uq:  $D2*D1=D(3+1)$ 
        APAC          ;  $0.5*C_1: D4+D4=D4$ 
        BGEZ  cmp1_big0 ; continue if bigger than zero
        ZAC          ; set to 0 if less than zero
cmp1_big0  SACH  temp    ; D4
        LT    temp    ; D4
        MPY  t1_periods ; *Tp:  $D4*D10 = D(14+1)$ 
        PAC          ;
        SACH  cmp_1    ;  $0.5C_1T_p: D15$  (integer)

; Calculate  $0.5C_2$  based on  $0.5C_2=U_d*M(2,1)+U_q*M(2,2)$ 
        LT    Ud      ; D2
        MPY  **      ; M(2,1) Ud:  $D2*D1=D(3+1)$ 
        PAC          ; D4
        LT    Uq      ; D4
        MPY  **      ; M(2,2) Uq:  $D2*D1=D(3+1)$ 
        APAC          ;  $0.5*C_2: D4+D4=D4$ 
        BGEZ  cmp2_big0 ; continue if bigger than zero
        ZAC          ; zero it if less than zero
cmp2_big0  SACH  temp    ; D4
        LT    temp    ; D4
        MPY  t1_periods ; *Tp:  $D4*D10 = D(14+1)$ 

```



```

PAC          ;
Sach  cmp_2  ; 0.5C2Tp: D15 (integer)

; Calculate compare value 3 based on 0.5C0Tp=(1-0.5C1-0.5C2)Tp
LACC  #t1_period_ ; Calculate 0.5*C0
SUB   cmp_1      ;
SUB   cmp_2      ; 0.5*C0Tp = (1-0.5*C1 -0.5*C2)Tp: D15
BGEZ  cmp0_big0  ; continue if bigger than zero
ZAC   ; zero it if less than zero
cmp0_big0  sfr   ; divide by 2
SACH  cmp_0     ; 0.25*C0Tp: D15 (integer)

```

Note that the D scaling notation is equivalent to the more popular Q notation. Their relationship is $Q_x=D(15-x)$. Therefore, the notation Dx means that the decimal point is at bit[15- x]. Whenever possible, the code examples in this report use maximum scaling to increase resolution and accuracy. For example, since the range of phase angle, θ , is 0 to 2π (or 0 to 6.283), it is designated as a $D3$ (or $Q12$) number for maximum resolution. Therefore the digital representation, θ_d , for θ is related to θ by $\theta_d=\theta*2^{12}$, i.e., scaled up by 2 to the 12th power.

It is necessary to know which sector the reference output voltage is in to determine the switching time instants and sequence. For applications where the reference output voltage vector is given in the form of magnitude and phase angle, such as the program example attached, sector determination is obvious. For applications where the reference output voltage is in terms of vector $[U_d, U_q]^t$, such as where the output voltage vector is derived from an inner current control loop in the d - q frame, the following algorithm can be used to determine the sector of the reference voltage vector. First calculate v_{ref1} , v_{ref2} and v_{ref3} based on equation 9, below.

$$\begin{aligned}
 v_{ref1} &= U_q \\
 v_{ref2} &= \sin 60^\circ U_d - \sin 30^\circ U_q \\
 v_{ref3} &= -\sin 60^\circ U_d - \sin 30^\circ U_q
 \end{aligned}
 \tag{equation 9}$$

Secondly, calculate $N=\text{sign}(v_{ref1})+2*\text{sign}(v_{ref2})+4*\text{sign}(v_{ref3})$. Thirdly, refer to Table 2 below to map N to the sector of U_{out} .

Table 2. Determination of the Sector of U_{out} Based on N

N	1	2	3	4	5	6
Sector	1	5	0	3	2	4

The code examples in this document are based on knowing the phase angle of the reference voltage U_{out} . Therefore, the look-up tables are all in term of sector number of U_{out} . The same look-up tables can easily be rearranged in terms of N instead when the reference voltage is given in terms of vector $[U_d, U_q]^t$.



As mentioned above, the reference voltage vector U_{out} is actually the desired three-phase output voltages mapped to the $d-q$ frame through $d-q$ transformation. When the desired output voltages are three-phase sinusoidal voltages with 120-degree phase shift, U_{out} becomes a revolving vector with the same frequency and a magnitude equal to the corresponding **line-to-line rms voltages**. The envelope of the hexagonal formed by the basic space vectors, as shown in Figure 3, is the locus of maximum U_{out} . Therefore, the magnitude of U_{out} must be limited to the shortest radius of this envelope when U_{out} is a revolving vector. This gives a maximum magnitude of $V_{dc}/\sqrt{2}$ for U_{out} .

Correspondingly, the maximum rms values of the line-to-line and phase output voltages are $V_{dc}/\sqrt{2}$ and $V_{dc}/\sqrt{6}$, which is $2/\sqrt{3}$ times higher than that which an original sinusoidal PWM technique can generate. For the same reason, the bus voltage (V_{dc}) needed for a motor rated at V_{rate} is determined by $V_{dc} = \sqrt{2} V_{rate}$ for SV PWM technique.

SV PWM Waveform Patterns

The arrangement of the order of U_x , $U_{x\pm60}$, O_{000} and or O_{111} in each PWM period is another problem that must be resolved. Different switching orders result in different waveform patterns. Two symmetric switching orders, one that can be easily implemented with TMS320C24x/F24x by software-determined toggling sequences and another implemented by the SV PWM hardware module on the TMS320C24x/F24x, are discussed in this section. Table 3 is a brief comparison between the two switching patterns.

Table 3. Hardware and Software Determined SV PWM Switching Pattern Comparison

Switching Pattern	CPU Overhead (Instruction Cycle)	Memory Usage (Word)	# Switching	Dead Band Imbalance
H/W determined	27	31	4	Yes
S/W determined	33	41	6	No

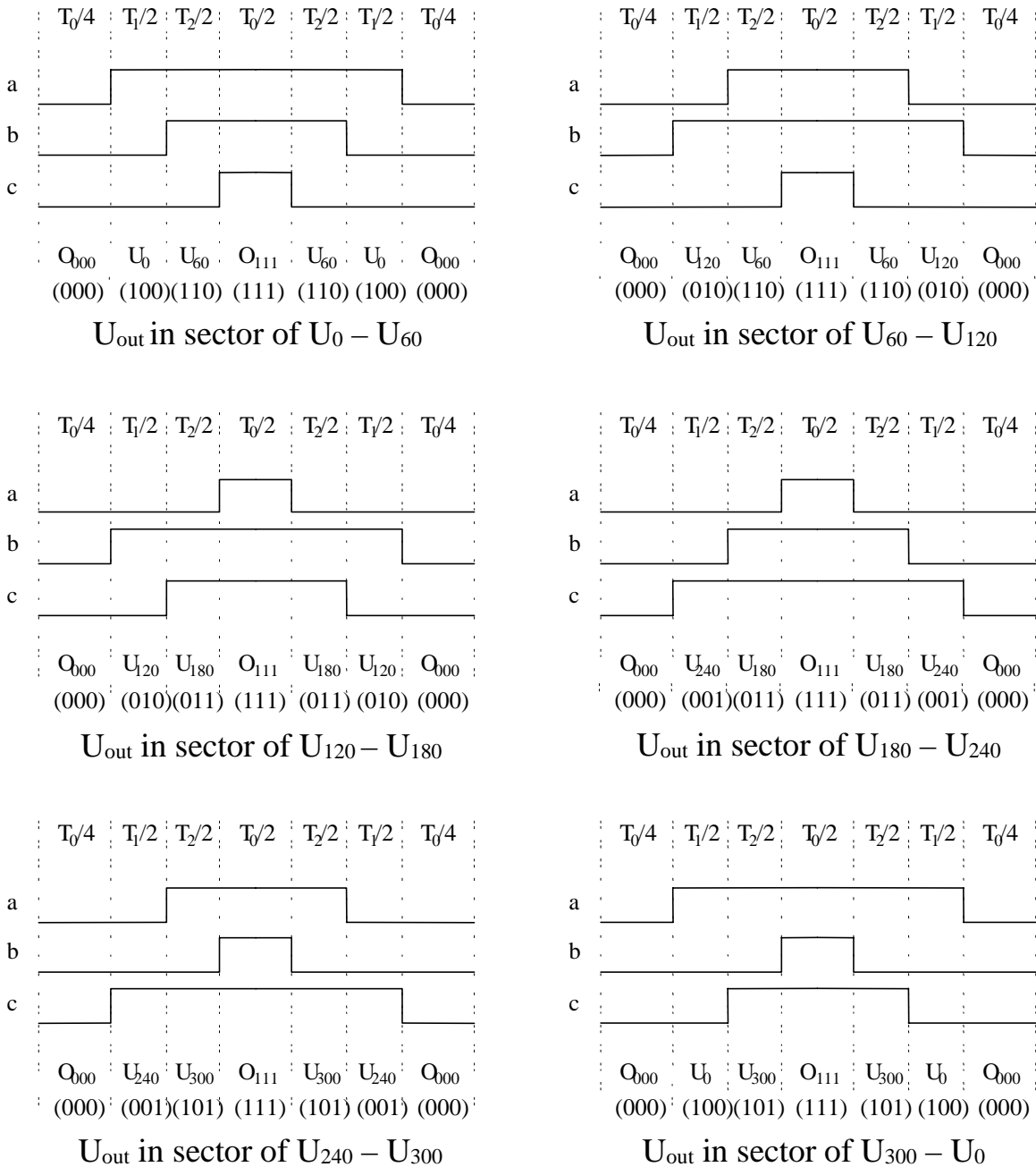
Software-Determined Switching Pattern

Figure 4 below shows the waveform for each sector of a symmetric switching scheme. This scheme can easily be implemented with the TMS320C24x/F24x using software determined switching order for the three PWM channels. Figure 5 is another illustration of the switching scheme, where the arrows indicate for each sector the order of the first and second basic space vectors. This switching scheme can be represented by $(O_{000}, U_x, U_{x\pm60}, O_{111}, O_{111}, U_{x\pm60}, U_x, O_{000})$, where x can be 0, 120 and 240. It has the following properties:

- Each PWM channel switches twice per every PWM period except when the duty cycle is 0% or 100%.
- There is a fixed switching order among the three PWM channels for each sector.
- Every PWM period starts and ends with O_{000} .
- The amount of O_{000} inserted is the same as that of O_{111} in each PWM period.



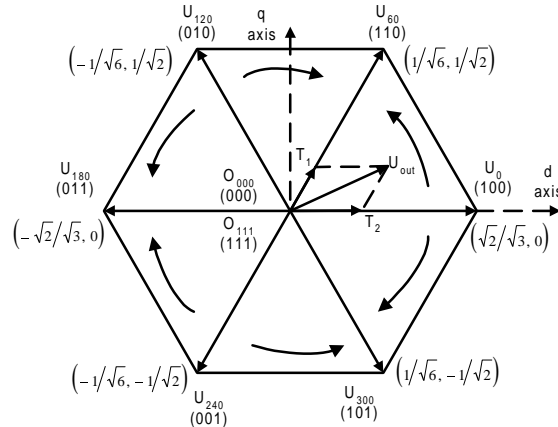
Figure 4. Software-Determined SV PWM Waveform Pattern



Implementation of this switching scheme with TMS320C24x/F24x involves two steps:

- 1) Initialization of the compare units and selected GP Timer for symmetric PWM
- 2) Determination of the channel-toggling sequence based on the look-up table and the load of compare registers based on which sector (s) U_{out} is in.

Figure 5. Switching Direction for Software Determined SV PWM Pattern



Example 2 shows a TMS320C24x/F24x code example that implements this SV PWM scheme.

Example 2. TMS320F243/1 Code for Software Determined Switching Pattern

```
.data
*****
** Addresses of compare registers corresponding to the 1st-to toggle*
** channels in a given period indexed by the sector, s, Uout is in. *
*****
first_      .WORD  CMPR1      ;
            .WORD  CMPR2      ;
            .WORD  CMPR2      ;
            .WORD  CMPR3      ;
            .WORD  CMPR3      ;
            .WORD  CMPR1      ;

*****
** Addresses of compare registers corresponding to the 2nd-to toggle*
** channels in a given period indexed by the sector, s, Uout is in. *
*****
second_     .WORD  CMPR2      ;
            .WORD  CMPR1      ;
            .WORD  CMPR3      ;
            .WORD  CMPR2      ;
            .WORD  CMPR1      ;
            .WORD  CMPR3      ;

            .bss   temp0,1    ; temporary storage
            .bss   temp1,1    ; temporary storage
            .bss   temp2,1    ; temporary storage

.text
*****
** Initialize GP Timer 1 and full compare units for symmetric PWM **
*****
; Set GP Timer 1 period according to PWM period.
; GP Timer 1 period = PWM period/50nS/2: t1_period_
        SPLK    #t1_period_,T1PER
```



```

; Set GP Timer 1 mode.
; Put GP Timer 1 in continuous-up/down mode for symmetric PWM
      SPLK  #100010000000010b,T1CON

; Set PWM output polarities.
; PWM1,3&5 are active high. PWM2,4&6 are active low.
      SPLK  #0000011001100110b,ACTR

; Define and enable dead band.
; Set dead band to 1*32*50nS=1.6uS
      SPLK  #1f4h,DBTCON

; Enable PWM outputs and compare operation
      SPLK  #1000001000000111b,COMCON
;
;
*****
** Determine channel toggling sequence and load compare registers **
** Input: s(0-5)-sector number **
**      cmp_0(0.25C0Tp), cmp_1(0.5C1Tp), cmp_2(0.5C2Tp) **
** Output: compare values in compare registers CMPR1,2,3 **
*****
      LACC  #first_      ;
      ADD   s            ; point at entry in
                        ;      1st-to-toggle lookup table
      TBLR  temp0        ; get compare register addr of
                        ;      1st-to-toggle channel
      LAR   AR0, temp0   ; point at the compare register
      LACC  cmp_0        ; get cmp_0
      SACL  *            ; load compare register

      LACC  #second_    ;
      ADD   s            ; point at entry in
                        ;      2nd-to-toggle lookup table
      TBLR  temp1        ; get the compare register addr of
                        ;      2nd-to-toggle channel
      LAR   AR0,temp1   ; point at the compare register
      LACC  cmp_0        ;
      ADD   cmp_1        ; cmp_0+cmp_1
      SACL  *            ; load compare register

      LACC  #CMPR3      ;
      SUB   temp0        ;
      ADD   #CMPR2      ;
      SUB   temp1        ;
      ADD   #CMPR1      ;
      SACL  temp2        ; get the compare register
                        ;      addr of 3rd-to-toggle channel
      LAR   AR0, temp2  ; point at the compare register
      LACC  cmp_0        ;
      ADD   cmp_1        ;
      ADD   cmp_2        ; cmp_0+cmp_1+cmp_2
      SACL  *            ; load compare register

```



Notice that the compare values must be integers; i.e., their scaling must be $D15$ (or $Q0$). For this code example, a, b and c in Figure 4 represent, respectively, the state of the PWM1, 3 and 5 outputs and the polarities of these PWM channels are ACTIVE HIGH. Figure 6 shows the PWM outputs, i.e., the inverter outputs, of this PWM waveform pattern after the carrier has been taken out with a low-pass filter. The first and third waveforms in the figure are two of the three PWM outputs. The waveform in the middle is the difference between the two, representing the line-to-line inverter output voltage applied to a motor load. Figure 7 shows the same PWM outputs when dead band is enabled. The waveforms are essentially the same.

Figure 6. SV PWM Outputs With Carrier Filtered Out

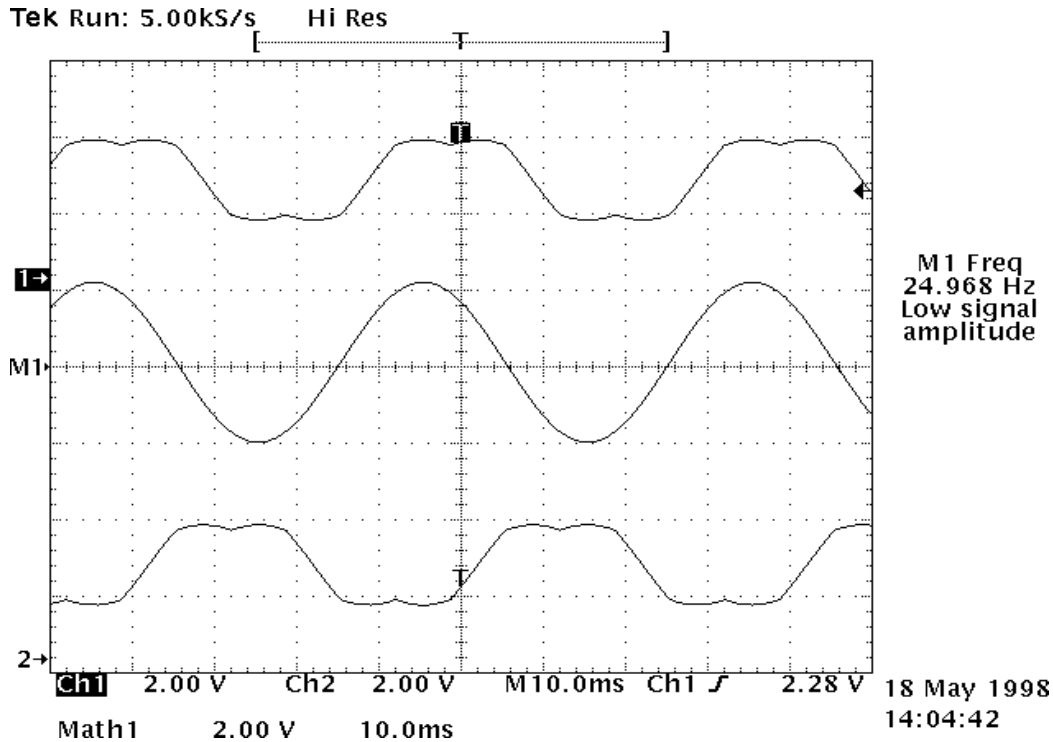
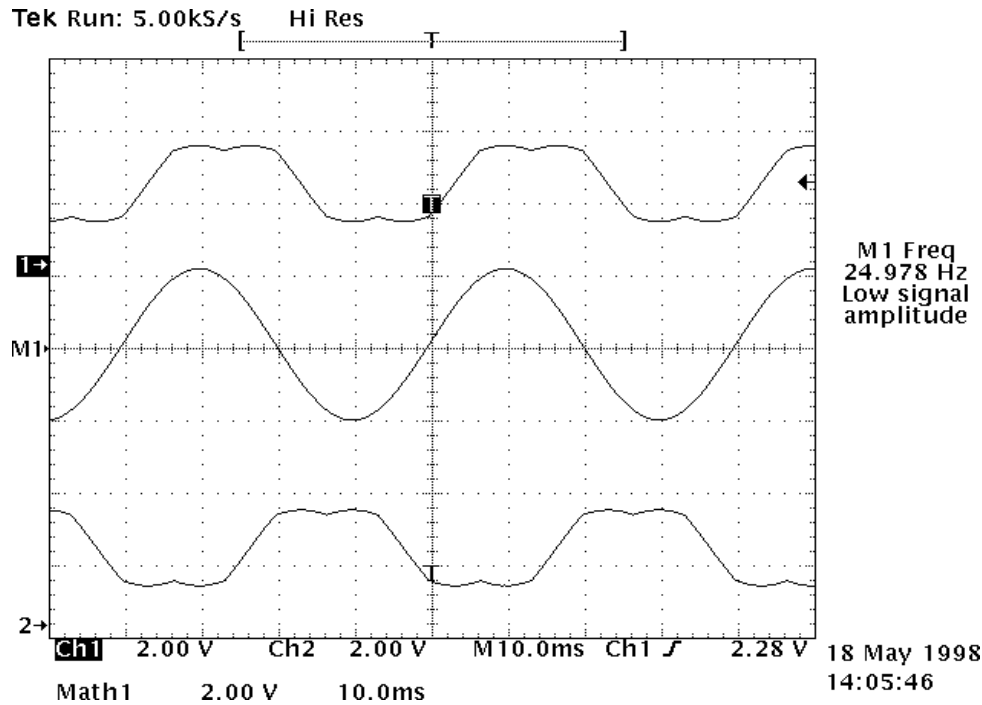


Figure 7. SV PWM Outputs With Carrier Filtered Out and Dead Band Enabled



Hardware-Implemented Switching Pattern

Figure 8 below shows two symmetric switching patterns implemented by the SV PWM hardware module on the TMS320C24x/F24x for each sector. The rule of these switching patterns can be summarized as $(U_x, U_{x\pm 60}, O_{yyy}, O_{yyy}, U_{x\pm 60}, U_x)$, where O_{yyy} can be O_{000} or O_{111} , whichever differs from $U_{x\pm 60}$ by the state of only one channel, and x can be 0, 60, 120, 180, 240, or 300. The following are some remarks about this switching scheme:

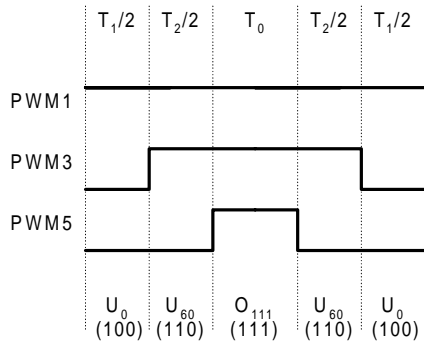
- There is always a channel staying constant for the entire PWM period. So the number of switching times for this scheme is less than the software-determined scheme. The obvious result of this is reduced switching losses.
- For the type of application addressed, dead band is necessary between the complimentary pairs of PWM channels, i.e., PWM1 and 2, PWM3 and 4, and PWM5 and 6 on the TMS320C24x/F24x to avoid shoot-through faults. Dead band is inserted only when there is a transition to turn off one device and turn on the other device on the same inverter leg.

Therefore dead band does not affect the channel that stays unchanged. Since the same channel may stay unchanged for the entire sector, this may be true for a long time duration depending on the commanding frequency. As a result, the dead band will affect the three PWM outputs unevenly, resulting in small harmonics in the inverter line-to-line outputs. Depending on the application, this drawback may or may not be an important issue.

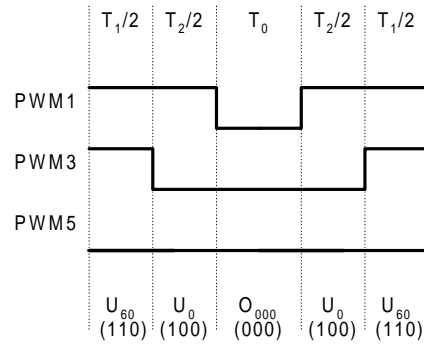
- The two switching patterns for each sector are results of two switching directions. Theoretically, different switching directions can be combined in different ways to obtain a composite switching order. However, no advantage has been observed until now to use a composite order other than maintaining a constant direction for all the sectors.



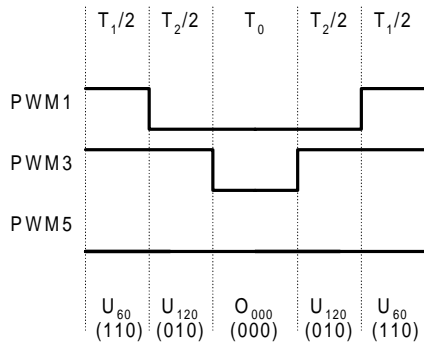
Figure 8. Hardware-implemented SV PWM Waveform Pattern



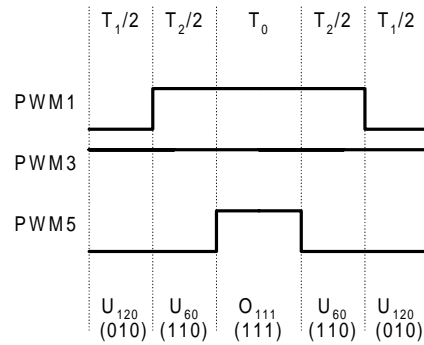
U_{out} in sector of $U_0 - U_{60}$,
SVRDIR=0, $(D_2 D_1 D_0)=(001)$



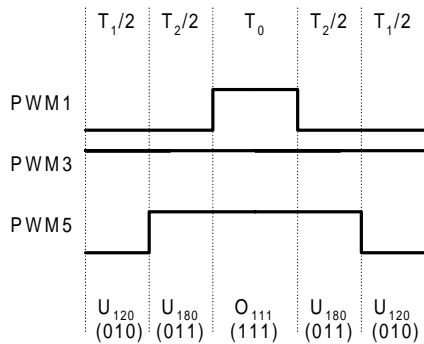
U_{out} in sector of $U_0 - U_{60}$,
SVRDIR=1, $(D_2 D_1 D_0)=(011)$



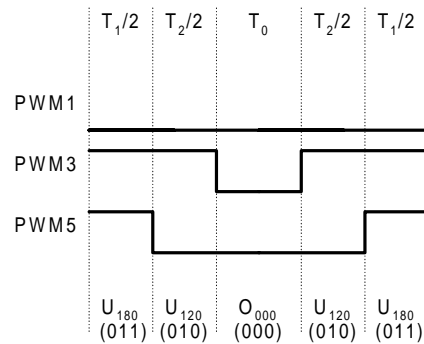
U_{out} in sector of $U_{60} - U_{120}$,
SVRDIR=0, $(D_2 D_1 D_0)=(011)$



U_{out} in sector of $U_{60} - U_{120}$,
SVRDIR=1, $(D_2 D_1 D_0)=(010)$



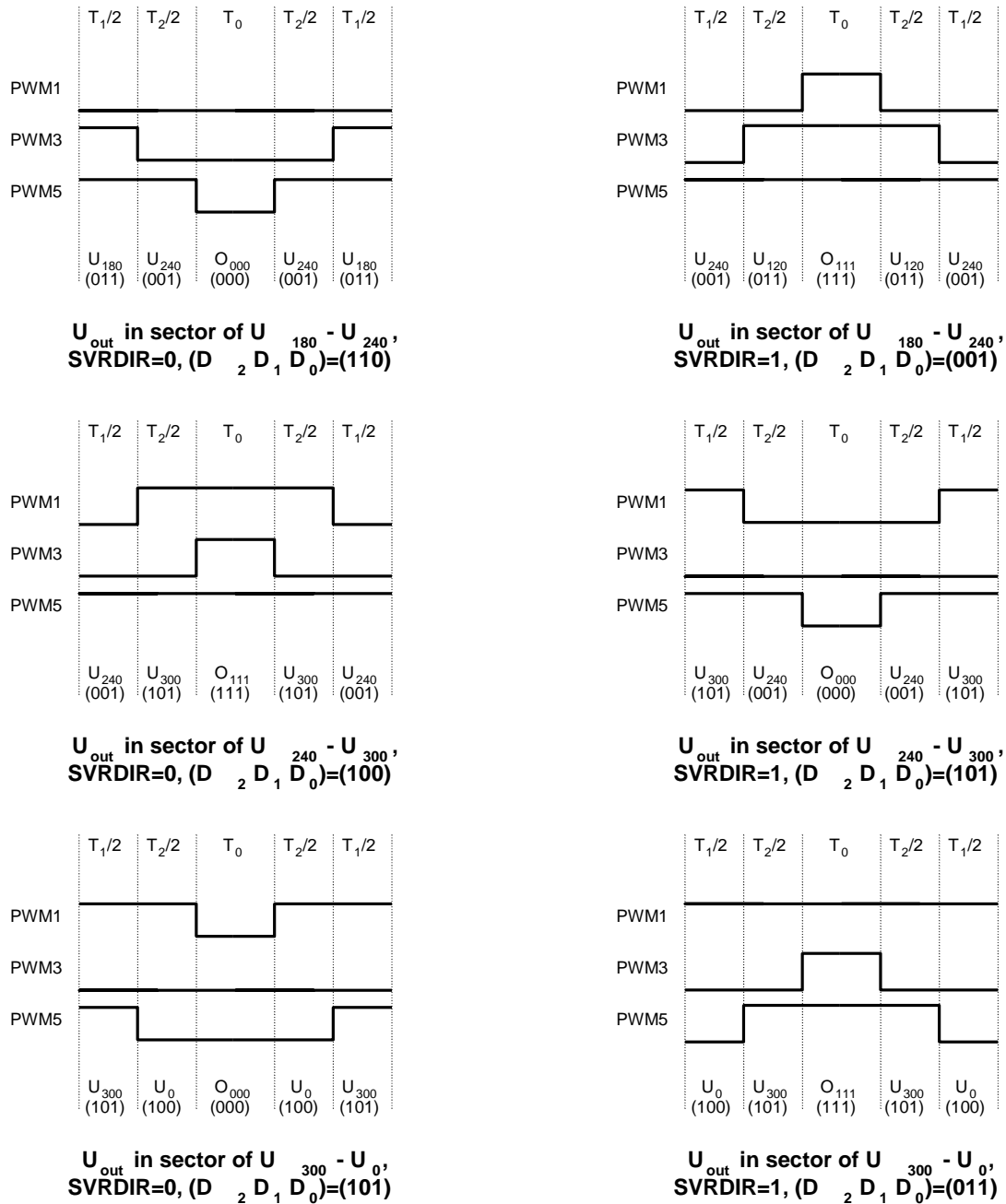
U_{out} in sector of $U_{120} - U_{180}$,
SVRDIR=0, $(D_2 D_1 D_0)=(010)$



U_{out} in sector of $U_{60} - U_{120}$,
SVRDIR=1, $(D_2 D_1 D_0)=(110)$



Figure 8. Hardware-implemented SV PWM Waveform Pattern (continued)



The SV PWM hardware on the TMS320C24x/F24x requires the application software to generate U_{out} , determine that the sector U_{out} is in, and perform the decomposition to get T_1 and T_2 (in terms of timer counts) for each PWM period. Then, for each PWM period, the software only needs to accomplish the following steps:

- 1) Load the binary bit pattern corresponding to the starting basic space vector into bits[12-14] of the Action Control Register (ACTR), and the switching direction into bit[15] of ACTR, with 0 representing anti-clockwise and 1 representing clockwise.



- 2) Load $0.5 \cdot T_1$ into the Full Compare Register 1 (CMPR1) and $0.5 \cdot T_1 + 0.5 \cdot T_2$ into the Full Compare Register 2 (CMPR2).

The state machine in the hardware SV PWM logic monitors the register configurations and respective compare matches for the whole PWM period and does what is necessary to generate the waveform patterns in Figure 5 according to the configuration. Therefore, implementation of SV PWM becomes even simpler with the help of the hardware SV PWM module, as shown in the following TMS320C24x/F24x code example:

Example 3. TMS320F243/1 Code Example Using the Hardware SV PWM Module

```

                .data
*****
** Lookup table for ACTR[15-12] for SV pulse-width    **
** modulation when the direction is clockwise, indexed by sector **
** number                                             **
*****
clkwise_      .WORD 1011000000000000b
               .WORD 1010000000000000b
               .WORD 1110000000000000b
               .WORD 1100000000000000b
               .WORD 1101000000000000b
               .WORD 1001000000000000b

*****
** Lookup table for ACTR[15-12] for SV pulse-width    **
** modulation when the direction is clockwise indexed by sector **
** number                                             **
*****
cckwise_      .WORD 0001000000000000b
               .WORD 0011000000000000b
               .WORD 0010000000000000b
               .WORD 0110000000000000b
               .WORD 0100000000000000b
               .WORD 0101000000000000b

               .bss svpat,1           ; temporary storage

                .text
*****
** Initialize GP Timer 1 and full compare units for symmetric PWM **
*****

; Set GP Timer 1 period according to PWM period.
; GP Timer 1 period = PWM period/50nS/2: t1_period_
        SPLK    #t1_period_,T1PER

; Set GP Timer 1 mode.
; Put GP Timer 1 in continuous-up/down mode for symmetric PWM
        SPLK    #100010000000010b,T1CON

; Set PWM output polarities.
; PWM1,3&5 are active high. PWM2,4&6 are active low.
        SPLK    #0000011001100110b,ACTR

; Define and enable dead band.
; Set dead band to 1*32*50nS=1.6uS

```



```

                SPLK    #1f4h, DBTCON

; Enable PWM outputs and compare operation
                SPLK    #1000001100000111b, COMCON
;
;
;

*****
** Determine ACTR pattern and reload ACTR and CMPR1&2          **
** based on sector, s, Uout is in.                            **
** Input: s(0-5)-sector number                                **
**      t1_period_: timer period (for PWM freq)              **
** Output: compare values in compare registers CMPR1,2       **
*****
                LACC    #cckwise_
                ADD     s          ; point to entry in lookup table
                TBLR    svpat      ; get the pattern
                LAR     AR0, #ACTR  ; point to ACTR
                LACC    *          ; read ACTR
                AND     #0FFFh     ; clear sv pattern bits
                OR      svpat      ; re-configure sv pattern bits
                SACL    *          ; re-load ACTR

                LAR     AR0, #CMPR1 ; point to CMPR1
                LACC    cmp_1      ;
                SACL    *+         ; cmp_1=>CMPR1, point to CMPR2

                ADD     cmp_2      ;
                SACL    *          ; cmp_2=>CMPR2

                SUB     #t1_period_ ; limit CMPR2
                BLEZ    in_lmt     ;
                SPLK    # t1_period _, * ;

in_lmt

```

Figure 9 shows the PWM outputs of this waveform pattern after the carrier is taken out with a low-pass filter. Again, the first and third waveforms are two of the three PWM outputs. The waveform in the middle is the difference between the two PWM outputs, representing the line-to-line inverter voltage output applied to a motor load. Figure 10 shows the same PWM outputs when dead band is enabled. The effects of dead band imbalance are seen as distortion or harmonics in the line-to-line inverter voltage output in Figure 10. This distortion can become significant when the dead band is big with respect to the magnitude of inverter voltage output.



Figure 9. SV PWM Outputs With Carrier Filtered Out

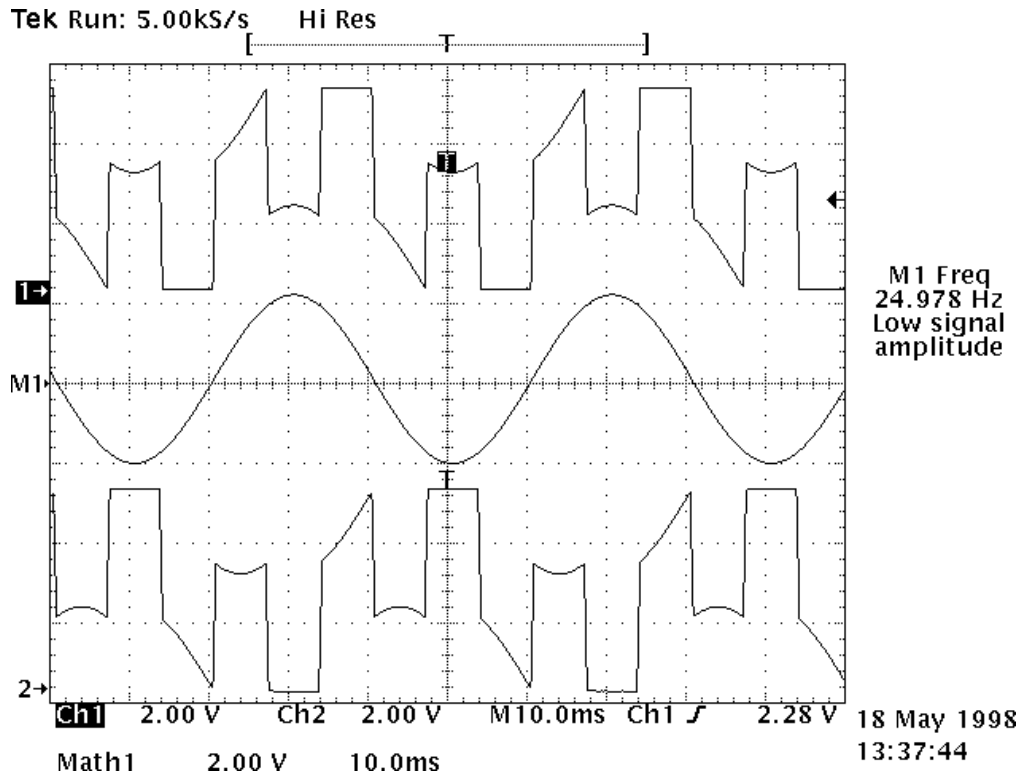
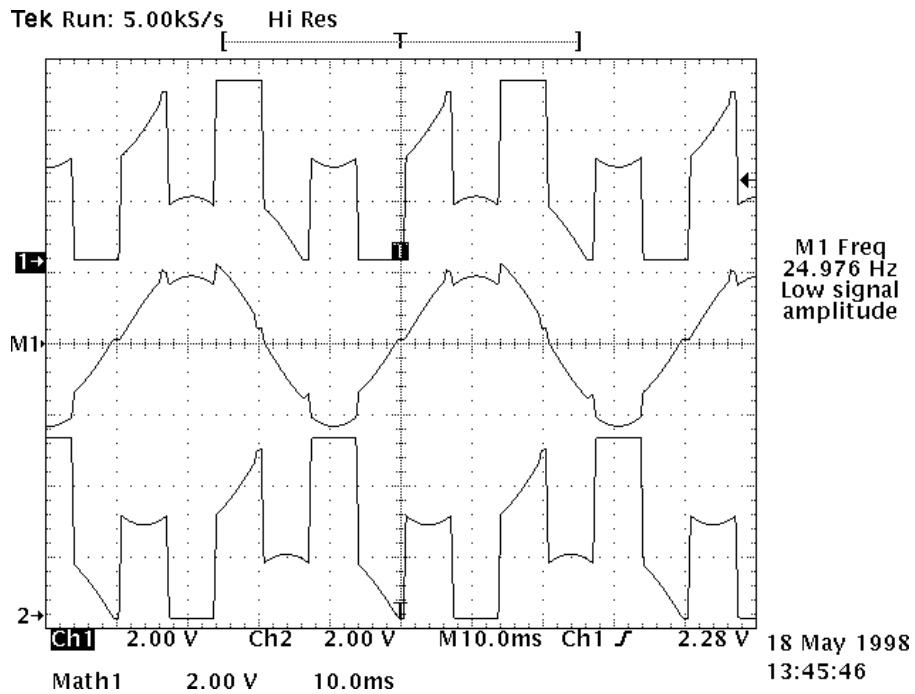


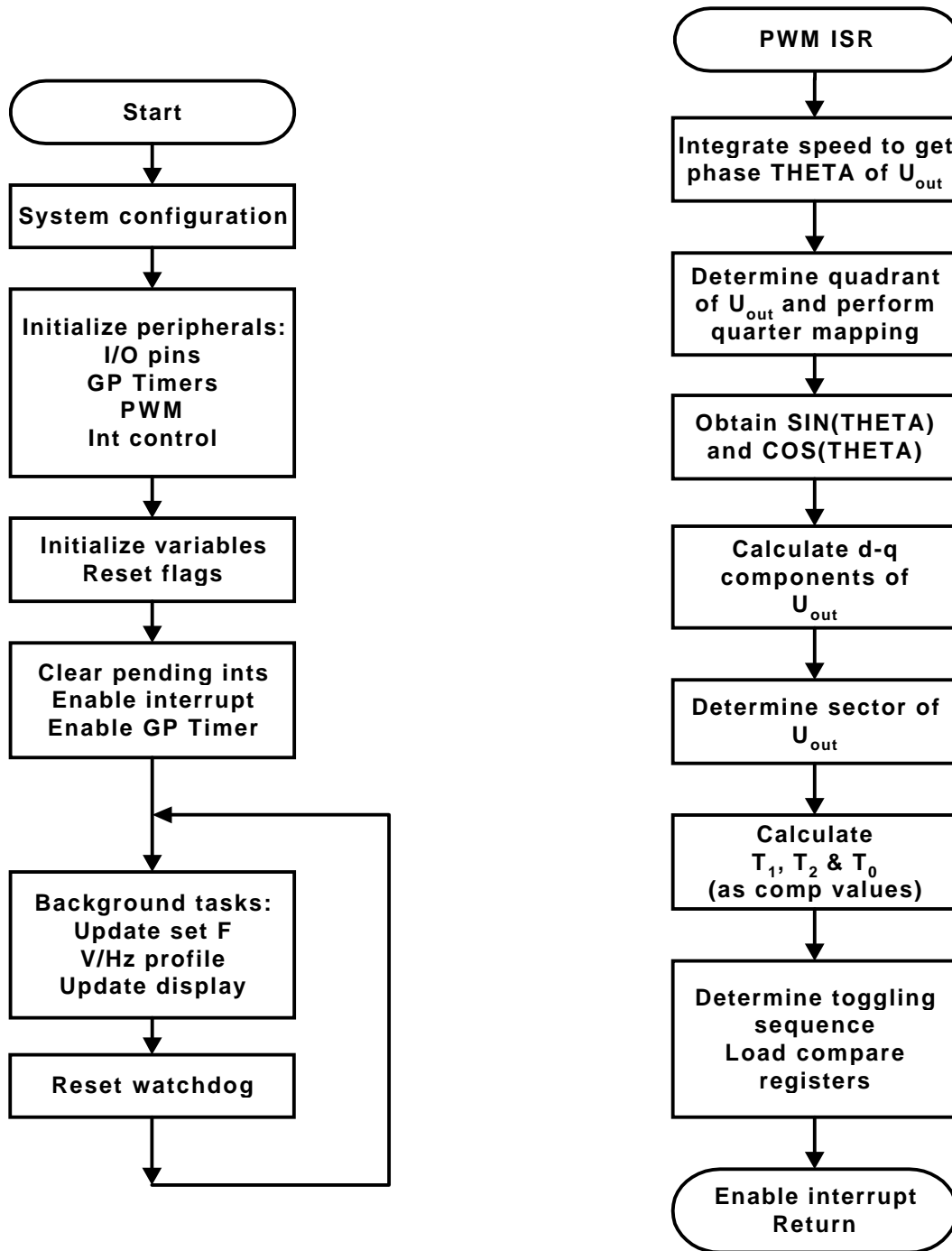
Figure 10. SV PWM Outputs With Carrier Filtered Out and Dead Band Enabled



Application in Three-Phase AC Induction Motor Control

An example of the application of SV PWM techniques in open-loop three-phase AC induction motor control is described in this section. Figure 11 shows the program flow chart of the example.

Figure 11. Program Flow Chart





The following are steps in the main program:

- 1) Configure the timers and compare units for symmetric PWM.
- 2) Input desired speed.
- 3) Obtain the magnitude of reference voltage vector U_{out} (based on constant V/Hz profile).
- 4) Update display, reset watchdog timer and loop back to 2.

The following are the steps in the interrupt driven SV PWM routine:

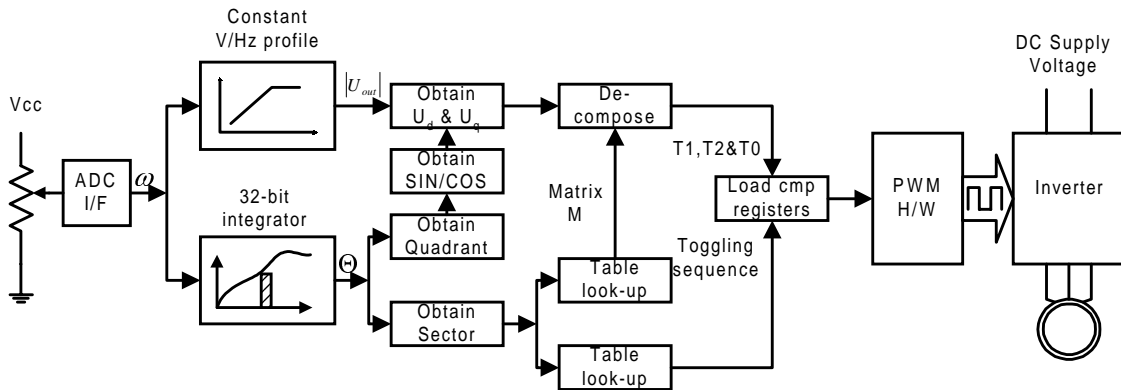
- 1) Obtain the phase (θ) of U_{out} by integrating the command speed.
- 2) Obtain the sine and cosine of θ with quarter mapping and table look-up, and calculate the $d-q$ component of U_{out} .
- 3) Determine which sector U_{out} is in.
- 4) Decompose U_{out} to obtain T_1 , T_2 and T_0 as compare values.
- 5) Determine the switching pattern (for hardware approach) or sequence (for software approach) and load the obtained compare values into corresponding compare registers.

The major features of this implementation are:

- 32-bit integration to obtain the phase of the reference voltage vector
- Quarter mapping to calculate SIN and COS functions
- Sector-based table look-up for decomposition matrix
- Sector-based table look-up for channel toggling order or Action Control Register reload pattern
- 20-KHz PWM and sampling frequency

The block diagram of the implementation is shown in Figure 12. The on-line background program takes about 4 μ s of CPU time. The interrupt driven SV PWM routine takes about 9 μ s for the software determined switching pattern and about 8.5 μ s for the hardware implemented switching pattern. The difference in code size is about 10 instruction words.

Figure 12. Block Diagram of an Open-Loop AC Induction Motor Control System



Experimental Results

Experimental results are presented below to demonstrate the effectiveness of the discussed algorithms. Figure 13 is the motor current waveform and spectrum obtained with the first scheme, which we call the software approach. Figure 14 is the motor current waveform and spectrum obtained with the second scheme, which makes use of the hardware SV PWM module. The inverter, LabDrive, used in the experiments is from Spectrum Digital. The inverter is interfaced with a TMS320F243 EVM on which the motor control program runs. A motor with a fan on the shaft was used as the load in the experiments. The motor is a 4-pole, 3-phase AC induction motor rated at 60 Hz, 144 V and 1/3 hp. It can be seen that little or no harmonics are present in the current spectrums, demonstrating the effectiveness of the implemented SV PWM technique.

Conclusions

It has been shown that the SV PWM technique utilizes DC bus voltage more efficiently and generates less harmonic distortion in a three-phase voltage-source inverter. This document has presented an overview of SV PWM theory and two ways of SV PWM implementation. Program examples for both approaches are given for Texas Instrument's TMS320C24x/F24x DSP controllers. The approach implemented with the hardware SV PWM module on TMS320C24x/F24x reduces the number of switching times as compared with the software-based approach. The direct result of this is switching reduced losses, which may become significant if the power rating of the inverter is high. Experimental results proved both implementations to be very effective.



Figure 13. Motor Current and Spectrum Obtained With the Software Approach

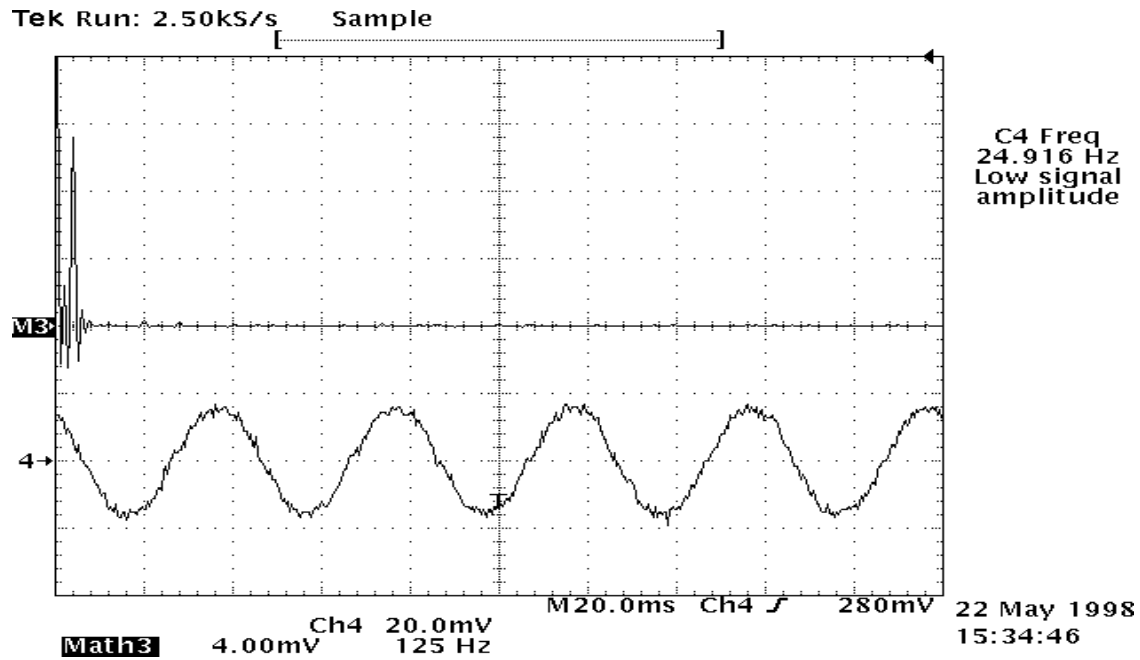
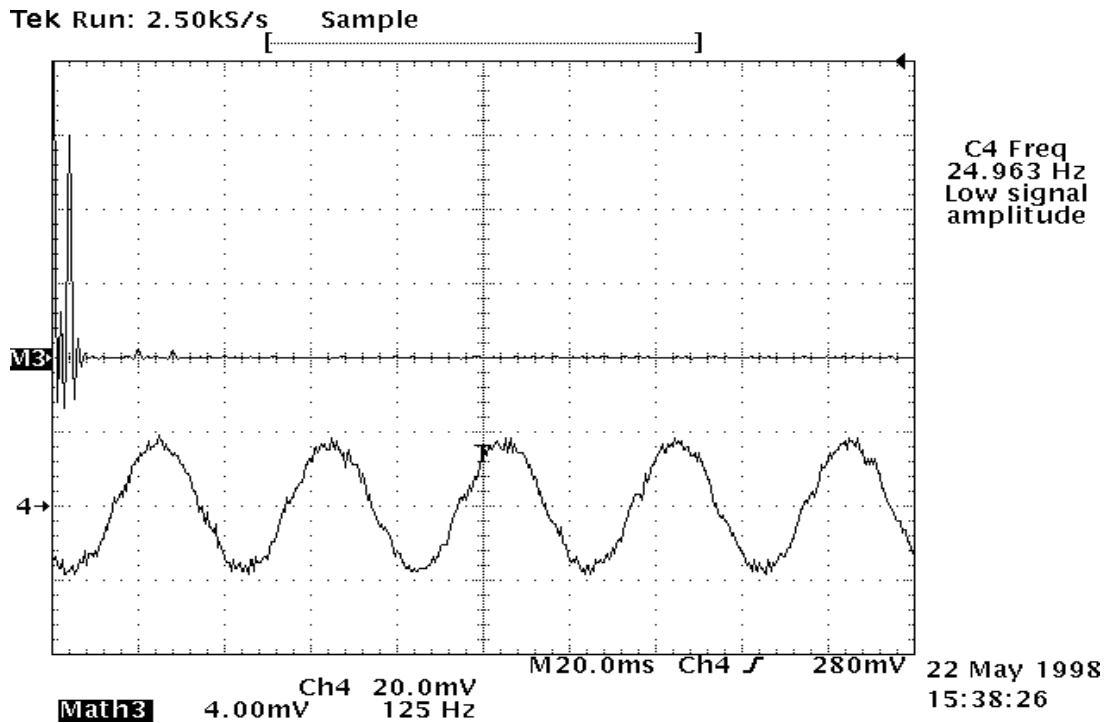


Figure 14. Motor Current and Spectrum Obtained With the Hardware Approach





References

- 1) Trzynadlowski, A. M.; *The Field Orientation Principle in Control of Induction Motors*; Kluwer Academic, 1994.
- 2) Trzynadlowski, A. M., Kirlin, L, and Legowski, S. F.; "Space-Vector PWM Technique With Minimum Switching Losses and a Variable Pulse Rate;" *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 2, April 1997.
- 3) Trzynadlowski, A. M.; "An Overview of Modern PWM Techniques for Three-Phase, Voltage-Controlled, Voltage-Source Inverter;" *International Symposium on Industrial Electronics* 1996.
- 4) Ogasawara, S., Akagi, H. et al; "A Novel PWM Scheme of Voltage Source Inverters Based on Space Vector Theory;" *EPE*, Aachen, 1989.
- 5) Zhenyu Yu, Figoli, David; *AC Induction Motor Control Using Constant V/Hz Principle and Space-Vector PWM Technique With TMS320C240*; Texas Instruments Literature Number SPRA284.
- 6) Van der Broeck, F. G., Skudelny, H. C., Stanke, G.; "Analysis And Realization of a Pulse Width Modulator Based on Voltage Space Vectors;" *IEEE Transactions on Industrial Applications*, vol. IA-24, no.1, 1988, pp.142-150.
- 7) Stefanovic, V. R.; *Space-Vector PWM Voltage Control With Optimized Switching Strategy*; IEEE/IAS Annual Meeting, pp.1025-1033, 1992.
- 8) Boglietti A., Griva G., Pastorelli M., Portumo F., Adam T.; *Different PWM Modulation Techniques Indexes Performance Evaluation*; IEEE International Symposium on Industrial Electronics, June1-3, 1993, Budapest, Hungary, pp.193-199.
- 9) Mallinson, N.; "Plug & Play" *Single Chip Controllers for Variable Speed Induction Motor Drivers in White Goods and HVAC Systems*; 1998 IEEE Applied Power Electronics Conference.
- 10) Lai, Y-S, and Bowes, S. R.; "A Novel High Frequency Universal Space-Vector Modulation Control Technique;" *Proceedings of 1997 International Conference on Power Electronics and Drive Systems*, 1997, pp. 510-507.
- 11) Bowes, S. R., and Lai, Y-S; "The Relationship Between Space-vector Modulation and Regular-Sampled PWM;" *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 5, October 1997.
- 12) Liu, Y-H, Chen, C-L, and Tu, R-J; "A Novel Space-Vector Current Regulation Scheme for a Field-Oriented-Controlled Induction Motor Drive;" *IEEE Transactions on Industrial Electronics*, Vol. 45, No. 5, October 1998.
- 13) Tzou, Y-Y, and Hsu, H-J; "FPGA Realization of Space-Vector PWM Control IC for Three-Phase PWM Inverters;" *IEEE Transactions On Power Electronics*, Vol. 12, No. 6, November 1997.



Appendix A. Program for Open-Loop Three-Phase AC Induction Motor Control With SV PWM Technique and Constant V/Hz Principle

```

*****
** File Name      : sv20b.asm                      **
** Project       : ACI motor control              **
** Originator    : Zhenyu Yu                     **
**              : Texas Instruments              **
**              : DSP Digital Control Systems Applications **
** Target        : TMS320F243 EVM + SD i/f + SD inverter **
*****
; Description
;-----
; This program implements an open-loop speed control algorithm for
; three-phase AC induction motors using constant v/f principle and
; SV PWM technique. The program allows the usage of either
; h/w or s/w determined switching patterns by changing the assembly
; directives.
;*****
; Status          : Worked correctly
; Last update    : 2/1/99
;-----
; Notes
;-----
; 1. This program implements an INT driven sampling and control loop
;    for three-phase AC induction motor control through a three-phase
;    voltage source inverter.
; 2. Constant v/f principle is used to generate the magnitude of
;    reference voltage from frequency input.
; 3. SV PWM technique is employed to generate a sinusoidal
;    type of three-phase voltage output from the inverter.
; 4. Both PWM and sampling frequencies are 20KHz.
; 5. Maximum scaling and 32 bit integration are used to maximize the
;    accuracy of integer math.
; 6. The motor is assumed to be rated at 60Hz, i.e., maximum voltage
;    output magnitude is achieved when freq (speed) input is 60Hz.
; 7. Frequency input is through push buttons connected to the IOPB6
;    (UP) and IOPB7 (DOWN). Frequency range is 0-120Hz.
; 8. The D scaling notation used here is related to the popular Q
;    scaling notation by Dx=Q15-x.

;=====
; Switching pattern
;-----
SWPAT      .set      0
HWPAT      .set      1
; -- Comment in one at a time
SVPAT      .set      SWPAT      ; Comment in to use s/w pattern
;SVPAT     .set      HWPAT      ; Comment in to use h/w pattern

;-----
; Processor
;-----
F241       .set      1
  
```



```

F243          .set    3
; -- Comment in one at a time
;DEVICE       .set    F241          ; Comment in for F241
DEVICE       .set    F243          ; Comment in for F243

;-----
; Peripheral Registers and constants of TMS320C24x/F24x
;-----
                .include "X24x.h"    ; X24x register addresses
ST0           .set    0              ; status register ST0
ST1           .set    1              ; status register ST1
wd_rst_1     .set    055h           ; watchdog timer reset string
wd_rst_2     .set    0aah           ; watchdog timer reset string
LED_ADDR     .set    0ch            ; addr of LED display on EVM

;-----
; Variables
;-----
                .bss    error,1      ; Number of errors
                .bss    temp,1       ; temporary storage
                .bss    one,1        ; +1
                .bss    upbutcntr,1  ; UP count push button
                .bss    dnbutcntr,1  ; Down count push button
                .bss    set_f,1      ; set F: D0 (-1.0-1.0, 1.0-120Hz)
                .bss    f_omega,1    ; set F to angular speed ratio: D10
                .bss    omega,1      ; set angular speed: D10
                .bss    omega_v,1    ; angular speed to voltage ratio:D-9
                .bss    set_v,1      ; set voltage: D1
                .bss    t_sample,1   ; sampling period: D-9
                .bss    theta_h,1    ; phase of ref vector hi word: D3
                .bss    theta_l,1    ; theta lo word
                .bss    theta_r,1    ; rounded theta_h: D3
                .bss    theta_m,1    ; theta mapped to 1st quadrant: D3
                .bss    theta_i,1    ; theta to index for sine table: D6
                .bss    SS,1         ; sin sign modification: D15
                .bss    SC,1         ; cos sign modification: D15
                .bss    sin_indx,1   ; index to sine table: D15
                .bss    sin_entry,1  ; beginning of sin table
                .bss    sin_end,1    ; end of sin table
                .bss    sin_theta,1  ; sin(theta): D1
                .bss    cos_theta,1  ; cos(theta): D1
                .bss    Ud,1         ; voltage Ud: D2
                .bss    Uq,1         ; voltage Uq: D2
                .bss    theta_s,1    ; theta to sector mapping: D0
                .bss    sector,1     ; sector reference U is in: D15
                .bss    theta_90,1   ; 90: D3
                .bss    theta_180,1  ; 180: D3
                .bss    theta_270,1  ; 270: D3
                .bss    theta_360,1  ; 360: D3
                .bss    dec_ms,24    ; Decomposition matrices: D1
                .bss    t1_periods,1 ; scaled Timer 1 period: D10
                .bss    cmp_1,1      ; decomp on 1st basic sp vector: D15
                .bss    cmp_2,1      ; decomp on 2nd basic sp vector: D15
                .bss    cmp_0,1      ; decomp on 0 basic sp vector /2: D15
                .bss    first_tog,1  ; the 1st-to-toggle channel
                .bss    sec_tog,1    ; the 2nd-to-toggle channel
                .bss    svpat,1      ; S/V pattern for ACTR
                .bss    led_dir,1    ; LED direction (1: left, 0: right)

```



```

        .bss  led_data,1      ; LED display
led_freq_ .set  3000         ; LED update sub-divider
        .bss  led_subdvdr,1 ; sub-divider counter for led

;-----
; Context
;-----
ST0_save .usect ".context",1 ; saved status register ST0
ST1_save .usect ".context",1 ; saved status register ST1
ACCH     .usect ".extcont",1 ; saved accumulator high
ACCL     .usect ".extcont",1 ; saved accumulator low
AR0_save .usect ".extcont",1 ; saved AR0 content
AR1_save .usect ".extcont",1 ; saved AR1 content
P_hi     .usect ".extcont",1 ; saved P high byte
P_lo     .usect ".extcont",1 ; saved P low byte
T_save   .usect ".extcont",1 ; saved T content

;-----
; Program parameters
;-----
debug_data .set  3FFFh      ; 60Hz-3FFF, 30Hz-1FFF, 25Hz-1AAB

; Scaled sampling period
; Ts*D-9=Ts*2**24, Ts=50uS
t_sample_ .set  0346h      ; D-9

; Set frequency to radian frequency conversion ratio
; 120*2*pi/7FFFh/D0 = 754.0052472756
; 7FFFh corresponds to 120Hz=753.9822368616 rad/sec
f_omega_ .set  24128       ; D10

; Minimum radian frequency
; min_F*2*pi*D10=12*2*pi*D10=75.39822368616*D10
; min_F=12Hz is the minimum frequency input, D10=2**5
min_omega_ .set  2413      ; D10

; Radian frequency to ref voltage conversion ratio -> V/Hz constant
; 1.0/sqrt(2)/(60*2*pi)*D24 = 0.001875658991994*D24
omega_v_ .set  31468       ; D-9

; Max magnitude of reference voltage
; 1.0/sqrt(2)*D1 = 0.7071067811865*D1
max_v_ .set  11585        ; D1. 1b less res to reduce # shiftingsa

; Min magnitude of reference voltage given by
; 1.0/sqrt(2)*min_F/60Hz*D1 = 0.1414213562373*D1
min_v_ .set  2317         ; D1

; Conversion from theta to index for sine table
; 360/(0.5pi)*D8, D8=2**(15-8)=2**7 ; 360 entry sine table
; theta_i_ .set  29335      ; D8
; 90/(0.5pi)*D6, D6=2**9 ; 90 entry sine table
theta_i_ .set  29335      ; D6

; Conversion from theta to sector
; 6/(2*pi)*D0, D0=2**(15-0)
theta_s_ .set  31291      ; D0

```



```

; No of cycles needed to qualify a button push
but_qual_ .set 100 ; 20*t_sample

        .sect ".vectors"
;=====
; Reset and interrupt vector table
;-----
RESET      B  _c_int0   ; PM 0Reset Vector
INT1       B  _c_int1   ; PM 2Int level 1
INT2       B  _c_int2   ; PM 4Int level 2
INT3       B  INT3     ; PM 6Int level 3
INT4       B  INT4     ; PM 8Int level 4
INT5       B  INT5     ; PM AInt level 5
INT6       B  INT6     ; PM CInt level 6

        .text
;=====
; Start of main body of code
;-----
_c_int0    DINT          ; Set global interrupt mask

        .ifDEVICE=F243
cfg_wsggr  LDP   #temp    ; Configure WSGR
           SPLK  #0,temp  ; temp<=0
           OUT  temp,0ffffh ; WSGR <= (temp)
        .endif

reset_wd0  LDP   #WDKEY>>7 ; Reset WD timer
           SPLK  #wd_rst_1,WDKEY
           SPLK  #wd_rst_2,WDKEY
           SPLK  #01101111b,WDCR

; Configure Shared Pins
; Group A shared pins all used for primary functions except TDR/IOPB6
; and TLKIN/IOPB7 used as UP and DN on SD platform
; Group B shared pins all used as default.
; SPISIMO/IOPC2, SPISOMI/IOPC3 used as digital output timing
; marks XF/IOPC2 as dr fault clr, BIO/IOPC1 as dr enable in,
; SPISTE/IOPC5 as dr reset IOPD4 as dr enable

cfg_pins   LDP   #OCRA>>7
           SPLK  #03FFFh,OCRA
           SPLK  #0,OCRB
           splk  #02C00h,PCDATDIR

t1_period_ .set 500      ; Tpwm/50nS/2=50uS/50nS/2=500
t1_periods_ .set 500*32 ; D10, scaled Timer 1 period
t2_period_ .set 1000    ;

init_ev    ldp   #T1CMPR>>7 ; set DP
           splk  #10,T1CMPR  ; Init GPT comp registers
           splk  #10,T2CMPR  ;
           SPLK  #t1_period_,T1PR ; Init GPT1 period reg
           SPLK  #t2_period_,T2PR ; Init GPT2 period reg
           splk  #0000001010101b,GPTCON
           ; set timer comps to active low
           SPLK  #1000100000000010b,T1CON

```



```

; Set GPT1 in Up/Dn mode for symm PWM
SPLK #1000100010000010b,T2CON
; Set GPT2 in Up/Dn mode

SPLK #t1_period_,CMPR1
; Init. PWM duty cycle to 0%
SPLK #t1_period_,CMPR2
SPLK #t1_period_,CMPR3
SPLK #0000011001100110b,ACTR ; Cfg PWM outputs
SPLK #01F4h,DBTCON ; Cfg deadband 1*32*50nS=1.6uS

.if SVPAT=SWPAT
SPLK #1000001000000000b,COMCON
; Enbl PWM outputs&cmp opera
.endif

.if SVPAT=HWPAT
SPLK #1001001000000000b,COMCON
; Enbl PWM outputs&cmp op&svpwm
.endif

init_vars LDP #error ; Point to B1 page 0
SPLK #0,error ; Reset error counter
SPLK #1,one ; +1 => one
SPLK #t_sample_,t_sample ; sampling period
SPLK #t1_periods_,t1_periods ; max compare value
splk #0,set_f ; zero set F.
splk #0,upbutcntr ; zero up count
splk #0,dnbutcntr ; zero down count
SPLK #f_omega_,f_omega
; set F to angular speed ratio
SPLK #omega_v_,omega_v
; angular speed to voltage ratio
SPLK #0,theta_l ; theta low byte
SPLK #0,theta_h ; theta high byte

LAR AR0,#theta_90 ; point to 1st destination
LAR AR1,#(28-1) ; 32 entries
LACC #angles_ ; point to 1st data item
LARP AR0 ;
init_tbl TBLR *,,1 ; move and point to next destination
ADD one ; point to next data item
BANZ init_tbl,0
splk #theta_i_,theta_i ; theta to sin_index ratio
splk #theta_s_,theta_s ; theta to sector ratio
SPLK #sin_entry_,sin_entry
; init 1st and last entries of sin tb
SPLK #(sin_entry_+90),sin_end

.if DEVICE=F243
splk #1,led_data ; Reset LED display on EVM
out led_data,LED_ADDR ; Set LED display
splk #led_freq_,led_subdvdr
; reset sub-divider counter
splk #1,led_dir ; set LED display direction
.endif

ldp #_OVERCURRENT_TRIP_FLAG ; set DP

```



```

splk  #0,_OVERCURRENT_TRIP_FLAG
                                           ; reset overcurrent flag

ldp  #phantom_int
splk  #0,phantom_int      ;

enbl_sd      LDP      #PCDATDIR>>7      ; Enable SD inverter
lacc  PCDATDIR           ;
or    #00020h           ;
sac1  PCDATDIR           ; pull LabDrive reset high
lacc  PDDATDIR           ;
or    #01000h           ;
and   #0FFEFh           ;
sac1  PDDATDIR           ; Enable LabDrive

enbl_pwmg    LDP      #T1CON>>7        ; Enable GPT1 and PWM'ing
SPLK  #1000100001000010b,T1CON

cfg_ints     ldp      #EVIFRA>>7      ; Cfg interrupts
SPLK  #0ffh,EVIFRA      ; Clear all Group A
                                           ; interrupt flags
SPLK  #0ffh,EVIFRB      ; Clear all Group B
                                           ; interrupt flags
SPLK  #0fh,EVIFRC       ; Clear all Group C
                                           ; interrupt flags
SPLK  #0201h,EVIMRA     ; Mask all but GPT1 UF&PDPINT
                                           ; Group A ints
SPLK  #0,EVIMRB         ; Mask all ints
SPLK  #0,EVIMRC         ; Mask all Grp C ints
LDP   #0                ; point to memory page 0
SPLK  #0ffh,IFR         ; Clear all core interrupt flags
splk  #00001111b,IMR    ; Unmask all EV
                                           ; interrupts+INT1 to CPU
EINT                                           ; Enable global interrupt

;=====
; Start of background loop
;-----
main_loop    ldp      #PCDATDIR>>7      ; set DP
lacc  PCDATDIR           ;
and   #0FFFBh           ; IOPC[2] to 0
sac1  PCDATDIR           ;

update_f     ldp      #PBDATDIR>>7      ;
BIT   PBDATDIR,BIT6     ; Has UP been pushed?
Ldp   #upbutcntr        ; point at page 0 of B1
BCND  up_butn,TC        ; UP button if yes
SPLK  #0,upbutcntr      ; Clear UP count if no

Ldp   #PBDATDIR>>7      ; point at sys reg page 1
BIT   PBDATDIR,BIT7     ; Has DN been pushed?
ldp   #dnbutcntr        ; point at page 0 of B1
BCND  dn_butn,TC        ; DN button if yes?
SPLK  #0,dnbutcntr      ; Cleat DN count if no
B     pbutnend          ; Return

up_butn     LACC  upbutcntr          ; Inc. UP count
ADD    one

```



```

SACL  upbutcntr
SUB   #but_qual_      ; Qualified?
BCND  pbutnend,LT    ; Return if no
SPLK  #0,upbutcntr   ; Reset count if yes &
LACC  set_f          ; Inc set frequency
ADD   one
SACL  set_f
SUB   #7fe0h         ; Bigger than max?
BCND  pbutnend,LEQ   ; Return if no
SPLK  #7fe0h,set_f   ; Saturate if yes &
B     pbutnend       ; return

dn_butn    LACC  dnbutcntr      ; Inc. DN count
           ADD   one
           SACL  dnbutcntr
           SUB   #but_qual_     ; Qualified?
           BCND  pbutnend, LT   ; Return if not
           SPLK  #0,dnbutcntr   ; Reset count if yes &
           LACC  set_f          ; Dec set frequency
           SUB   one
           SACL  set_f
           BCND  pbutnend,GEQ   ; Return if no
           SPLK  #0,set_f       ; Saturate if yes &

pbutnend

; Comment out following line to use push button to control speed
SPLK  #debug_data,set_f ; Replace with debug data

f2omega    LT   set_f          ; set f -> omega: D0
           MPY  f_omega        ; D0*D10=D(10+1)
           PAC                ; product -> ACC: D11
           SACH  omega,1       ; -> set angular speed: D10
           lacc  omega         ;
           sub   #min_omega_   ; compare W with its lower limit
           BGZ  winlimit       ; continue if within limit
           splk  #min_omega_,omega ; saturate if not

winlimit

; Note the following implies constant v/f
omega2v    LT   omega          ; set angular speed -> T: D10
           MPY  omega_v        ; D10*D-9=D(1+1)
           PAC                ; product -> ACC: D2
           SACH  set_v,1       ; -> mag of ref voltage and -> D1
           lacc  set_v         ;
           sub   #max_v_       ; compare Uout w/ its upper limit
           BLEZ  uinuplim      ; continue if within limit
           splk  #min_v_,set_v ; saturate if not
           B     uinlolim      ;

uinuplim

           LACC  set_v         ;
           SUB   #min_v_       ; compare Uout with its lower limit
           BGEZ  uinlolim      ; continue if within limit
           splk  #min_v_,set_v ; saturate if not

uinlolim

update_led .if  DEVICE=F243
           ldp  #led_subdvdr   ;
           lacc led_subdvdr    ;

```



```

sub    one           ; update sub_divide counter
sac1  led_subdvdr   ; time to update LED display?
BNZ   noledupdate   ; no
splk  #led_freq_,led_subdvdr ; yes, reset
                                ; subdivide counter

bit   led_dir,BIT0   ; left shift?
bcnd  right_shift,NTC ; no
lacc  led_data,1     ; yes
sac1  led_data       ; left shift one bit
bit   led_data,BIT7  ; time to change direction?
bcnd  updateled,NTC  ; no
splk  #0,led_dir     ; yes
b     updateled      ;
right_shift  lacc  led_data,15 ;
sach  led_data       ; right shift one bit
bit   led_data,BIT0  ; time to change direction?
bcnd  updateled,NTC  ; no
splk  #1,led_dir     ; yes
updateled  out   led_data,LED_ADDR ; update LED display
noledupdate

        .endif

reset_wd  LDP   #WDKEY>>7           ; Reset WD timer
          SPLK  #wd_rst_1,WDKEY      ;
          SPLK  #wd_rst_2,WDKEY      ;
          SPLK  #0000000001101111b,WDCR

          Ldp   #PCDATDIR>>7         ; set DP
          lacc  PCDATDIR              ;
          or    #00004h               ; IOPC[2] to 1
          sac1  PCDATDIR              ;
          B     main_loop              ; End of background loop

;=====
; Phantom interrupt
;-----
        .bss  phantom_int,1          ;

phantom_isr
        ldp   #phantom_int
        splk  #0badh,phantom_int ;
        ret

;=====
; PDPINT interrupt service
;-----
        .bss  _OVERCURRENT_TRIP_FLAG,1 ;

_c_int1
        SST   #ST0,ST0_save          ; save status register ST0
        SST   #ST1,ST1_save          ; save status register ST1
        LDP   #ACCH                   ; set DP
        SACH  ACCH                    ;
        SACL  ACCL                    ; save ACC

        Ldp   #PIVR>>7               ; set DP
        LACC  PIVR                    ; load peripheral INT
                                ; vector/ID/offset
        SUB   #020h                   ; PDPINT?

```




```

        bcnd  not_pdp,NEQ      ; not pdp
        call  pdp_isr        ; pdp_isr
        b     rest_int1      ;

not_pdp      call  phantomISR  ; got a phantom int if not

rest_int1    ldp   #ACCH      ; set DP
              ZALH  ACCH      ; restore ACC high
              ADDS  ACCL      ; restore ACC low
              LDP   #0        ; point to B2
              LST   #ST1,ST1_save ; restore status register ST1
              LST   #ST0,ST0_save ; restore status register ST0
              EINT                ;
              RET                ; return

pdp_isr      ldp   #_OVERCURRENT_TRIP_FLAG ; set DP
              splk #1,_OVERCURRENT_TRIP_FLAG ; set flag
              ret

;=====
; Interrupt driven inner loop for PWM
;-----
_c_int2      SST   #ST0,ST0_save ; save status register ST0
              SST   #ST1,ST1_save ; save status register ST1
              LDP   #ACCH      ; set DP
              MAR   *,AR0      ; set ARP
              SACH  ACCH      ;
              SACL  ACCL      ; save ACC
              Sph   P_hi      ;
              spl   P_lo      ; save P register
              mpy   #1        ; P<=T
              spl   T_save     ; save T register
              sar   AR0,AR0_save ; save AR0

              ldp   #PIVR>>7   ; set DP
              LACC  PIVR      ; read id of int
              SUB   #029h     ; GPT1 UF INT?
              cc    tluf_isr,EQ ; TLUF isr if yes
              b     rest_cntxt

              call  phantomISR  ; got a phantom int if not

rest_cntxt   LDP   #ACCH      ; set DP
              lar   AR0, AR0_save ; restore AR0
              lt    P_lo      ; T<=P_lo
              mpy   #1        ; P (low byte) <=1*P_lo
              lph   P_hi      ; P high byte <=P_hi
              lt    T_save     ; restore T
              ZALH  ACCH      ;
              ADDS  ACCL      ; restore ACC
              LDP   #0        ; point to B2
              LST   #ST1,ST1_save ; restore status register ST1
              LST   #ST0,ST0_save ; restore status register ST0
              EINT                ;
              RET                ; return

;=====
; SV PWM routine

```



```

; The routine refers to the following variables:
; omega - set angular frequency with scale of D10 in unit of rad
; t_sample - sampling period with scale of D-9 in unit of second
; t1_period_ - T1 period, the maximum compare value
;-----
tluf_isr      ldp    #EVIFRA>>7      ;
              splk   #0200h,EVIFRA    ; clear GPT1 UF INT flag

              ldp    #PCDATDIR>>7    ; set DP
              lacc   PCDATDIR         ;
              and    #0FFF7h         ; IOPC[3] to 0
              sacl   PCDATDIR         ;

;-----
; Generate revolving voltage vector Uout=trans(Ud Uq)
;-----
              ldp    #omega          ; Integrate speed to get phase
LT            omega                  ; set W -> T: D10
MPY          t_sample                ; D10*D-9=D(1+1)
PAC          ; product -> ACC: D2
SFR          ; -> D3
ADDH        theta_h                  ; D3+D3=D3 (32 bit)
ADDS        theta_l                  ;
SACH        theta_h                  ; save
SACL        theta_l                  ;
chk_lolim   bcnd   chk_uplim,GEQ     ; check upper limit if positive
              ADDH  theta_360         ; D3+D3=D3, rollover if not
              SACH  theta_h           ; save
              B     rnd_theta         ;

chk_uplim   SUBH  theta_360          ; D3-D3=D3 compare with 2*pi
              bcnd  rest_theta,LEQ   ; resume theta_h if within limit
              SACH  theta_h           ; rollover if not
              B     rnd_theta         ;

rest_theta  ADDH  theta_360          ; resume theta high
rnd_theta   ADD   #1,15              ; round up to upper 16 bits
              SACH  theta_r           ;

;-----
; Quadrant mapping
;-----
              LACC  one                ; assume theta (theta_h) is in
              ;                          quadrant 1
              SACL  SS                 ; 1=>SS, sign of SIN(theta)
              SACL  SC                 ; 1=>SC, sign of COS(theta)
              LACC  theta_r            ;
              SACL  theta_m           ; theta=>theta_m
              SUB   theta_90          ;
              BLEZ  E_Q                ; jump to end if 90>=theta

              ; assume theta (theta_h) is in quadrant 2
              splk  #-1,SC             ; -1=>SC
              LACC  theta_180         ;
              SUB   theta_r           ; 180-theta
              SACL  theta_m           ; =>theta_m
              BGEZ  E_Q                ; jump to end if 180>=theta

              ; assume theta (theta_h) is in quadrant 3

```



```

splk #-1,SS      ; -1=>SS
LACC theta_r     ;
SUB theta_180    ; theta-180
SACL theta_m     ; =>theta_m
LACC theta_270   ;
SUB theta_r      ;
BGEZ E_Q        ; jump to end if 270>=theta

                                ; theta (theta_h) is in quadrant 4
splk #1,SC       ; 1=>SC
LACC theta_360   ;
SUB theta_r      ;
SACL theta_m     ; 360-theta_h=>theta_m

E_Q

;-----
; sin(theta), cos(theta)
;-----
lt theta_m      ; D3. Find index
mpy theta_i     ; D3*D6=D(9+1)
pac             ; D10
sach sin_indx   ; D10
lacc sin_indx,11 ; r/s 5 by l/s 11 -> integer (D15)
sach sin_indx   ; right shift 5 bits => D15

lacc sin_entry  ; Look up sin
add sin_indx    ;
tblr sin_theta  ;
lacc sin_end    ;
sub sin_indx    ;
tblr cos_theta  ;

LT SS          ; Look up cos
MPY sin_theta  ; modify sign: D15*D1=D(16+1)
PAC           ;
SACL sin_theta ; left shift 16 bits and save: D1
LT SC         ;
MPY cos_theta  ; modify sin: D15*D1=D(16+1)
PAC           ;
SACL cos_theta ; left shift 16 bits and save: D1

;-----
; The following 4 lines are for purpose of debugging
;-----
; lacc sin_theta,10 ;
; add #04000h,10    ; Add 1
; ldp #T2CMPR>>7   ;
; sach T2CMPR       ; save to T2CMPR for debug

;-----
; Calcualte Ud & Uq
;-----
LT set_v       ; set v -> T: D1
MPY cos_theta  ; set v*cos(theta): D1*D1=D(2+1)
PAC           ; product -> ACC: D3
SACH Ud,1     ; d component of ref Uout: D2
MPY sin_theta  ; set v*sin(theta): D1*D1=D(2+1)
PAC           ; product -> ACC: D3

```



```

                SACH    Uq,1            ; q component of ref Uout: D2

;-----
; Determine sector
;-----
                lt      theta_r        ; D3
                mpy    theta_s        ; D3*D0=D4
                pac
                sach   sector         ;
                lacc   sector,5       ; r/s 11 by 1/s 5 -> integer (D15)
                sach   sector         ; right shift 11 bits

;-----
; Calculate T1&T2 based on:      Tpwn Uout=V1*T1+V2*T2
;
; i.e.      [T1 T2]=Tpwn*inverse[V1 V2]*Uout
; i.e.      [0.5*T1 0.5*T2]=Tp*inverse[V1 V2]*Uout
; i.e.      [0.5*C1 0.5*C2]=inverse[V1 V2]*Uout=M(sector)*Uout
;
; where     C1=T1/Tp, C2=T2/Tp, are normalized wrt Tp
;           M(sector)=inverse of [V1 V2] = decomposition matrix
;           obtained through table lookup
;           Uout=Transpose of [Ud Uq]
;           Tp=Timer 1 period = 0.5*Tpwm
;           Tpwm=PWM period Tpwm
;-----
                LACC   #dec_ms
                ADD    sector,2       ;
                SACL   temp           ; get the pointer
                LAR    AR0,temp       ; point to parameter table

; Calculate 0.5*C1 based on 0.5*C1=Ud*M(1,1)+Uq*M(1,2)
                LT     Ud              ; D2
                MPY   *+              ; M(1,1) Ud: D2*D1=D(3+1)
                PAC
                LT     Uq              ; D4
                MPY   *+              ; M(1,2) Uq: D2*D1=D(3+1)
                APAC
                BGEZ  cmp1_big0       ; continue if bigger than zero
                ZAC
cmp1_big0     SACH   temp             ; 0.5*C1: D4
                LT     temp           ; D4
                MPY   t1_periods      ; D4*D10 = D(14+1)
                PAC
                .if   SVPAT=HWPAT
                ADD   one,16          ; Avoid C1=0
                .endif

                SACH  cmp_1           ; 0.5*C1*Tp: D15

; Calculate 0.5*C2 based on 0.5*C2=Ud*M(2,1)+Uq*M(2,2)
                LT     Ud              ; D2
                MPY   *+              ; M(2,1) Ud: D2*D1=D(3+1)
                PAC
                LT     Uq              ; D2
                MPY   *+              ; M(2,2) Uq: D2*D1=D(3+1)
                APAC
                .if   SVPAT=HWPAT
                ADD   one,16          ; Avoid C2=0
                .endif

```



```

                                BGEZ  cmp2_big0      ; continue if bigger than zero
                                ZAC   cmp2_big0      ; zero it if less than zero
cmp2_big0                       SACH  temp          ; 0.5*C2: D4
                                LT    temp          ; D4
                                MPY   t1_periods    ; D4*D10 = D(14+1)
                                PAC   cmp2_big0     ; D15

                                .if   SVPAT=HWPAT
                                ADD   one,16        ; Avoid 0 C2 = 0
                                .endif

                                SACH  cmp_2         ; 0.5*C2*Tp: D15

; Calculate 0.5*C0 based on 0.5*C3*Tp=Tp*(1-0.5*C1-0.5*C2)
                                LACC  #t1_period_   ;
                                SUB   cmp_1         ;
                                SUB   cmp_2         ; D15
                                BGEZ  cmp0_big0     ; continue if bigger than zero
                                ZAC   cmp0_big0     ; zero it if less than zero
cmp0_big0                       SACL  cmp_0         ;
                                LACC  cmp_0,15      ; right shift 1b (by 1/s 15b)
                                SACH  cmp_0         ; 0.25*C0*Tp

                                .if   SVPAT=HWPAT
;-----
; Determine the ACTR pattern and reload ACTR and CMPR1&2
;-----
                                LACC  #cckwise_     ;
                                ADD   sector        ; point to entry in lookup table
                                TBLR  svpat        ; get the pattern
                                LAR   AR0,#ACTR     ; point to ACTR
                                LACC  *            ; Read ACTR
                                AND   #0FFFh      ; Clear sv pattern bits
                                OR    svpat        ; Re-configure sv pattern bits
                                SACL  *            ; Re-load ACTR

                                LAR   AR0,#CMPR1    ; point to CMPR1
                                LACC  cmp_1         ;
                                SACL  *+          ; cmp_1=>CMPR1, point to CMPR2

                                ADD   cmp_2         ;
                                SACL  *            ; cmp_2=>CMPR2

                                SUB   #t1_period_  ; limit CMPR2
                                BLEZ  in_lmt       ;
                                SPLK  #t1_period_,*

in_lmt                           .endif

                                .if   SVPAT=SWPAT
;-----
; Determine channel toggling sequence and load compare registers
;-----
                                LACC  #first_      ;
                                ADD   sector        ; point to entry in look up table
                                TBLR  first_tog    ; get 1st-to-toggle channel
                                LAR   AR0,first_tog ; point to the channel
                                LACC  cmp_0         ;

```



```

SACL * ; cmp_0 => the channel

LACC #second_ ;
ADD sector ; point to entry in look up table
TBLR sec_tog ; get 2nd-to-toggle channel
LAR AR0,sec_tog ; point to the channel
LACC cmp_0 ;
ADD cmp_1 ; cmp_0+cmp_1
SACL * ; => the channel

LACC #CMPR3 ;
SUB first_tog ;
ADD #CMPR2 ;
SUB sec_tog ;
ADD #CMPR1 ;
SACL temp ; get 3rd-to-toggle channel
LAR AR0,temp ; point to the channel
LACC cmp_0 ;
ADD cmp_1 ;
ADD cmp_2 ; cmp_0+cmp_1+cmp_2
SACL * ; =>the channel
#endif

ldp #PCDATDIR>>7 ; set DP
lacc PCDATDIR ;
or #00008h ; IOPC[3] to 1
sac1 PCDATDIR ;
RET ; return

.data
;-----
; Frequently used angles
;-----
*****
** The order between these angles and the decomposition **
** matrices in the following must not be changed. **
*****
angles_ .WORD 01922h ; pi/2: D3
        .WORD 03244h ; pi: D3
        .WORD 04b66h ; 3*pi/2: D3
        .WORD 06488h ; 2*pi: D3

        .if SVPAT=SWPAT
;-----
; Decomposition matrices indexed by the sector Uout is in for s/w
; implemented SV PWM scheme
;-----
        .WORD 20066 ; D1
        .WORD -11585
        .WORD 0
        .WORD 23170

        .WORD -20066
        .WORD 11585
        .WORD 20066
        .WORD 11585

        .WORD 0

```



```
.WORD      23170
.WORD      -20066
.WORD      -11585

.WORD      0
.WORD      -23170
.WORD      -20066
.WORD      11585

.WORD      -20066
.WORD      -11585
.WORD      20066
.WORD      -11585

.WORD      20066
.WORD      11585
.WORD      0
.WORD      -23170
.endif

.if      SVPAT=HWPAT
;-----
; Decomposition matrices indexed by the sector Uout is in for h/w
; implemented SV PWM scheme
;-----
.WORD      20066      ; D1
.WORD      -11585
.WORD      0
.WORD      23170

.WORD      20066
.WORD      11585
.WORD      -20066
.WORD      11585

.WORD      0
.WORD      23170
.WORD      -20066
.WORD      -11585

.WORD      -20066
.WORD      11585
.WORD      0
.WORD      -23170

.WORD      -20066
.WORD      -11585
.WORD      20066
.WORD      -11585

.WORD      0
.WORD      -23170
.WORD      20066
.WORD      11585
.endif

.if      SVPAT=SWPAT
;-----
```



```

; Addresses of compare registers of the 1st-to-toggle channels
; indexed by the sector, ref v is in
;-----
first_          .WORD    CMPR1    ;
                .WORD    CMPR2    ;
                .WORD    CMPR2    ;
                .WORD    CMPR3    ;
                .WORD    CMPR3    ;
                .WORD    CMPR1    ;

;-----
; Addresses of compare registers of the 2nd-to-toggle channels
; indexed by the sector, ref v is in
;-----
second_         .WORD    CMPR2    ;
                .WORD    CMPR1    ;
                .WORD    CMPR3    ;
                .WORD    CMPR2    ;
                .WORD    CMPR1    ;
                .WORD    CMPR3    ;
                .endif

                .if      SVPAT=HWPAT
;-----
; Lookup table for ACTR[15-12] indexed by sector number
;-----
cckwise_        .WORD    0001000000000000b
                .WORD    0011000000000000b
                .WORD    0010000000000000b
                .WORD    0110000000000000b
                .WORD    0100000000000000b
                .WORD    0101000000000000b
                .endif

;-----
; sine table for theta from 0 to 90 per every 1 degree
;-----
sin_entry_      .WORD    0          ; sin table
                .WORD    286        ; D1
                .WORD    572
                .WORD    857
                .WORD    1143
                .WORD    1428
                .WORD    1713
                .WORD    1997
                .WORD    2280
                .WORD    2563
                .WORD    2845
                .WORD    3126
                .WORD    3406
                .WORD    3686
                .WORD    3964
                .WORD    4240
                .WORD    4516
                .WORD    4790
                .WORD    5063
                .WORD    5334

```




.WORD 5604
.WORD 5872
.WORD 6138
.WORD 6402
.WORD 6664
.WORD 6924
.WORD 7182
.WORD 7438
.WORD 7692
.WORD 7943
.WORD 8192
.WORD 8438
.WORD 8682
.WORD 8923
.WORD 9162
.WORD 9397
.WORD 9630
.WORD 9860
.WORD 10087
.WORD 10311
.WORD 10531
.WORD 10749
.WORD 10963
.WORD 11174
.WORD 11381
.WORD 11585
.WORD 11786
.WORD 11982
.WORD 12176
.WORD 12365
.WORD 12551
.WORD 12733
.WORD 12911
.WORD 13085
.WORD 13255
.WORD 13421
.WORD 13583
.WORD 13741
.WORD 13894
.WORD 14044
.WORD 14189
.WORD 14330
.WORD 14466
.WORD 14598
.WORD 14726
.WORD 14849
.WORD 14968
.WORD 15082
.WORD 15191
.WORD 15296
.WORD 15396
.WORD 15491
.WORD 15582
.WORD 15668
.WORD 15749
.WORD 15826
.WORD 15897
.WORD 15964



.WORD	16026
.WORD	16083
.WORD	16135
.WORD	16182
.WORD	16225
.WORD	16262
.WORD	16294
.WORD	16322
.WORD	16344
.WORD	16362
.WORD	16374
.WORD	16382
.WORD	16384



TI Contact Numbers

INTERNET

TI Semiconductor Home Page

www.ti.com/sc

TI Distributors

www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone +1(972) 644-5580

Fax +1(972) 480-7800

Email sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone

Deutsch +49-(0) 8161 80 3311

English +44-(0) 1604 66 3399

Español +34-(0) 90 23 54 0 28

Français +33-(0) 1-30 70 11 64

Italiano +33-(0) 1-30 70 11 67

Fax +44-(0) 1604 66 33 34

Email epic@ti.com

Japan

Phone

International +81-3-3457-0972

Domestic 0120-81-0026

Fax

International +81-3-3457-1259

Domestic 0120-81-0036

Email pic-japan@ti.com

Asia

Phone

International +886-2-23786800

Domestic

Australia 1-800-881-011

TI Number -800-800-1450

China 10810

TI Number -800-800-1450

Hong Kong 800-96-1111

TI Number -800-800-1450

India 000-117

TI Number -800-800-1450

Indonesia 001-801-10

TI Number -800-800-1450

Korea 080-551-2804

Malaysia 1-800-800-011

TI Number -800-800-1450

New Zealand 000-911

TI Number -800-800-1450

Philippines 105-11

TI Number -800-800-1450

Singapore 800-0111-111

TI Number -800-800-1450

Taiwan 080-006800

Thailand 0019-991-1111

TI Number -800-800-1450

Fax 886-2-2378-6808

Email tiasia@ti.com

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.



IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.

Copyright © 1999 Texas Instruments Incorporated