AC Induction Motor Control Using Constant V/Hz Principle and Space Vector PWM Technique with TMS320C240

APPLICATION REPORT: SPRA284A

Zhenyu Yu and David Figoli
DSP Digital Control System Applications

Digital Signal Processing Solutions
April 1998

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software 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 application 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, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

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

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.

TRADEMARKS

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Tables

AC Induction Motor Control Using Constant V/Hz Principle and Space Vector PWM Technique with TMS320C240

Abstract

The principles of constant V/Hz control for AC Induction motor and space vector PWM technique are reviewed. Two different implementations are presented. Implementation issues such as command voltage generation, switching pattern determination, speed measurement and scaling are discussed. Experimental data are shown. Full programs are attached in the appendices.

Product Support

World Wide Web

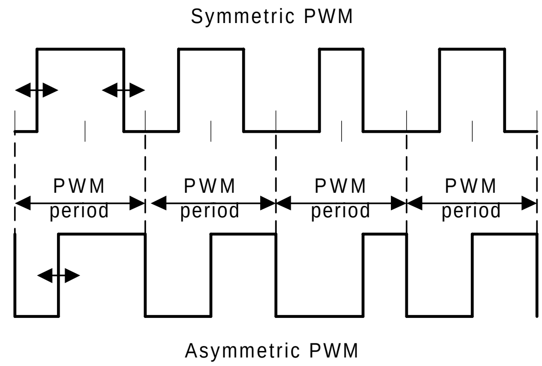
Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Introduction

Because of advances in solid state power devices and micro-processors, variable speed AC Induction motors powered by switching power converters are becoming more and more popular. Switching power converters offer an easy way to regulate both the frequency and magnitude of the voltage and current applied to a motor. As a result much higher efficiency and performance can be achieved by these motor drives with less generated noises. The most common principle of this kind, is the constant V/Hz principle which requires that the magnitude and frequency of the voltage applied to the stator of a motor maintain a constant ratio. By doing this, the magnitude of the magnetic field in the stator is kept at an approximately constant level throughout the operating range. Thus, (maximum) constant torque producing capability is maintained. When transient response is critical, switching power converters also allow easy control of transient voltage and current applied to the motor to achieve faster dynamic response. The constant V/Hz principle is considered for this application.

The energy that a switching power converter delivers to a motor is controlled by Pulse Width Modulated (PWM) signals applied to the gates of the power transistors. PWM signals are pulse trains with fixed frequency and magnitude and variable pulse width. There is one pulse of fixed magnitude in every PWM period. However, the width of the pulses changes from period to period according to a modulating signal. When a PWM signal is applied to the gate of a power transistor, it causes the turn on and turn off intervals of the transistor to change from one PWM period to another PWM period according to the same modulating signal. The frequency of a PWM signal must be much higher than that of the modulating signal, the fundamental frequency, such that the energy delivered to the motor and its load depends mostly on the modulating signal. Figure 1 shows two types of PWM signals, symmetric and asymmetric edge-aligned. The pulses of a symmetric PWM signal are always symmetric with respect to the center of each PWM period. The pulses of an asymmetric edge-aligned PWM signal always have the same side aligned with one end of each PWM period. Both types of PWM signals are used in this application.

Figure 1. Symmetric and asymmetric PWM signals



It has been shown that symmetric PWM signals generate less harmonics in the output current and voltage.

Different PWM techniques, or ways of determining the modulating signal and the switch-on/switch-off instants from the modulating signal, exist. Popular examples are sinusoidal PWM, hysterises PWM and the relatively new space vector PWM. These techniques are commonly used with three phase Voltage Source power inverters for the control of three-phase AC induction motors. The space vector PWM technique is employed in this application.

Background

In this section, the principle of constant V/Hz for AC induction motor and the theory of space vector pulse-width modulation are reviewed for better understanding of this application.

The Principle of Constant V/Hz for AC Induction Motor

Assume the voltage applied to a three phase AC Induction motor is sinusoidal and neglect the voltage drop across the stator resistor. Then we have, at steady state,

$$\hat{V} \approx j\omega\,\hat{\Lambda} \qquad (1)$$

i.e.

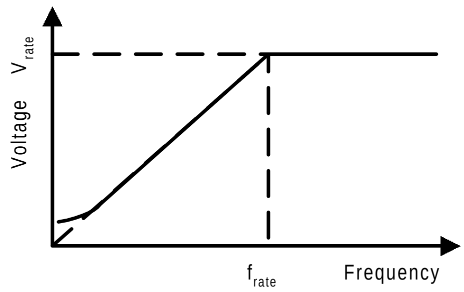
$$V \approx \omega\,\Lambda \qquad (2)$$

where $\hat{V}$ and $\hat{\Lambda}$ are the phasors of stator voltage and stator flux, and V and Λ are their magnitude, respectively. Thus, we get

$$\Lambda \approx \frac{V}{\omega} = \frac{1}{2\pi}\frac{V}{f} \qquad (3)$$

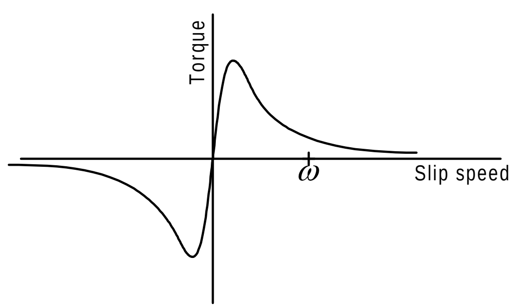
from which it follows that if the ratio V/f remains constant with the change of f, then Λ remains constant too and the torque is independent of the supply frequency. In actual implementation, the ratio between the magnitude and frequency of the stator voltage is usually based on the rated values of these variables, or motor ratings. However, when the frequency and hence also the voltage are low, the voltage drop across the stator resistance cannot be neglected and must be compensated. At frequencies higher than the rated value, the constant V/Hz principle also have to be violated because, to avoid insulation break down, the stator voltage must not exceed its rated value. This principle is illustrated in Figure 2.

Figure 2. Voltage versus frequency under the constant V/Hz principle



Since the stator flux is maintained constant, independent of the change in supply frequency, the torque developed depends on the slip speed only, which is shown in Figure 3. So by regulating the slip speed, the torque and speed of an AC Induction motor can be controlled with the constant V/Hz principle.

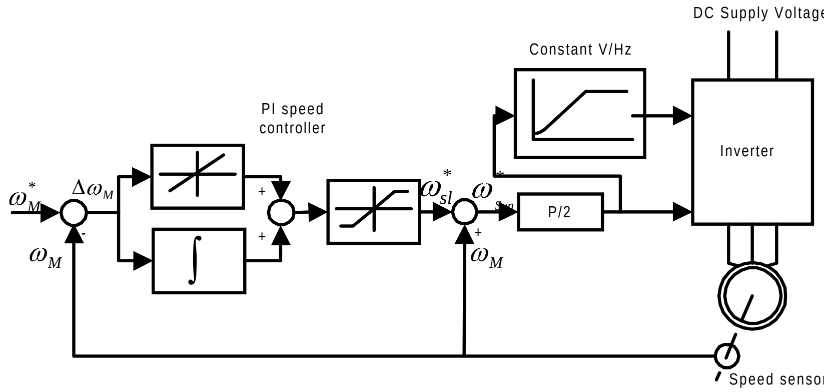
Figure 3. Torque versus slip speed of an induction motor with constant stator flux



Both open and closed-loop control of the speed of an AC induction motor can be implemented based on the constant V/Hz principle. Open-loop speed control is used when accuracy in speed response is not a concern such as in HVAC (heating, ventilation and air conditioning), fan or blower applications. In this case, the supply frequency is determined based on the desired speed and the assumption that the motor will roughly follow its synchronous speed. The error in speed resulted from slip of the motor is considered acceptable.

When accuracy in speed response is a concern, closed-loop speed control can be implemented with the constant V/Hz principle through regulation of slip speed, as illustrated in Figure 4, where a PI controller is employed to regulate the slip speed of the motor to keep the motor speed at its set value.

Figure 4. Closed-loop PI speed control based on constant V/Hz

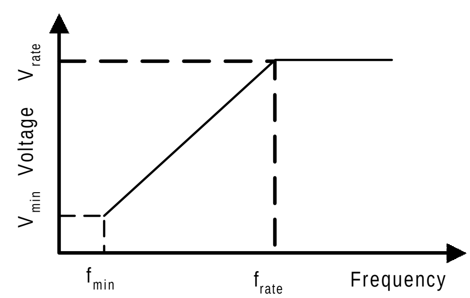


Two variations of the profile in Figure 2 are implemented here:

Profile I:

The profile in Figure 2 is used except that a lower limit is imposed on frequency. This approach is acceptable to applications such as fan and blower drives where the speed response at low end is not critical. Since the rated voltage which is also the maximum voltage is applied to the motor at rated frequency, only the rated, minimum and maximum frequency information is needed to implement the profile.

Figure 5. V/Hz profile I



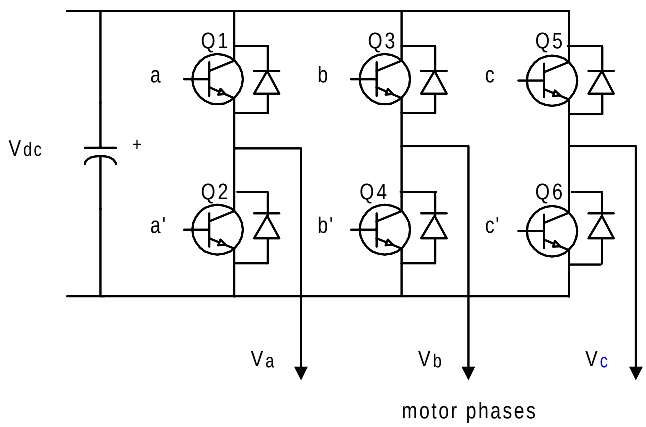
Profile II:

All four parameters, V_{rate}, $V_{\min}$, f_{rate} and $f_{\min}$, in Figure 5 are used. However the command frequency is allowed to go below $f_{\min}$ with the command voltage saturating at $V_{\min}$. This way, the V/Hz profile can be modified to off set the voltage drop across the stator resistance and the inverter.

Space Vector PWM Technique

The structure of a typical three-phase voltage source power inverter is shown in Figure 6. Va, Vb and Vc are the output voltages applied to the windings of a motor. Q1 through Q6 are the six power transistors that shape the output, which are controlled by a, a', b, b', c and c'. For AC Induction motor control, 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 Q1, Q3 and Q5, or equivalently, the state of a, b and c, are sufficient to evaluate the output voltage.

Figure 6. Three phase power inverter diagram



The relationship between the switching variable vector $[a, b, c]^t$ and the line-to-line voltage vector $[V_{ab}\ V_{bc}\ V_{ca}]^t$ is given by (4) in the following:

$$\begin{bmatrix} V_{ab} \\ V_{bc} \\ V_{ca} \end{bmatrix} = V_{dc} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad (4)$$

from which one can arrive at equation (5) as follows which determines the phase voltage vector $[V_a\ V_b\ V_c]^t$, where V_{dc} is the DC supply voltage, or the bus voltage.

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \frac{1}{3} V_{dc} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad (5)$$

Switching Patterns and the Basic Space Vectors

As shown in Figure 6, there are eight possible combinations of on and off patterns for the three upper power transistors that feed the three phase power inverter. Notice that the on and off states of the lower power transistors are opposite to the upper ones and so are completely determined once the states of the upper power transistors are known. The eight combinations and the derived output line-to-line and phase voltages in terms of DC supply voltage V_{dc}, according to equations (4) and (5), are shown in Table 1.

Space Vector PWM refers to a special switching sequence of the upper three power transistors of a three phase power inverter. It has been shown to generate less harmonic distortion in the output voltages and or currents applied to the phases of an AC motor and provides more efficient use of supply voltage in comparison with direct sinusoidal modulation technique.

Table 1. Switching patterns and output voltages of a 3-phase power inverter

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

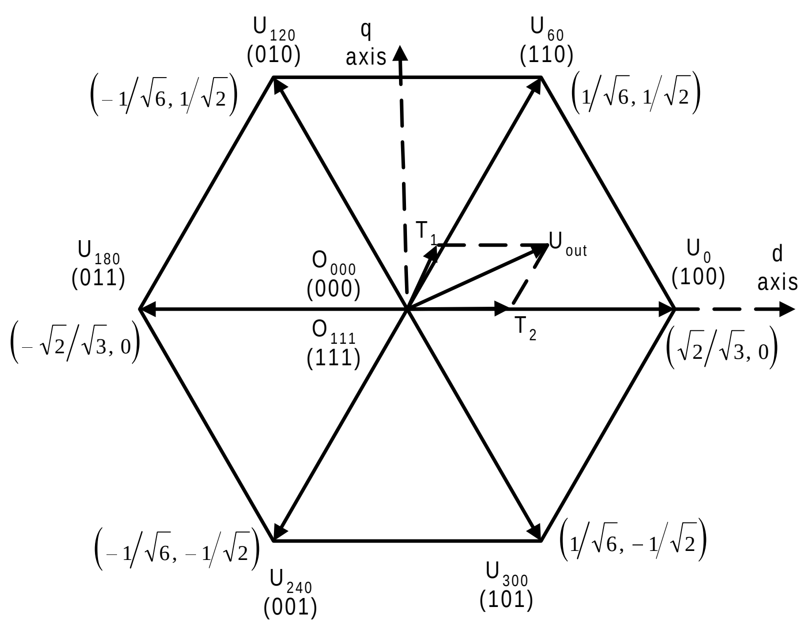
Assuming d and q are the horizontal and vertical axes of the stator coordinate frame, then the d-q transformation given in equation (6) can transform a three phase voltage vector into a vector in the d-q coordinate frame which represents the spatial vector sum of the three phase voltage. The phase voltages corresponding to the eight combinations of switching patterns can be mapped into the d-q plane in Figure 7 by the same d-q transformation.

$$T_{abc\text{-}dq} = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \qquad (6)$$

This transformation is equivalent to an orthogonal projection of $[a, b, c]^t$ onto the two dimensional plane perpendicular to the vector $[1, 1, 1]^t$ (the equivalent d-q plane) in a three-dimensional coordinate system, the results of which are six non-zero vectors and two zero vectors. The nonzero vectors form the axes of a hexagonal as shown in Figure 7. 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 motor. The eight vectors are called the basic space vectors and are denoted by U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000} and O_{111}. The same transformation can be applied to the desired output voltage to get the desired reference voltage vector U_{out} in the d-q plane.

The objective of space vector PWM technique is to approximate the reference voltage vector U_{out} by a combination of the eight switching patterns. One simple means of approximation is to require the average output of the inverter (in a small period, T) to be the same as the average of U_{out} in the same period. This is shown in equation (7), where T_1 and T_2 are the respective durations in time for which switching patterns U_x and $U_{x\pm60}$ are applied within period T, and U_x and $U_{x\pm60}$ form the sector containing U_{out}. Assuming the PWM period, $T_{PWM,}$ is small, and the change of U_{out} is relatively slow, from equation (7) we get equation (8).

Figure 7. The basic space vectors and switching patterns



$$\frac{1}{T}\int_{nT}^{(n+1)T} U_{out} = \frac{1}{T}\left(T_1 U_x + T_2 U_{x\pm 60}\right) \qquad n = 0,1,2,.....,T_1 + T_2 \le T \quad (7)$$

$$\int_{nT_{PWM}}^{(n+1)T_{PWM}} U_{out} = T_{PWM} U_{out} = \left(T_1 U_x + T_2 U_{x\pm 60}\right) \qquad n = 0,1,2,.....,T_1 + T_2 \le T_{PWM} \quad (8)$$

Approximation of Output with Basic Space Vectors

Equation (8) means for every PWM period, the desired reference voltage U_{out} can be approximated by having the power inverter in switching pattern U_x and U_{x+60} (U_{x-60}) for T_1 and T_2 duration of time respectively. Since the sum of T_1 and T_2 is less than or equal to T_{pwm}, the power inverter needs to have a 0 (000 or 111) pattern inserted for the rest of the period. Therefore equation (8) becomes equation (9) in the following, where $T_1 + T_2 + T_o = T_{pwm}$.

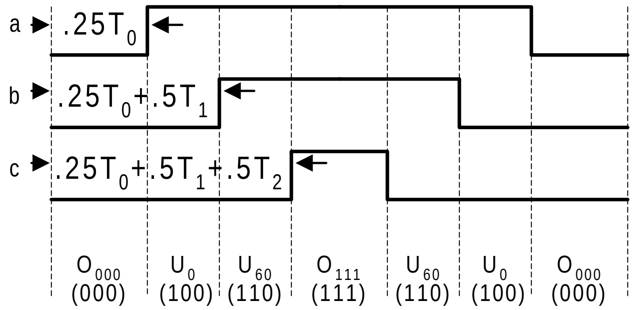
$$T_{pwm} U_{out} = T_1 U_x + T_2 U_{x\pm 60} + T_0 (0_{000}\, or\, 0_{111}) \qquad (9)$$

Note that the third term on the right-hand side of equation (8) above doesn't affect the vector sum on the left-hand side.

The reference voltage vector U_{out} is obtained by mapping the desired three phase output voltages to the d-q plane through the same d-q transform. When the desired output voltages are three phase sinusoidal voltages with 120 degree phase shift, U_{out} becomes a vector rotating around the origin of the d-q plane with a frequency corresponding to that of the desired three phase voltages. The envelope of the hexagonal formed by the basic space vectors, as shown in Figure 6 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 rotating vector. This gives a maximum magnitude of $V_{dc}/\sqrt{2}$ for U_{out}. Correspondingly, the maximum rms values of the fundamental line-to-line and line-to-neutral output voltages are $V_{dc}/\sqrt{2}$ and $V_{dc}/\sqrt{6}$, which is $2/\sqrt{3}$ times higher than what a sinusoidal PWM technique can generate. Therefore the bus voltage V_{dc} needed for a motor rated at V_{rate} is determined by $V_{dc} = \sqrt{2}\, V_{rate}$.

An example of symmetric space vector PWM wave forms are shown in Figure 8 where it is assumed that the reference voltage U_{out} is in the sector formed by vectors U_0 and U_{60}.

Figure 8. A symmetric space vector PWM switching pattern



Implementation

To control a three-phase AC Induction motor, one needs a three-phase inverter with the required DC link and driving circuits, and a digital processor that supplies the PWM signals based on a selected control algorithm. Here we assume that a three-phase switching power inverter with the necessary driving circuits and DC link is available and focus on algorithm and software implementation issues. A 3-phase AC induction motor control algorithm based on the discussed constant V/Hz principle and the space vector PWM technique generally contains the following steps:

1) Configure the timers and compare units to generate symmetric or asymmetric PWM outputs;

2) Input desired speed, use it as the command speed if open-loop speed control is implemented;

3) Measure speed feedback if closed-loop speed control is implemented;

4) Obtain command frequency with speed controller if closed-loop speed control is implemented;

5) Obtain the magnitude of reference voltage vector Uout (command voltage) based on V/Hz profile;

6) Obtain the phase of U_{out} based on command frequency;

7) Determine which sector Uout is in;

8) Decompose Uout to obtain T_1, T_2 and T_0;

9) Determine the switching pattern or sequence to be used and load the calculated compare values into the corresponding compare registers.

The above procedure assumes that the digital signal processor has all the needed timers and compare units with associated PWM outputs. This is true in the case of TMS320C240. The major features of the TMS320C240 include:

- TMS320C2xx CPU core with 50nS instruction cycle time;
- 544 words of on-chip data/program memory, 16K words of on-chip program ROM or Flash EEPROM, 64K words of program, 64Kwords of data and 64K words of I/O space of address reach;
- Dual 10-bit A/D converter with 6.6 μS of converter time per two input channels;

- ❑ PLL, Watchdog Timer, SCI, SPI, and 28 multiplexed I/O pins;
- ❑ Event Manager featuring
  - a) 12 compare/PWM outputs, 9 of which are independent;
  - b) Three general-purpose up and up/down timers, each with a 16-bit compare unit capable of generating one independent PWM output;
  - c) Three 16-bit full compare units capable of generating 6 complimentary PWM outputs with programmable dead band;
  - d) Three 16-bit simple compare units capable of generating 3 independent PWM outputs;
  - e) Four capture units each with one capture input and a two-level deep FIFO stack;
  - f) Direct QEP encoder interface shared with two capture inputs;

TMS320C240 has the necessary features to allow easy implementation of different motor control algorithms and PWM techniques. For the application here, the following set up is needed for the generation of PWM outputs:

- ❑ GP Timer 1 is configured in either continuous-up or continuous-up/down mode to generate correspondingly asymmetric or symmetric PWM.
- ❑ The three full compare units are configured in PWM mode to generate six complementary PWM outputs.

Once the above items are completed, all that is needed to generate the required PWM outputs is for the application code to update the compare values based on the discussed principle and PWM techniques.

Next, two separate implementations are discussed in detail, with the first corresponding to the code in Appendix I. Open-loop speed control for AC induction motor based on constant V/Hz principle and space vector PWM, and the second corresponding to the code in Appendix II. Closed-loop speed control for AC induction motor based on constant V/Hz principle and space vector PWM.

Implementation I - Open-loop speed control for 3-phase AC induction motor

There are two major issues that must be resolved to implement the discussed principle and PWM technique. One is how to generate or represent the revolving reference voltage vector U_{out} given the command frequency and magnitude of the reference voltage vector. The other is the determination of the switching pattern based on this reference voltage vector.

Overview

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, and sector-based table look-up for PWM channel toggling sequence. GP Timer 1 is used as the time base for PWM output generation with the Full Compare Units. GP Timer 2 is used to time the sampling period allowing independent control of sampling frequency from PWM frequency. However, the same frequency of 25KHz is used here. The flow chart of this implementation is illustrated in Figure 9 in the following.

An ADC channel is used to input the speed command. It is assumed that the accuracy of speed response is not a concern. Therefore, open-loop speed control is implemented. Applications such as blower/fan drives typically don't care too much about the accuracy in speed response.

The major steps involved in this implementation are:

- Integrate the command speed to get the phase, theta, of the reference vector;
- Determine the quarter theta is in, map theta to the first quarter and record the correct signs of SIN(theta) and COS(theta);
- Use theta based look-up table to obtain SIN(theta) and COS(theta) and the d and q components of the reference voltage vector;
- Determine the sector, s, theta is in;
- Use s based look-up table to find the decomposition matrix and decompose the reference voltage vector;
- Use s based look-up table to determine which PWM channel toggles first, second and third and load the compare registers with appropriate values.

The assumption here is that the timers and compare units and associated compare/PWM outputs have been properly configured to generate the right PWM outputs based on on-line determined compare values. The following explains the details of these steps.

Step by Step Explanation

32-bit integration to obtain the phase of the reference voltage vector to minimize error accumulation, as shown in the following code segment:

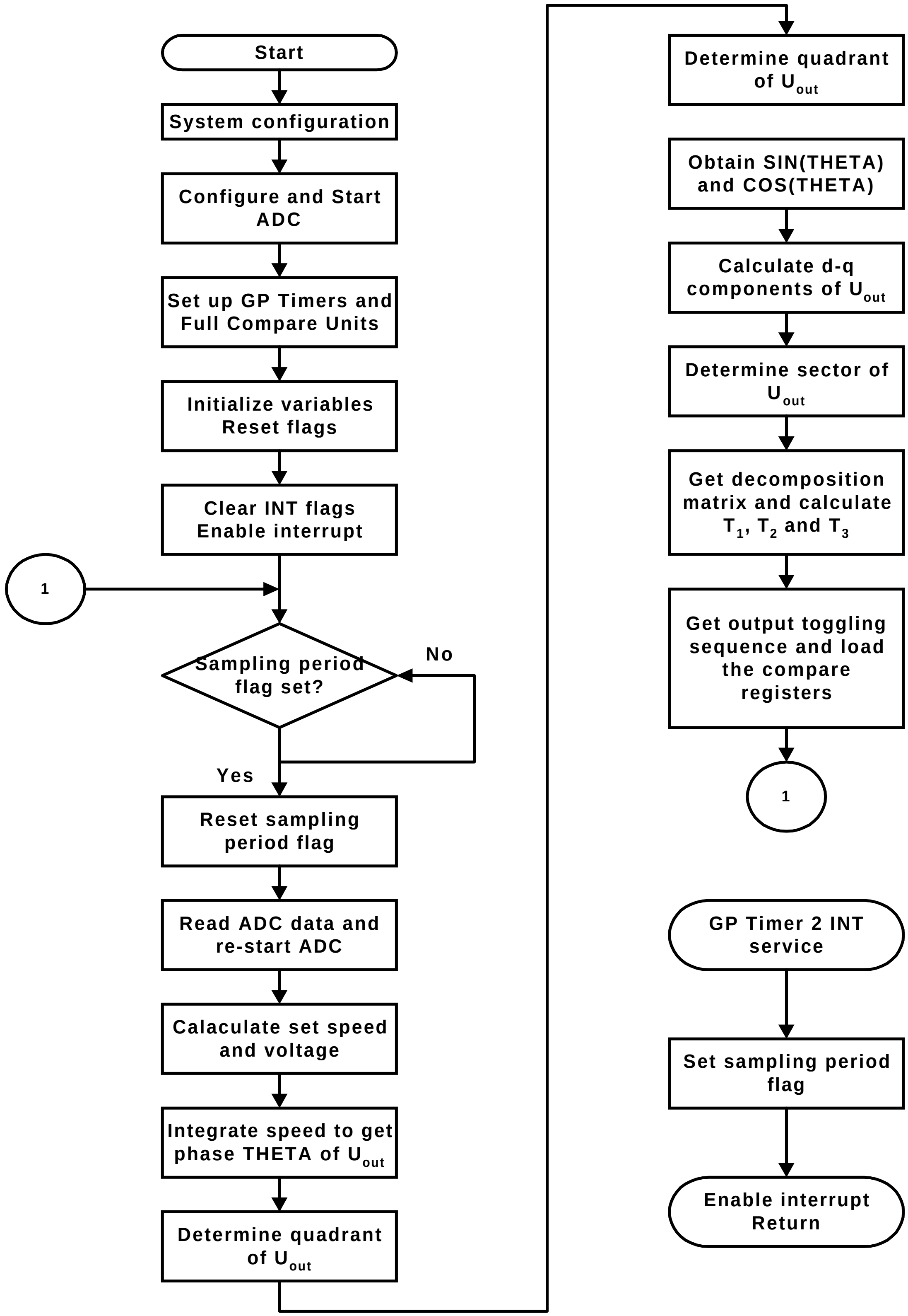
```
*********************************************************************
** Obtain theta (phase of Uout) through 32 bit integration  **
*********************************************************************
      LT    S_W             ;
      MPY   T_sample        ; D-9*D11=D(2+1)
      PAC                   ;

      ADDS  THETAL          ;
      ADDH  THETAH          ;
      SACH  THETAH          ;
      SACL  THETAL          ; accumulate: D3+D3=D3

SUBH  theta_360             ; compare with 2*pi: D3-D3=D3
      BLEZ  Theta_in_limit; continue if within limit
      SACH  THETAH          ; mod(2*pi, THETA) if not
Theta_in_limit

      ZALH  THETAH          ;
      ADDS  THETAL          ;
      ADD   one,15          ;
      SACH  theta_r         ; round up to upper 16 bits
```

Figure 9. Program flow chart



Notice that modulo 360 is applied to theta to keep theta within 360 range. However, the result is rounded to the higher 16 bits for later reference. Notice that the notation *D*x is used to indicate and track the scaling of variables. It relates to the more popular *Q* scaling notation according to the following:

$$D_x = Q(15 - x) \qquad (10)$$

Quarter mapping. Since SIN and COS of any angle can always be obtained by mapping the angle to the first quarter with correct sign modification, only SIN values of the first quarter are needed. Determination of which quarter theta is in is done by simply comparing theta to the quarter limits. The signs of SIN and COS and mapping of theta to the first quarter are obtained in the mean time. The following code implements this operation:

```
**********************************************************************
** Determine quadrant                        **
**********************************************************************
      LACC  one          ; assume THETA (THETAH) 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

      LACK  #-1          ; assume THETA (THETAH) is in quadrant 2
                         ; if not
      SACL  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

      LACK  #-1          ; assume THETA (THETAH) is in quadrant 3
                         ; if not
      SACL  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

      LACC  one          ; THETA (THETAH) is in quadrant 4 if not
      SACL  SC           ; 1=>SC
      LACC  theta_360    ;
      SUB   theta_r      ;
      SACL  theta_m      ; 360-THETAH=>theta_m
E_Q
```

Table look-up to determine SIN and COS of theta. Two tables are used here. The first table lists all the discrete theta values while the second lists the corresponding SIN values. This way uneven spacing of theta values is allowed if necessary. A pointer is maintained and updated based on comparing the newly determined theta with entries in the theta table. This pointer is then used to index the SIN table to get the SIN and COS of theta. The following code implements this operation:

```
*********************************************************************
** Obtain theta table entry                                       **
*********************************************************************
      LACC  theta_1stentry  ;
      ADD   SP              ;
      TBLR  GPR0            ; get table(SP)

      LACC  theta_m         ;
      SUB   GPR0            ; compare theta_m with table(SP)
      BZ    look_end        ; end look-up if equal
      BGZ   inc_SP          ; increase SP if bigger

dec_SP   LACC  SP           ; decrease SP other wise
      SUB   one             ;
      SACL  SP              ; SP-1=>SP

      ADD   theta_1stentry  ; point to SP-1
      TBLR  GPR0            ; get table(SP-1)

      LACC  theta_m         ;
      SUB   GPR0            ; compare theta_m with table(SP-1)
      BLZ   dec_SP          ; decrease SP further if smaller
      B     look_end        ; jump to end if not

inc_SP LACC SP              ;
      ADD   one             ;
      SACL  SP              ; SP+1=>SP

      ADD   theta_1stentry  ; point to SP+1
```

```
      TBLR  GPR0            ; get table(SP+1)

      LACC  theta_m         ;
      SUB   GPR0            ; compare theta_m with table(SP+1)
      BGZ   inc_SP          ; increase further if bigger
look_end                    ; end if not

*********************************************************************
** Get sin(theta)                                                  **
*********************************************************************
      LACC  SIN_1stentry    ;
      ADD   SP              ;
      TBLR  sin_theta       ; get sin(THETA)

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

*********************************************************************
** Get cos(theta)                                                  **
*********************************************************************
      LACC  SIN_lastentry   ;
      SUB   SP              ;
      TBLR  cos_theta       ; get cos(THETA)

      LT    SC              ;
      MPY   cos_theta       ; modify sin: D15*D1=D(16+1)
      PAC                   ;
      SACL  cos_theta       ; left shift 16 bits and save: D1
```

Once SIN and COS of theta are obtained, two multiplications give the d and q components of the reference voltage vector.

By simply comparing theta with the sector limits, the sector s of theta is obtained as shown in the following code:

```
**********************************************************************
** Determine sector                                                 **
**********************************************************************
      MAR   *,AR0             ; ARP points to AR0
      LAR   AR0,#1            ; assume S=1

      LACC  theta_r           ;
      SUB   theta_60          ; compare with 60 (in sector 1?)
      BLEZ  E_S               ; jump to end if yes
      MAR   *+                ; assume S=2 if not

      LACC  theta_r           ;
      SUB   theta_120         ; compare with 120 (in sector 2?)
      BLEZ  E_S               ; jump to end if yes
      MAR   *+                ; assume S=3 if not.

      LACC  theta_r           ;
      SUB   theta_180         ; compare with 180 (in sector 3?)
      BLEZ  E_S               ; jump to end if yes
      MAR   *+                ; assume S=4 if not

      LACC  theta_r           ;
      SUB   theta_240         ; compare with 240 (in sector 4?)
      BLEZ  E_S               ; jump to end if yes
      MAR   *+                ; assume S=5 if not

      LACC  theta_r           ;
      SUB   theta_300         ; compare with 300 (in sector 5?)
      BLEZ  E_S               ; jump to end if yes
      MAR   *+                ; S=6 if not

E_S   SAR   AR0,S             ;
```

Based on s, the decomposition matrix is fetched. Decomposition of the reference voltage vector onto the basic space vectors of the sector is done by 2-by-2 matrix multiplication. The following code accomplishes this operation:

```
*********************************************************************
** Calculate T1 & T2 based on                                     **
**           Tpwm Uout = V1*T1 + V2*T2                            **
** or                                                             **
**             [T1 T2] = Tpwm [V1 V2]' Uout                       **
**             [0.5*T1 0.5*T2] = Tp [V1 V2]' Uout                 **
**             = Mdec(S) Uout                                     **
** where                                                          **
**             Mdec(S) = Tp [V1 V2]'                              **
**             Uout = Trans([Ud Uq])                              **
** Mdec is obtained through table look-up.                        **
** Note that timer period is half of PWM period.                  **
*********************************************************************

        LACC  #(decpar_1stent-4)
        ADD   S,2            ;
        SACL  GPR0           ; get the pointer
        LAR   AR0,GPR0       ; point to parameter table

        LT    Ud             ; calculate 0.5*T1
        MPY   *+             ; M(1,1) Ud: D4*D10=D(14+1)
        PAC                  ;
        LT    Uq             ;
        MPY   *+             ; M(1,2) UqP D4*D10=D(14+1)
        APAC                 ; 0.5*T1: D15+D15=D15
        BGEZ  cmp1_big0      ; continue if bigger than zero
        ZAC                  ; zero it if less than zero
cmp1_big0 SACH cmp_1         ;

        LT    Ud             ; Calculate 0.5*T2
        MPY   *+             ; M(2,1) Ud: D4*D10=D(14+1)
        PAC                      ;
        LT    Uq             ;
        MPY   *+             ; M(2,2) Uq: D4*D10=D(14+1)
        APAC                  ; 0.5*T2: D15+D15=D15
```

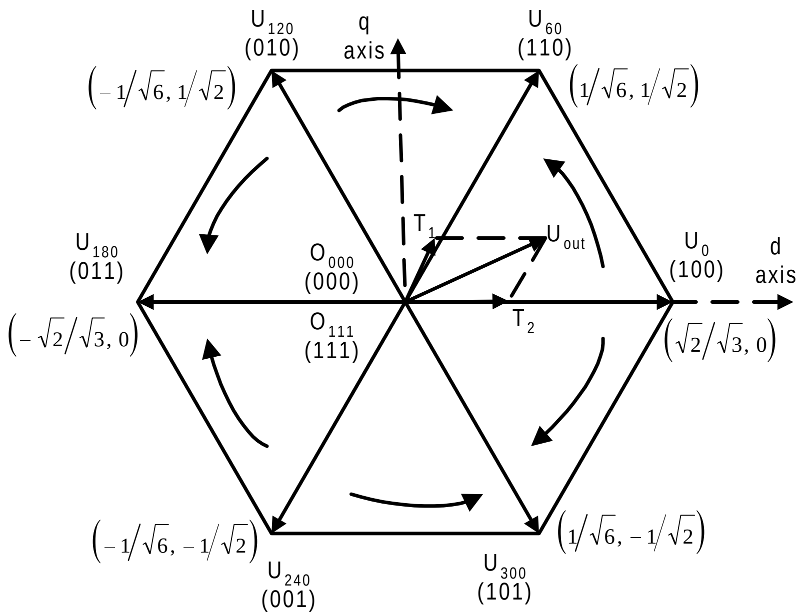
```
        BGEZ  cmp2_big0      ; continue if bigger than zero
        ZAC                  ; zero it if less than zero
cmp2_big0 SACH  cmp_2        ;

        LACC  #max_cmp_      ; Calculate 0.5*T0
        SUB   cmp_1          ;
        SUB   cmp_2          ; 0.5*T0 = Tp - 0.5*T1 - 0.5*T2: D15
        BGEZ  cmp0_big0      ; continue if bigger than zero
        ZAC                  ; zero it if less than zero
cmp0_big0 SACL  cmp_0        ;

        LACC  cmp_0,15       ; left shift 15 (right shift 1) bit
        SACH cmp_0           ; 0.25*T0
```

Before moving on to the next step, some discussion on the switching patterns implementing space vector PWM must be done. There are many switching patterns to implement space vector PWM. The pattern in Figure 8 is just one of them and is used in this implementation. This switching pattern is fixed for each sector and can be summarized as 000- U_i - $U_{i\pm 60}$-111- $U_{i\pm 60}$- U_i -0, meaning the PWM outputs switch sequentially from 000 to U_i, $U_{i\pm 60}$, 111, $U_{i\pm 60}$, U_i, and back to 000 in each period, where U_i and $U_{i\pm 60}$ are the basic space vectors forming the sector the reference voltage vector is in. Obviously, there are two possible switching directions for each sector, clock wise and counter clock wise. However, only one direction is such that only one channel toggles at a time, except when the reference voltage vector is on one of the basic space vectors. This approach has chosen the switching direction for each sector that results in one channel toggling at a time, as shown in Figure 10 below. Therefore, once the sector of U_{out} has been determined, the channels that toggle first, second and third are determined also. Based on this analysis, two look-up tables are constructed to use the sector s as an index to look for the respective compare register addresses for the channels that toggle the first and second in a PWM period. The compare register address for the channel that toggles the third can then be easily calculated. The compare registers are then loaded with obtained compare values. The correct PWM output pattern are then generated by the compare logic.

Figure 10. Switching sequence for each sector.



1) The above operation is carried out by the following code:

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

********************************************************************
** Addresses of compare registers corresponding to channels to    **
** toggle the 2nd in a given period indexed by the sector THETA   **
** (Uout) is in.                                                  **
********************************************************************
second_         .WORD CMPR2    ;
```

```
                  .WORD CMPR1    ;
                  .WORD CMPR3    ;
                  .WORD CMPR2    ;
                  .WORD CMPR1    ;
                  .WORD CMPR3    ;
..
.
*********************************************************************
** Determine the channel toggling sequence and load compare values**
*********************************************************************
                  LACC  #(first_-1) ;
                  ADD   S           ; point at entry in look up table
                  TBLR  CL          ; get the channel to toggle the
                                    ; 1st
                  LAR   AR0,CL      ; point at the 1st channel
                  LACC  cmp_0       ;
                  SACL  *           ; cmp_0=>the 1st channel

                  LACC  #(second_-1);
                  ADD   S           ; point at entry in look up table
                  TBLR  CM          ; get the channel to toggle the
                                    ; 2nd
                  LAR   AR0,CM      ; point at the 2nd channel
                  LACC  cmp_0       ;
                  ADD   cmp_1       ; cmp_0+cmp_1
                  SACL  *           ; => the 2nd channel

                  LACC  #CMPR3      ;
                  SUB   CL          ;
                  ADD   #CMPR2      ;
                  SUB   CM          ;
                  ADD   #CMPR1      ;
                  SACL  GPR0        ; get the channel to toggle the
                                    ; 3rd
                  LAR   AR0,GPR0    ; point at the 3rd channel
```

```
LACC  cmp_0       ;
ADD   cmp_1       ;
ADD   cmp_2       ; cmp_0+cmp_1+cmp_2
SACL  *           ; =>the 3rd channel
```

Notice that the final compare values are obtained at the same time the switching sequence is determined. Notice also that the scaling on the final compare values has to be *D*15 (or *Q*0) to result in integer values.

Scaling and Accuracy

As has been pointed out that a different scaling notation is used here which uses *Dx* to indicate and track the scaling of variables and *Dx* relates to *Q* notation by the equation $D_x = Q(15-x)$. Therefore a variable with scale *D*x can represent numbers from $-2^x + \Delta$ to $2^x - \Delta$, where Δ is an infinitely small number. In fact, a scaling approach different from the popular *Q*15 approach is adopted here which maximizes the dynamic accuracy of calculation. Consider the equation:

$$X = a*Y + b*Z \qquad (11)$$

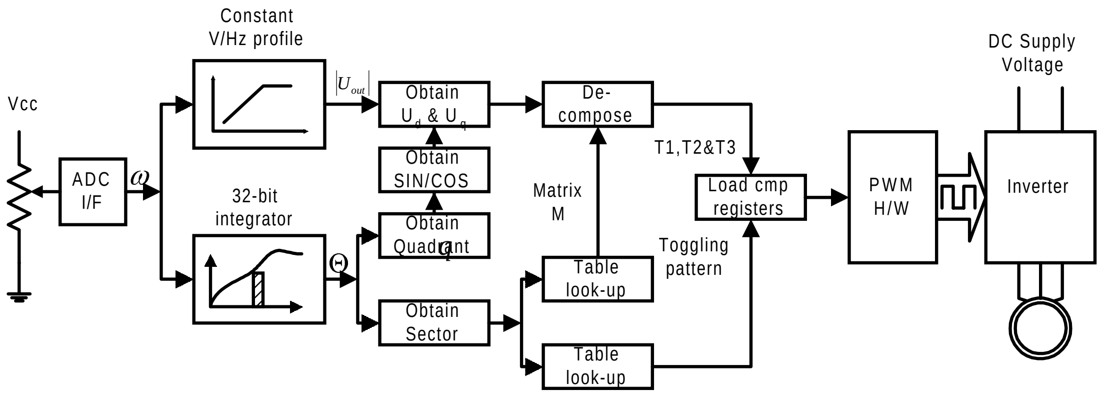
where *a* = 0.124, *b* = 0.4. Assume ranges of *Y* and *Z* are separately 0-7.999 and 0-0.999. In stead of assigning a scale of *Q*15 to all the variables, the scaling approach here assigns a separate scaling to *a*, *b*, *Y* and *Z* each with maximum possible resolution. In this case, the scale of *a* is assigned to be *D*-3 (or *Q*18) which can represent numbers from -0.12499 to 0.12499. Similarly, the scales of *b*, *Y* and *Z* are assigned to be *D*-1, *D*3 and *D*0, respectively. The scales of $a*Y$ and $b*Z$ are then *D*0 and *D*-1 based on the equation:

$$D_x * D_y = D(x+y) \qquad (12)$$

Therefore, the result of $b*Z$ has to be right shifted by one bit to change the scaling to D0 before the addition is made to obtain *X* which, in this case will have a scale of *D*0. The user must pay attention to the additions and subtractions just like in the popular *Q*15 format, because they may result in overflow. Therefore, right shift of both operands by one bit may be needed before an addition or subtraction. However, in most cases, the scaling of variables is determined based on the underlying physical quantities so that little or no concern of overflow is needed.

The block diagram of this implementation is shown in Figure 11in the following.

Figure 11. Block diagram of implementation I.



Implementation II - Closed loop speed control for 3-phase AC induction motor

This section covers the details of the second implementation - closed-loop speed control with V/Hz principle for AC induction motors. The PWM technique is the same, though a different implementation method is used. The closed loop speed control is based on the classical Proportional-Integral (PI) controller. The description given here is divided into 4 sections, namely:

1) Software Flow Overview
2) Space vector PWM
3) Measuring the motor shaft rotation speed
4) Closed loop speed control

Software Flow Overview

The entire application s/w is driven by an Interrupt service routine (ISR). As shown in Figure 12, the main code (i.e. background loop) consists simply of TMS320C240 peripheral initialization (e.g. PLL, Watchdog, Interrupt control & Event manager) and a time-out loop which allows the motor to startup in open-loop for a fixed duration until the closed-loop PI controller takes over. The remainder of the code is taken up entirely by PWM_ISR. This ISR is invoked every 41.6uS (24KHz) by the Period event flag on Timer 1 of the Event manager.

Figure 12. Software structure of Implementation II.

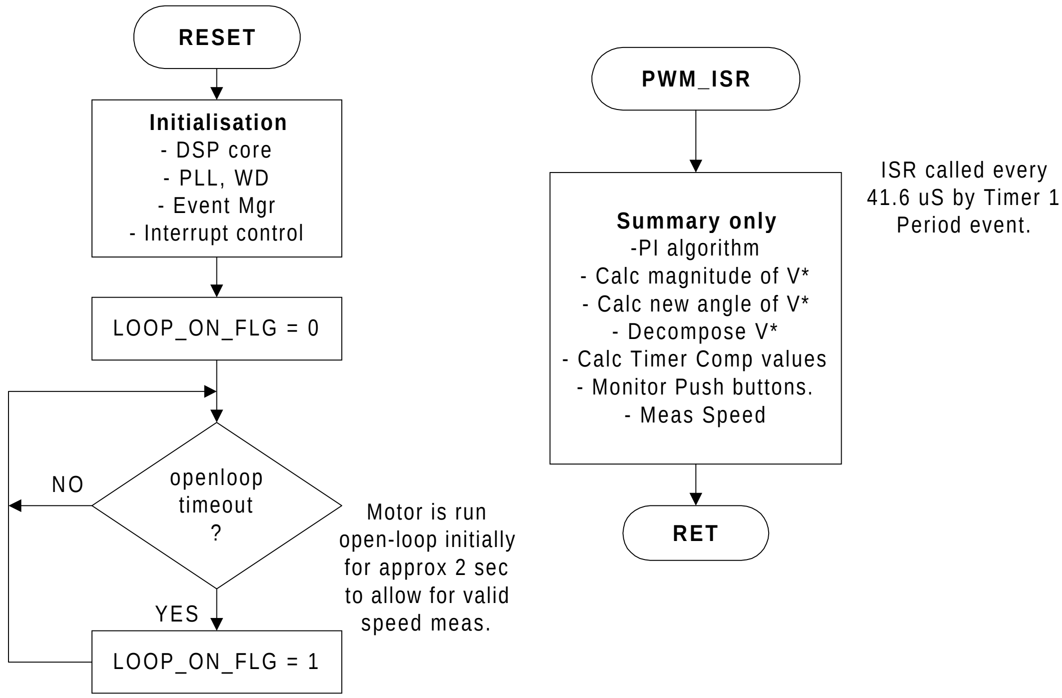


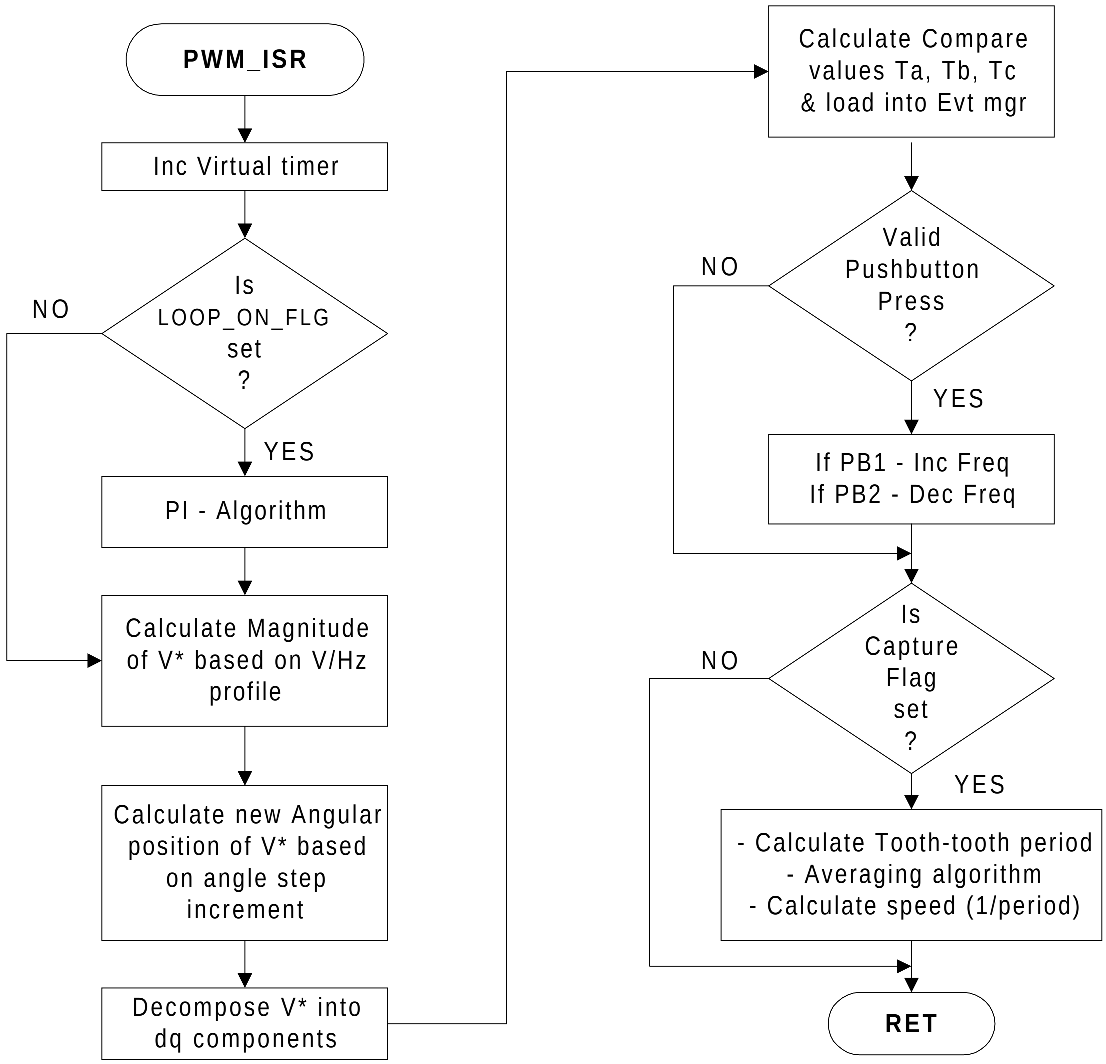
Figure 13 shows the entire flow of PWM_ISR. Most of the code is “in-line” code with the exception of 3 main conditional blocks, that is:

1) The PI algorithm is bypassed until LOOP_ON_FLG is set once the startup time-out occurs.

2) The up/down push-buttons which control the motor speed set point are interrogated every ISR cycle but no action is taken unless PB1 or PB2 is depressed.

3) The motor shaft period measurement and speed calculation is done only when the Capture 1 flag is set indicating a valid rising edge was detected from the hall effect sensor.

Figure 13. Flow chart of Implementation II.



Since there is only one ISR which is invoked every 41.6uS all algorithms shown in the flow diagram in Figure 13 make use of the common sampling rate of 24KHz. Although this rate may be appropriate for the Space vector PWM generation it may be “over-kill” for the PI speed control loop, motor shaft speed measurement and push-button debounce. However as will be discussed later, since the C240 DSP core at 20 MIPs has ample bandwidth to easily complete the entire ISR within the 41.6 uS it makes the s/w implementation much cleaner, since a multi-interrupt structure with priority and synchronization issues is avoided. In addition the algorithms benefit from the performance increase obtained when using high sampling rates.

Table 2 below gives an outline of the number of cycles taken to complete each major block of the ISR and how much total DSP CPU bandwidth is utilized by the entire application.

Table 2 CPU Cycles of Major Program Blocks

S/W block	**# cycles**	**# uS @ 50nS (20 MIPs)**	**% CPU loading**
Volts/Hz profile	24	1.20	2.88
V* positioning & decomposition	33	1.65	3.96
Compare value calculation & transform	53	2.65	6.37
Push button debounce & action	32	1.60	3.85
Period measurement & speed calculation	114	5.70	13.70
PI-loop algorithm	28	1.40	3.37
Misc overhead	43	2.15	5.17
TOTAL	**327**	**16.35 uS**	**39.3 %**

Space vector PWM

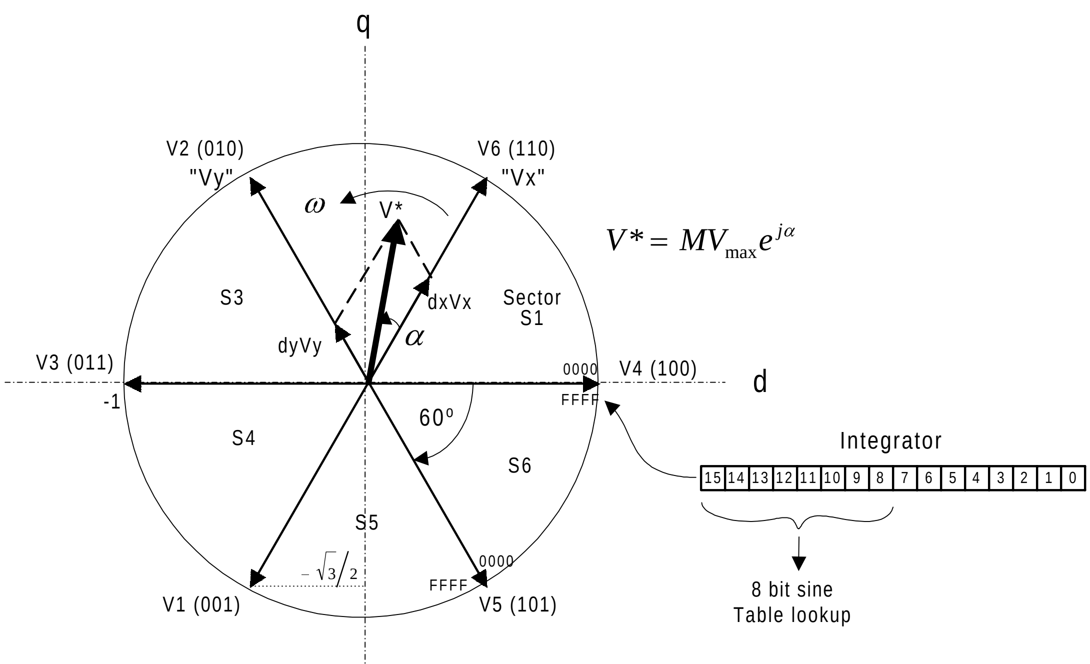
In order to create the required rotating MMF (magneto-motive force) in the stator of an AC induction machine the power inverter in Figure 6 needs to be driven with the correct switching variable vector $[a,b,c]^t$. To do this, 3 main elements need to be addressed:

- **Generating the Reference voltage vector**: This requires precise positioning of the Reference voltage vector (V*) within the d-q plane shown in Figure 14. This implies accurately controlling the rotational speed, ω and magnitude of this vector, M.
- **Decomposing the Reference Voltage vector:** For any given position of V* within the d-q plane it must be decomposed (transformed) into the set of appropriate switching variables a, b and c.
- **Realization of the switching pattern using PWM outputs**: By using the decomposed form of V* appropriate compare values are calculated for use with the C240 Event manager PWM generation units, i.e. outputs PW1→PWM6.

Generating the Reference Voltage Vector

To accurately position the Reference vector V* in both space and time, precise control of both the vector magnitude M (also called the modulation index) and the angle α is required. The aim here is to rotate V* in the d-q plane at a given angular speed (frequency) ω. This angular speed represents the rate of displacement of the electrical degrees of the AC induction machine. The vector magnitude M controls the resultant peak line voltage which is supplied by the inverter.

Figure 14. Generating and representing the reference voltage vector



In this implementation the angular speed ω is controlled by a precision frequency generation algorithm which relies on the modulo nature (i.e. wrap-around) of a finite length register, called Integrator in Figure 14. The upper 8 bits of this integrator (a data memory location in the C240) is used as a pointer to a 256 word Sine lookup table. By adding a fixed value (step size) to this register, causes the 8 bit pointer to cycle at a constant rate through the Sine table. In effect we are integrating angular velocity to give angular position. At the end limit the pointer simply wraps around and continues at the next modulo value given by the step size. The rate of cycling through the table is very easily and accurately controlled by the value of step size.

As shown in Figure 14, sine of α is needed to decompose the reference voltage vector into the basic space vectors of the sector the voltage vector is in. Since this decomposition is identical among the six sectors, only a 60 degree sine lookup table is needed. In order to complete one revolution (360^o) the sine table must be cycled through 6 times.

For a given step size the angular frequency (in cycles/sec) of V* is given by:

$$\omega = \frac{STEP \times fs}{6 \times 2^m} \tag{13}$$

where

f_s = sampling frequency (i.e. PWM frequency)

STEP = angle stepping increment

m = # bits in the integration register.

In the attached software implementation f_s = 24KHz , m=16 bits & STEP ranges from 0→2048. Table 3 gives an example of the resulting angular frequency for various values of STEP.

Table 3. Frequency mapping

STEP	Freq(Hz)	STEP	Freq(Hz)	STEP	Freq(Hz)
1	0.061	600	36.62	1700	103.76
20	1.22	700	42.72	1800	109.86
40	2.44	800	48.83	1900	115.97
60	3.66	900	54.93	2000	122.07
80	4.88	1000	61.04	2100	128.17
100	6.10	1100	67.14	2200	134.28

From the table it is clear that a STEP value of 1 gives a frequency of 0.061Hz, this defines the frequency setting resolution, i.e. the actual line voltage frequency delivered to the AC motor can be controlled to better than 0.1 Hz.

For a given f_s the frequency setting resolution is determined by m the number of bits in the integration register. Table 4 shows the theoretical resolution which results from various sizes of m.

Table 4. Resolution of frequency mapping

m (# bits)	Freq res(Hz)	m (# bits)	Freq res(Hz)
8	15.6250	17	0.0305
12	0.9766	18	0.0153

14	0.2441	19	0.0076
16	0.0610	20	0.0038

Another important parameter is the size of the lookup table. This directly effects the harmonic distortion produced in the resulting synthesized sine wave. As mentioned previously a 256 entry sine table is used which has a range of 60^o. This gives an angle lookup resolution of $60^o / 256 = 0.23^o$. Although not implemented here, interpolation techniques can improve the accuracy of the extracted sine values significantly. The table entries are given in Q15 format and a summarized version is shown below.

```
;---------------------------------------------------
;No. Samples: 256,  Angle Range: 60, Format: Q15
;---------------------------------------------------
;             SINVAL   ;  Index Angle  Sin(Angle)
;---------------------------------------------------
STABLE .word  0        ;     0     0     0.00
       .word  134      ;     1     0.23  0.00
       .word  268      ;     2     0.47  0.01
       .word  402      ;     3     0.70  0.01
       .word  536      ;     4     0.94  0.02
       .word  670      ;     5     1.17  0.02
       .word  804      ;     6     1.41  0.02
       .word  938      ;     7     1.64  0.03
       .word  1072     ;     8     1.88  0.03
       .word  1206     ;     9     2.11  0.04
         "       "           "      "      "
         "       "           "      "      "
         "       "           "      "      "
         "       "           "      "      "
       .word  27684    ;     246   57.66    0.84
       .word  27756    ;     247   57.89    0.85
       .word  27827    ;     248   58.13    0.85
       .word  27897    ;     249   58.36    0.85
       .word  27967    ;     250   58.59    0.85
       .word  28037    ;     251   58.83    0.86
       .word  28106    ;     252   59.06    0.86
       .word  28175    ;     253   59.53    0.86
```

```
.word  28243   ;    254   59.53   0.86
.word  28311   ;    255   59.77   0.86
```

Decomposing the reference voltage vector

Figure 14 shows the 6 base vectors (V1, V2, V3, V4, V5 & V6) which represent the line-to-neutral voltages for various switching combinations of [a, b, c]. At any given time the Inverter can produce only one of these vectors. Two other vectors not shown in the figure are V0 and V7, these are the zero vectors corresponding to states 0 (000) and 7 (111) of the switching variables. In order to produce an arbitrary Reference vector V*, a time average of given base vectors is required, i.e. the desired voltage vector V* located in a given sector, can be synthesized as a linear combination of the two adjacent base vectors, Vx and Vy, which are framing the sector, and either one of the two zero vectors, hence:

$$V^* = d_x V_x + d_y V_y + d_z V_z \qquad (14)$$

where Vz is the zero vector, and dx, dy and dz are the duty ratios of the states X, Y and Z within the PWM switching interval. The duty ratios must add to 100% of the PWM period, i.e: dx + dy + dz = 1.

Vector V* in Figure 14 can also be written as:

$$V^* = MV_{max}e^{j\alpha} = d_x V_x + d_y V_y + d_z V_z \qquad (15)$$

where M is the modulation index.

By decomposing V* into its dq components it can be shown that:

$$\frac{\sqrt{3}}{2} \times M\cos(\alpha) = dx + \frac{1}{2}dy \qquad (16)$$

$$\frac{\sqrt{3}}{2} \times M\sin(\alpha) = \frac{\sqrt{3}}{2}dy \qquad (17)$$

Solving for dx and dy gives:

$$dx = M\sin(60-\alpha) \qquad (18)$$

$$dy = M\sin(\alpha) \qquad (19)$$

These same equations apply to any sector, since the d-q reference frame, which has here no specific orientation in the physical space, can be aligned with any base vector. This is the reason why only a 60 degree sine lookup table is needed in this implementation. Shown below is the code section which implements the previously described V* positioning & decomposition:

```
;--------------------------------------------------------------------
;Calculate the angle ALPHA for the new position of the Space vector
;& perform the decomposition into the appropriate Base unit vectors.
;--------------------------------------------------------------------
NEW_ALPHA         LACC  ENTRY_NEW
                  SACL  ENTRY_OLD
                  LACC  ALPHA
                  ADD   STEP_ANGLE     ;Inc angle.
                  SACL  ALPHA          ;Save
                  LACC  ALPHA,8        ;Prepare pointer for Sine table
                  SACH  ENTRY_NEW
                  LACC  S_TABLE
                  ADD   ENTRY_NEW      ;ACC = actual table pointer
                                       ; value
                  TBLR  dy             ;dy=Sin(ALPHA)
                  LT    dy             ;dy is in Q15
                  MPY   V              ;V  is in Q15
                  PAC                  ;P = V * dy
                  SACH  dy,1           ;shift 1 to restore Q15 format
                  LACC  dy,11          ;scale for 10 bit integer
                                       ;resolution
                  SACH  dy             ;Save in Q0 format
                  LACC  #0FFh          ;ACC=60 deg
                  SUB   ENTRY_NEW
                  ADD   S_TABLE
                  TBLR  dx             ;dx=Sin(60-ALPHA)
                  LT    dx
                  MPY   V
                  PAC                  ;P = V * dx
                  SACH  dx,1           ;shift 1 to restore Q15 format
```

```
                LACC  dx,11          ;scale for 10 bit integer
                                     ;resolution
                SACH  dx             ;Save in Q0 format

;Determine which Sector Space the vector is in.
                LACC  ENTRY_NEW
                SUB   ENTRY_OLD
                BCND  BRNCH_SR, GEQ ;If negative need to change
                                     ;Sector
```

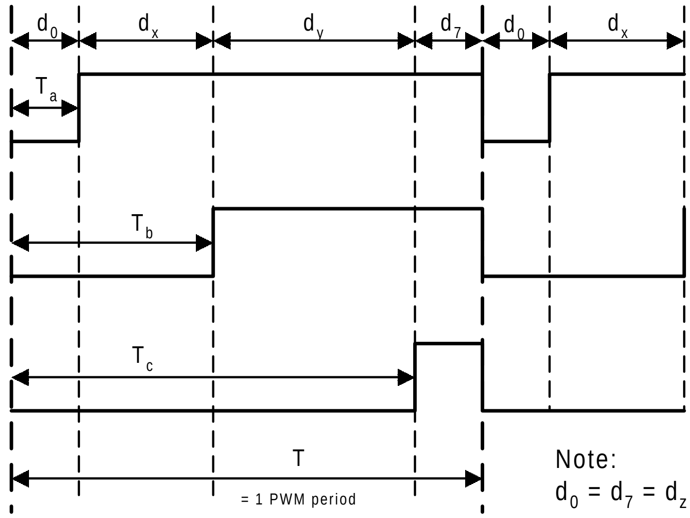
Some points to note in the code are:

- Q15 math is used to calculate dx & dy from sine table.
- Final format for dx and dy is in Q0 so that integer compare values are obtained.
- The modulation index M (named V in code) is derived from the V/Hz profile based upon the Frequency set point entered via Up/down push buttons by the user.
- Since no distinction is made between sectors for dx and dy calculations, a sector pointer needs to be maintained so that the appropriate compare values (Ta, Tb and Tc) are obtained.

Realization of the PWM Switching Pattern

Once the PWM duty ratios dx, dy and dz are calculated, the appropriate compare values for the compare registers of the C240 Event Manager can be evaluated. The switching pattern in Figure 15 is adopted here and is implemented with the Full Compare Units of C240. A set of 3 new compare values (Ta, Tb and Tc) need to be calculated every PWM period (T=41.6uS) to generate this switching pattern.

Figure 15. PWM output switching pattern of Implementation II.



From Figure 15, it can be seen:

Ta = (T - dx -dy)/2 (20)

Tb = dx + Ta (21)

Tc = T -Ta (22)

The code section below calculates the resultant PWM compare values from dx and dy. The appropriate phase "scrambling" to maintain correct switching sequences is embedded within the sector calculation section. This code section is repeated with a different mapping of Ta, Tb and Tc for each of the 6 sectors. Notice that only one sector calculation is performed every PWM period.

```
;----------------------------------------------------
;Sector 1 calculations - a,b,c --> a,b,c
;----------------------------------------------------
SECTOR_SR1:
      LACC  T     ;Acc = T
      SUB   dx    ;Acc = T-dx
      SUB   dy    ;Acc = T-dx-dy
      SFR         ;Acc = Ta = 1/2(T-dx-dy)  <A>
      SACL  Ta
      ADD   dx    ;Acc = Tb = dx+Ta   <B>
      SACL  Tb
      LACC  T     ;ACC = T
```

```
SUB   Ta    ;ACC = T-Ta
SACL  Tc    ;ACC = Tc = T-Ta   <C>
B     LOAD_COMPARES
```

The switching pattern shown in Figure 15 is an asymmetric PWM implementation. However, GP Timer 1 of C240 can also be put in continuous-up/down mode to generate symmetric PWM instead. Little change to the above code is needed to accommodate for this change. The choice between the symmetrical and asymmetrical case depends on the other care-about in the final implementation.

It is generally considered that symmetric PWM generates less harmonic distortion in motor voltage and currents. However, asymmetric PWM does have twice the resolution of symmetric PWM when the resolution of the timer is fixed. At 20 MIPs, C240 has a PWM (time) resolution of 50nS when asymmetrical PWM outputs are generated and a resolution of 100nS when symmetric PWM outputs are generated. Table 5 shows how the effective PWM resolution in number of bits compare between the asymmetrical and symmetrical PWM for a given PWM frequency of 24KHz.

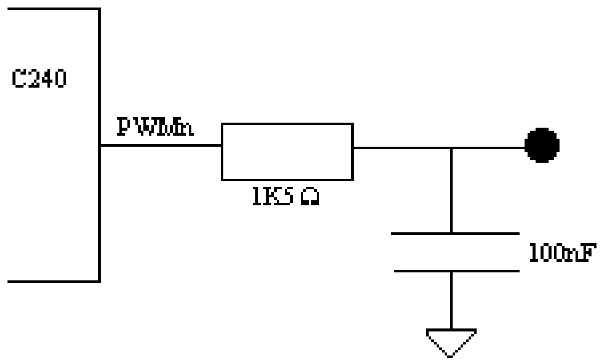
Table 5. Asymmetric and symmetric PWM resolution

	Asymmetric PWM	Symmetric PWM
PWM freq	24KHz	24KHz
PWM resolution	50nS	100nS
Effective resolution	10 bits	9 bits

Verification of space vector PWM algorithm

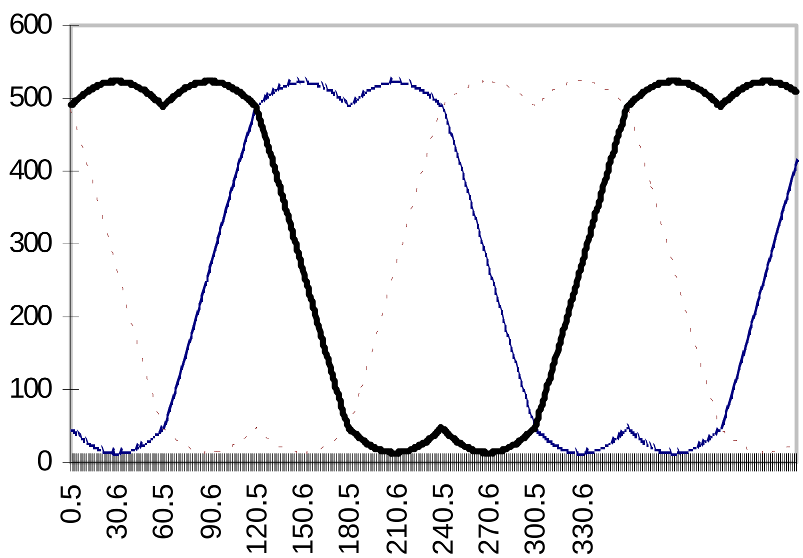
The correctness of the space vector PWM algorithm can be verified by probing the filtered PWM outputs of C240 using a very simple low-pass filter with a cutoff frequency of $f_c \cong 1000$Hz and viewing the resultant signal on a scope. Figure 16 shows a simple setup which can be used to filter the PWM output, and Figure 17 shows the expected wave forms for all the phases.

Figure 16. Filtering the PWM outputs



Note the resulting wave forms are a “time-averaged” version of the PWM switching signals and show clearly the fundamental frequency at ω and the third harmonic at 3ω which is inherently generated by the space vector method. As expected the 3 wave forms are spaced 120° apart.

Figure 17. The wave form of filtered space vector PWM outputs

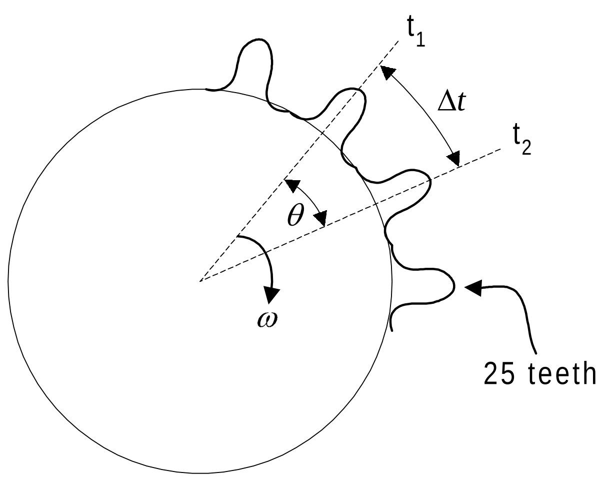


Measuring the motor shaft rotation speed

In order to have closed loop control of motor shaft speed, a speed error signal is required between the desired set speed and the actual measured speed. In the implementation described here a low cost shaft sprocket with 25 teeth and a hall effect gear tooth sensor is used with good results.

Figure 18 shows the physical details associated with the sprocket and how it relates to the angular velocity. The hall effect sensor outputs a square wave pulse every time a tooth rotates within it's proximity. The resultant pulse rate is 25 pulses / revolution. The hall effect output is fed directly to the C240 Capture 1 input where the tooth-to-tooth period (t_2-t_1) can be measured. In order to reduce jitter or period fluctuation, an average of the most recent 25 period measurements is performed each time a new pulse is detected. Although not shown in the equation for ω in Figure 19, the averaging performed is actually a "box-car" average in which the newest value replaces the oldest value in the 25 deep circular buffer.

Figure 18. Speed measurement with a sprocket



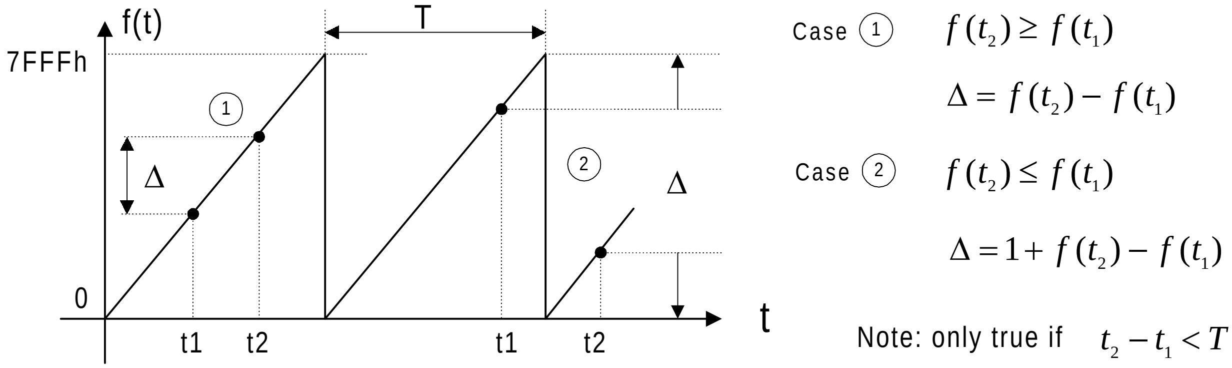
$$\Delta t = t_2 - t_1 \quad \text{sec}$$

$$\theta = \frac{360}{25} = 14.4^0$$

$$\omega = \frac{1}{\sum_{n=1}^{25} \Delta t_n} \quad \text{rev/sec}$$

The C240 capture unit allows accurate time measurement (in multiples of clock cycles and defined by a prescaler selection) between events. In this case the events are selected to be the rising edge of the incoming pulse train. What we are interested in is the delta time between events and hence for this implementation Timer 1 is allowed to free run with a prescale of 32 (1.6uS resolution) and the delta time Δ, in clock counts, is calculated according to Figure 19.

Figure 19. Calculation of speed

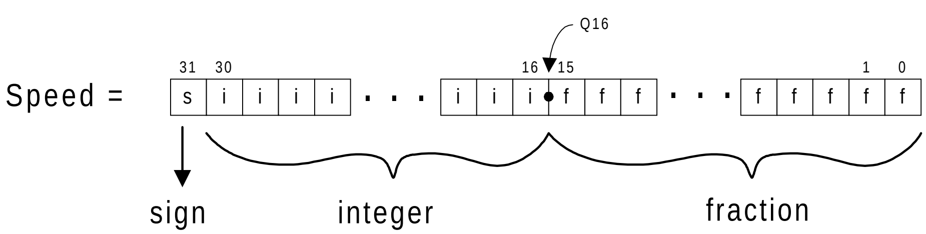


In Figure 19, the vertical axis *f(t)* represents the value of the Timer counter which is running in continuous up count mode and resetting when the period register = 7FFFh. This allows Δ to be calculated in *Q*15 format for later use in the averaging and division algorithms. Note that 2 cases need to be accounted for: 1. the simple case where the Timer has not wrapped around and; 2. where it has wrapped around. By keeping the current and previous capture values it is easy to test for each of these cases.

Once a “robust” period measurement is extracted from the averaging algorithm, the angular velocity (frequency) needs to be calculated by inverting the period value. In order to maintain high precision in the calculation for the full range of motor speeds, a 32-bit/16bit division is performed as shown in Figure 20 in the following.

Figure 20. 32-bit/16bit division.

$$\frac{1}{period} = \frac{7FFFFFFF(Q31)}{period(Q15)} = speed(Q16 \rightarrow 32bit)$$



Below is the code section which performs the 32bit/16bit division using the SUBC instruction. Once complete the result is a 32 bit value in *Q*16 format containing both an integer and fractional part. This value is subsequently scaled to a 16 bit, *Q*15 format value for later calculation of the speed error (see Figure 20).

```
;Calculate the Speed of rotation, i.e. speed =
;1/period
      ;Phase 1
      LACC  #07FFFh     ;Load Numerator Hi
      RPT   #15
      SUBC  BCAVG
      SACL  SPEED_HI
      XOR   SPEED_HI
      OR    #0FFFFh     ;Load Numerator Lo
      ;Phase 2
      RPT   #15
      SUBC  BCAVG
      SACL  SPEED_LO    ;Result is in Q16 in 32 bit
                        ; fmt

   ;Scale 32 bit value to fit in 16 bits (i.e. Q15)
      LACC  SPEED_LO
      ADDH  SPEED_HI
      SFL
      SACH  SPEED_fb, 7  ;SPEED_fb = SPEED_HI:LO x
                         ;256
```

Closed loop speed control

The Closed loop speed control described here uses the classical Proportional-Integral (PI) controller implemented in the digital or discrete (Z) domain.

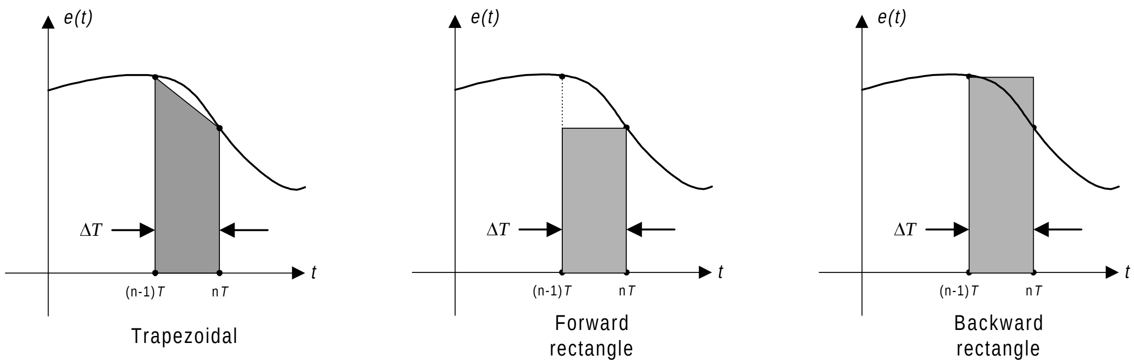
A PI controller in the continuous-data domain is given by:

$$G(s) = K_p + \frac{K_i}{s} \qquad (23)$$

There are a number of numerical integration rules that can be used to digitally approximate the integral controller K_i / s. As is well known, the 3 basic methods to approximate the area of a function numerically are: 1. Trapezoidal, 2. Forward rectangular and 3. Backward rectangular integration rules. These are shown below in Figure 21. It is clear from Figure 24 that if the sampling rate is increased, i.e.$\Delta T \rightarrow 0$ then all three methods above produce equally good results.

In the implementation described here the sampling rate used for the PWM switching (24KHz) is also used for the speed loop, i.e. ΔT = 41.6uS. For an ACI motor speed loop this is much higher than is typically needed, but works out very conveniently here since the C240 allows ample spare time within the 41.6uS PWM ISR to process the entire application including the PI controller.

Figure 21. Approximating the integral



Hence with the high sampling rate the choice of integration method is of very little importance and the simplest approach is to use the forward rectangle rule which gives the following Z-transform for the integral:

$$\frac{K_i}{s} \Leftrightarrow \frac{K_i}{1-z^{-1}} \qquad (25)$$

The final difference equation for the PI controller implemented here is then:

$$Y_n = Y_{n-1} + K_i e_n + K_p e_n \qquad (26)$$

The complete diagram with all the control blocks of the system implemented here is shown in Figure 22.

In focusing on the PI portion of Figure 22, the following things are worth noting in the software section given below:

- Variables ω_{sp} (set speed), ω_{fb} (speed feedback) & ω_e (speed error) are all represented in *Q*15 format.
- The integrator part of the difference equation $Y_n=Y_{n-1}+K_i.e_n$ is actually performed in 32 bit precision, the accumulation or integration register is in *Q*31 format, defined as a 32 bit quantity.

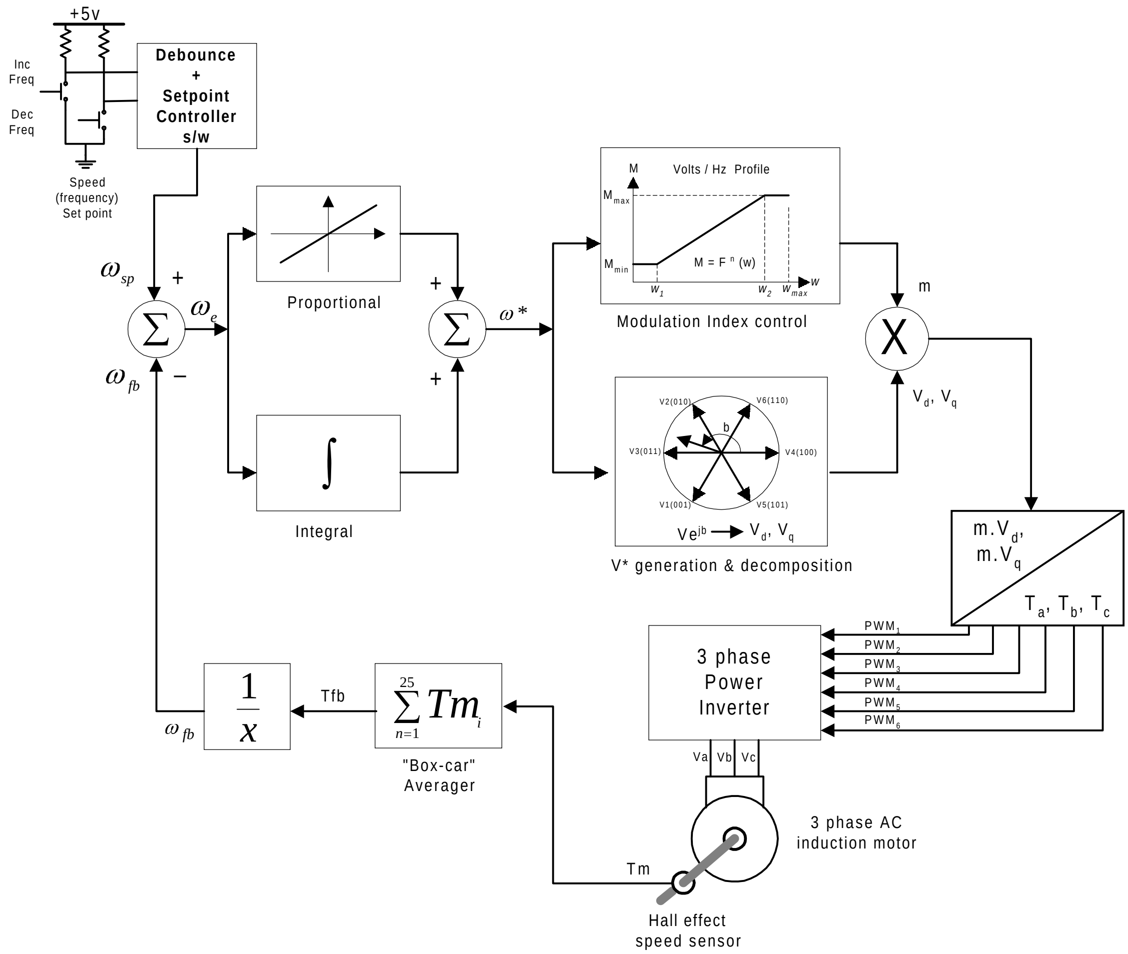
- The resulting controller output, ω^* is converted to *Q*0 format and scaled such that the range is 0 → 2048, this corresponds to a range of input Voltage frequency of 0 → 125Hz. ω^* is actually the Step angle which is used in the V* placement in the d-q plane as discussed previously.

- ω^* is also used in the calculation of the modulation index M by the V/Hz profile block. The V/Hz profile block consists of 3 sections. An upper & lower limit and a proportional range between the range of $\omega_1 \rightarrow \omega_2$. Every value of ω^* is mapped to corresponding value of M.

- The PI controller runs open-loop for the first 2 sec after startup so as to allow the motor to start spinning whereby giving a valid speed measurement from the hall effect speed sensor.

- In the final implementation the values for constants Ki & Kp were chosen by trial & error. Although not a very “scientific” method it worked out quite convenient since no parameter data was available for the low-cost “off the shelf” AC induction motor being used. Initially both Ki & Kp were chosen very small and then increased (one at a time) until instability (oscillations or hunting) was encountered. The value was then simply backed off a little.

Figure 22. The overall block diagram of Implementation II



Below is the code section which implements the PI controller:

```
;-------------------------------------------------------------------
; PI controller section
;-------------------------------------------------------------------
;Calculate Speed Setpoint - i.e. the Setpoint for the system.
;This variable is used in the error calculation when closing the
;loop.
        LT    FREQ_SETPT       ;SPEED_sp = FREQ_SETPT x 22
        MPY   #22              ;22 is a scaling factor
        PAC
        SACL  SPEED_sp

;The PI controller uses STEP_ANGLE to control the Stator MMF
```

```
;rotation speed. STEP_ANGLE has a range of 1(0.06Hz) --> 2048(125Hz)

PI_LOOP POINT_B0
        SPM   1                 ;Allow shift left of 1
        LACC  SPEED_sp
        SUB   SPEED_fb          ;Calculate error term E
        SACL  SPD_ERROR         ;Save it

;----------------------------------
;Check if it's time to close the PI loop. PI loop is closed only
;after a start-up delay in which motor starts to spin & the Speed
;measurement from the Hall sensor is valid
        LACC  LOOP_ON_FLG       ;Skip PI loop if Flag not
        SUB   #0Fh              ;set.
        BCND  PROFILE1, NEQ
;----------------------------------

PL_1    LT SPD_ERROR
        MPY   Ki
        PAC                     ;ACC = E*Ki
        ADD   STEP_INTEG_L      ;Keep a 32 bit integ value I
        ADDH  STEP_INTEG_H      ;I = I + Ki*E (E=SPD_ERROR)
        SACL  STEP_INTEG_L      ;Store Low word
        SACH  STEP_INTEG_H      ;Store Hi word

        LT    SPD_ERROR
        MPY   Kp                ;
        APAC                    ;ACC  = I + Ki*E + Kp*E
        SACH  GPR0              ;GPR0 = I + Ki*E + Kp*E

        LT    GPR0              ;GPR0 is in Q15
        MPY   #2048             ;2048 is in Q0
        PAC                     ;Result is Q15 in 32bit format
        SACH  STEP_ANGLE        ;Store as Q0
```

Experimental Results

Experimental Data of Implementation I

Experimental figures are presented in this section to demonstrate the effectiveness of the discussed algorithms. Figure 23 and Figure 24 show the current wave forms and spectrums obtained with the first implementation. A motor with fan load controlled by an experimental Spectrum Digital power converter is used in this setup. The TMS320C240/F240 EVM is used to run the motor control program. The converter is designed to interface directly with the TMS320C240/F240 EVM. The motor is a 4-pole 3-phase AC induction motor rated at 60Hz, 144V and 1/3hp. It can be seen that little or no harmonics are present in the current spectrums.

Figure 23. Motor current and its spectrum obtained with implementation I for F=25Hz

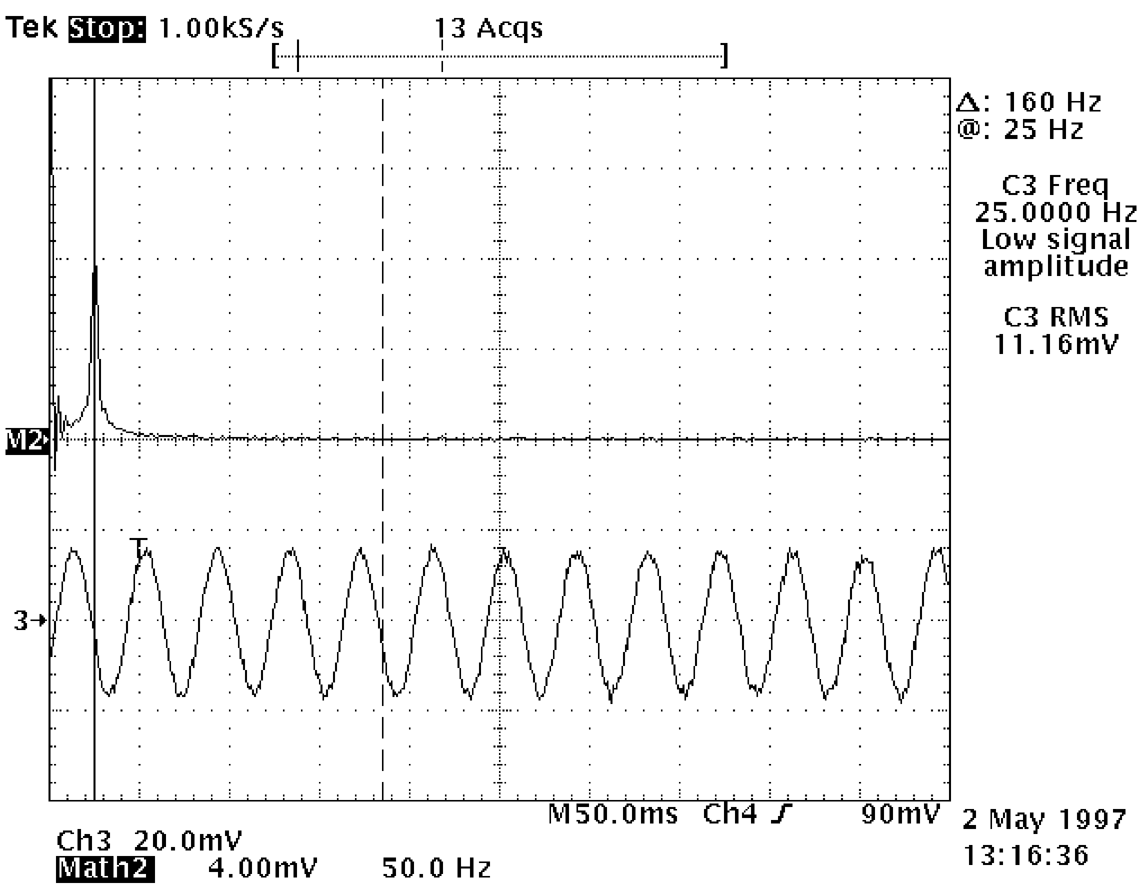
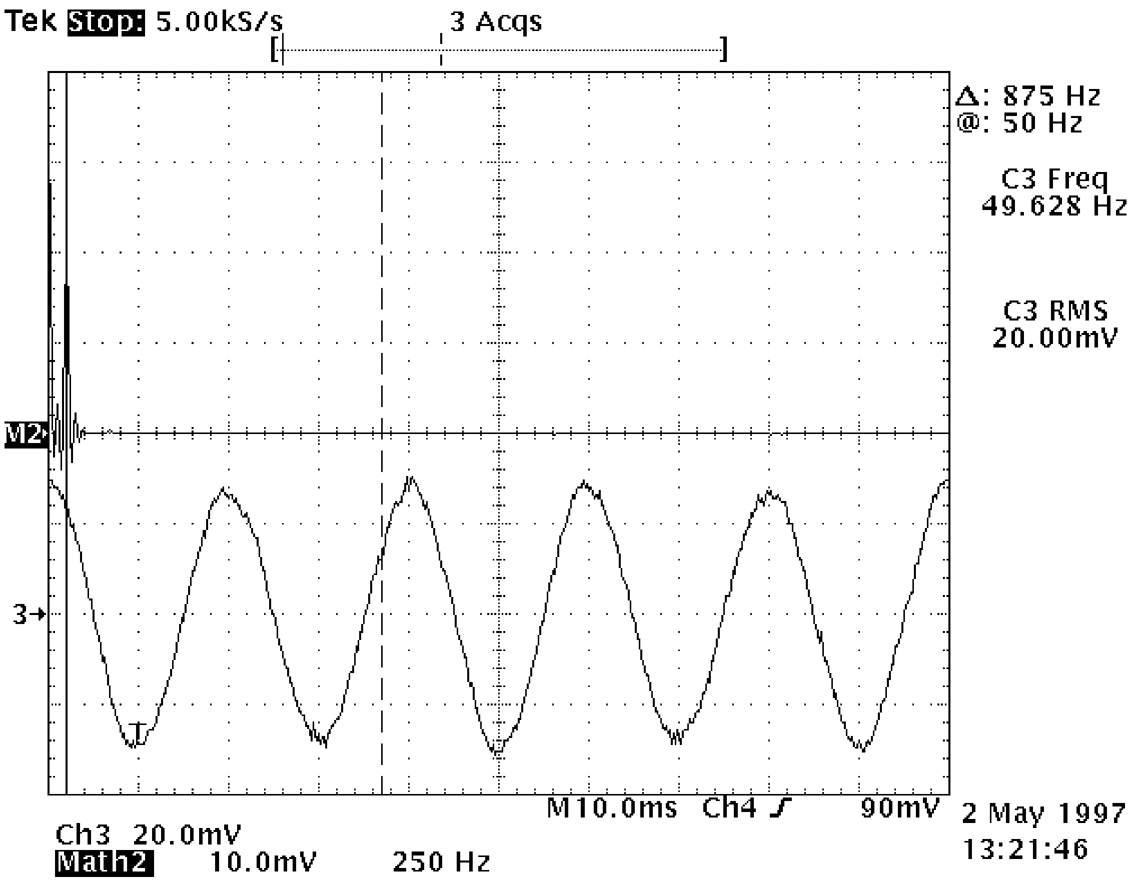


Figure 24. Motor current and spectrum obtained with implementation I for F=55Hz



Experimental Data of Implementation II

Figure 25 and Figure 26 show the current wave forms and spectrums obtained with the second implementation. A generic off-the-shelf GE motor rated at 60Hz, 220V and 1/4hp and a beta power converter from International Rectifier are used in the setup. The TMS320C240/F240 test-bed is used to run the motor control program. It can be seen again that little or no harmonics are present in the current spectrums.

Figure 25. Motor current and current spectrum obtained with implementation II, F_{in}=30Hz

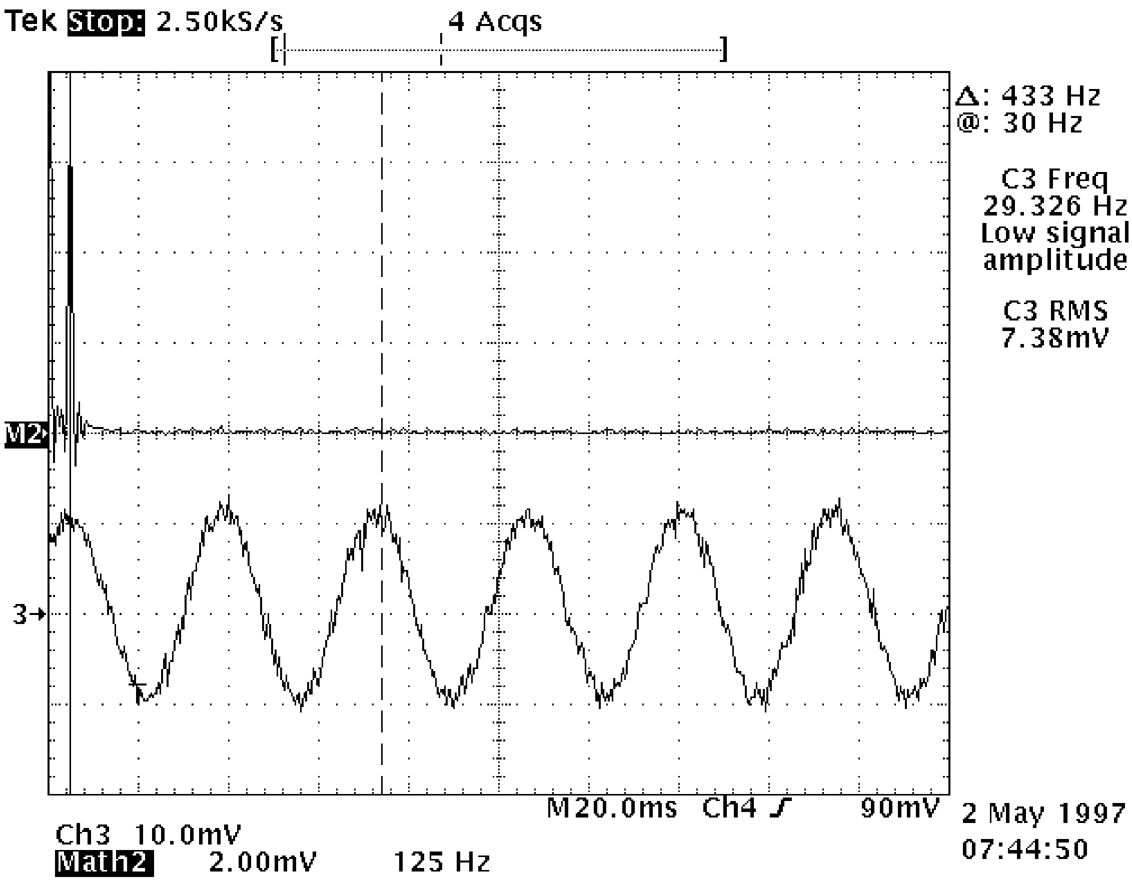
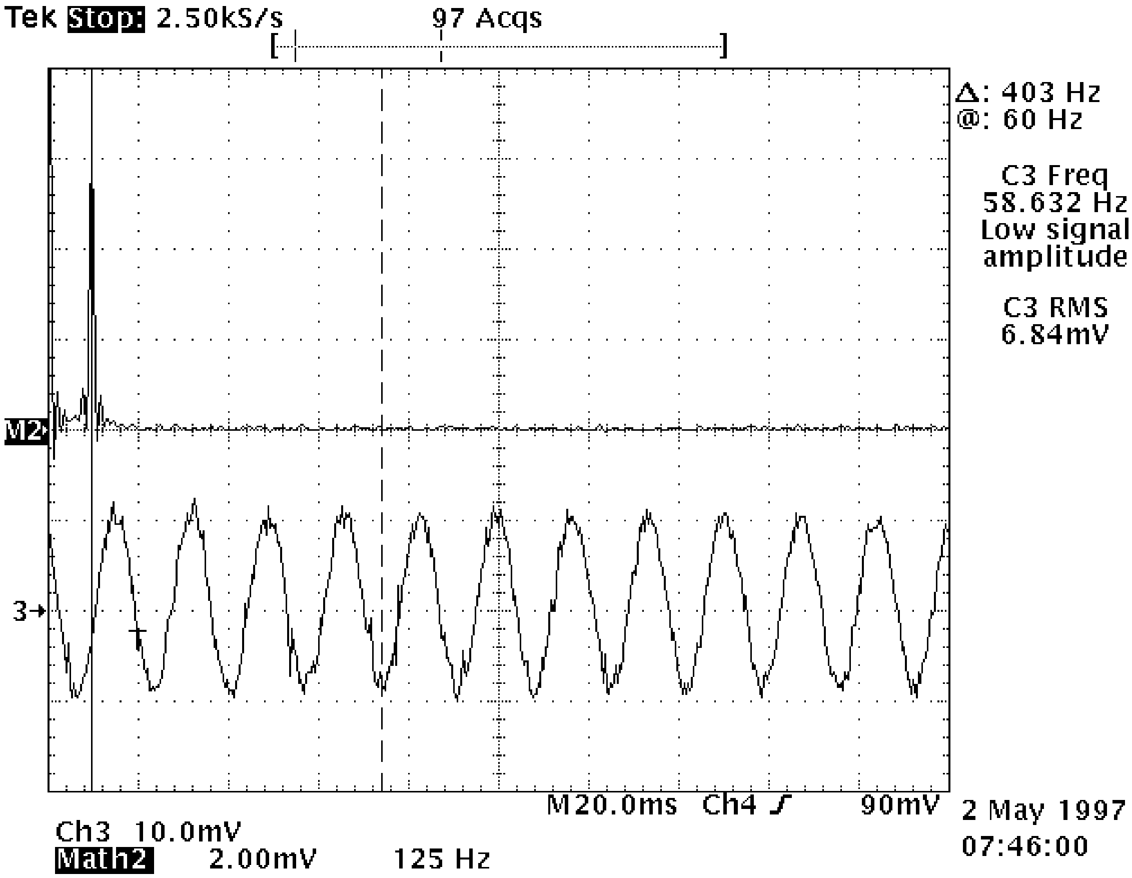


Figure 26. Motor current and spectrum obtained with implementation II, F_{in}=60Hz



References

1) Trzynadlowski, A.M., The field orientation principle in control of induction motors, Kluwer Academic, 1994.

2) Ogasawara, S., Akagi, H. et al, A novel PWM scheme of voltage source inverters based on space vector theory, EPE, Aachen, 1989.

Appendix I. Open-loop speed control for AC induction motor based on constant V/Hz principle and space vector PWM

```
*********************************************************************
** File Name     : sv25_attach.asm                                 **
** Project       : ACI motor drive                                 **
** Originator    : Zhenyu Yu                                       **
**                 Texas Instruments                               **
**                 DSP Automotive/Industrial Applications          **
**Target         : TMS320C240/F240(EVM)                            **
*********************************************************************


;---------------------------------------------------------------------
; Description
;---------------------------------------------------------------------
; This program implements an open-loop speed control algorithm for
; three-phase AC induction motors using constant V/Hz principle and
; space vector PWM technique.

;---------------------------------------------------------------------
; Notes
;---------------------------------------------------------------------
; 1. This program implements a sampling loop to carry out all the
;    calculations. The PWM and sampling frequencies are
;    independently controlled.
; 2. Constant V/Hz principle is used to generate the magnitude of
;    voltage command from frequency input;
; 3. Space vector PWM technique is used to generate the pulse-width
;    modulated signals controlling a three-phase voltage source
;    power
;    inverter so that desired voltage magnitude and frequency are
;    applied to the phased of a three-phase AC induction motor.
; 4. Both the PWM and sampling frequencies have been chosen to be
;    20KHz.
; 5. Maximum scaling and 32 bit integration are used to maximize the
```

```
;    accuracy of integer math involved to achieve a better dynamic
;    response.
; 6. The D scaling notation used here is equivalent to the popular Q
;    notation based on equation Dx=Q(15-x).
; 7. The motor is rated at 60Hz (that is, maximum duty ratio is
;    achieved when input is 60Hz).
; 8. Frequency input is through an ADC interface, ADC value 0
;    corresponds to 0Hz, ADC value 7fe0h corresponds to 120Hz.

;--------------------------------------------------------------------
; Peripheral Registers and constants of TMS320C240
;--------------------------------------------------------------------
         .include "c240app.h"
ST0      .set  0                 ; status register ST0
ST1      .set  1                 ; status register ST1
wd_rst_1 .set  055h              ; watchdog timer reset strings
wd_rst_2 .set  0aah              ;
LED_addr .set  0Ch               ; addr of LED display on EVM

;--------------------------------------------------------------------
; Variable definitions
;--------------------------------------------------------------------
*********************************************************************
** Variables in B1 page 0                        **
*********************************************************************
         .bss  GPR0,1            ; temporary storage

         .bss  one,1             ; +1

         .bss  wd_period,1       ; watchdog timer period
         .bss  wd_reset1,1       ; watchdog timer reset string 1
         .bss  wd_reset2,1       ; watchdog timer reset string 2

         .bss  period_flag,1     ; period start flag
```

```
        .bss  adc0_7,1         ; adc 0, channel 0 data
        .bss  adc0_6,1         ; adc 0, channel 1 data
        .bss  adc0_5,1         ; adc 0, channel 2 data
        .bss  adc1_15,1        ; adc 1, channel 0 data
        .bss  adc1_14,1        ; adc 1, channel 1 data
        .bss  adc1_13,1        ; adc 1, channel 2 data

        .bss  A_W,1            ; D10, ADC data to set W ratio
        .bss  A_U,1            ; D1, ADC data to set U ratio
        .bss  S_W,1            ; set angular speed: D11
        .bss  min_W,1          ; lower limit on set W (frequency)
        .bss  S_U,1            ; normalized set voltage: D2
        .bss  max_U,1          ; upper limit on set U: D2
        .bss  min_U,1          ; lower limit on set U: D2

        .bss  T_sample,1       ; sampling period: D-9

        .bss  THETAH,1         ; D3, angular position higher word
        .bss  THETAL,1         ; angular position lower word
        .bss  theta_r,1        ; rounded THETAH

        .bss  theta_m,1        ; D3, THETA mapped to 1st quadrant
        .bss  theta_1stent,1   ; beginning of theta table
        .bss  SS,1             ; sin sign modification: D15
        .bss  SC,1             ; cos sign modification: D15
        .bss  SP,1             ; sin table entry
        .bss  SIN_1stent,1     ; beginning of sin table
        .bss  SIN_lastent,1    ; end of sin table
        .bss  sin_theta,1      ; sin(THETA): D1
        .bss  cos_theta,1      ; cos(THETA): D1
        .bss  Ud,1             ; voltage Ud: D4
        .bss  Uq,1             ; voltage Uq: D4
        .bss  S,1              ; D15, sector reference U is in

        .bss  theta_60,1       ; 60: D3
```

```
        .bss  theta_90,1       ; 90: D3
        .bss  theta_120,1      ; 120: D3
        .bss  theta_180,1      ; 180: D3
        .bss  theta_240,1      ; 240: D3
        .bss  theta_270,1      ; 270: D3
        .bss  theta_300,1      ; 300: D3
        .bss  theta_360,1      ; 360: D3

        .bss  decpar_1stent,24 ; Decomposition matrices: D10

        .bss  cmp_1,1          ; component on 1st basic sp vector
        .bss  cmp_2,1          ; component on 2nd basic sp vector
        .bss  cmp_0,1          ; component on 0 basic sp vector /2

        .bss  CL,1             ; channel to toggle 1st
        .bss  CM,1             ; channel to toggle 2nd

        .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_count,1      ; sub-divider counter for LED

********************************************************************
** Variables in B2                      **
********************************************************************
ST0_save .SET  060h             ; saved status register ST0
ST1_save .set  061h             ; saved status register ST1
ACCH     .SET  062h             ; saved accumulator high
ACCL     .SET  063h             ; saved accumulator low
BSRS     .SET  064h             ; saved BSR
WSTORE   .SET  065h             ; working storage

;--------------------------------------------------------------------
; Program parameters
;--------------------------------------------------------------------
```

```
; Debug data used to substitute ADC input to debug the program.
debug_data  .set  01aa5h           ; 100Hz,6a9d)(50Hz,354b)(25Hz,1aa5

; ADC to radian frequency conversion ratio given by
; 120*2*pi/7fe0h(D0)=05721018.
; 7fe0h corresponds to 120Hz (754.3512 rad/Sec)
adc_to_afrequency.set 24222       ; D10
A_W_  .set  adc_to_afrequency     ; D10

; Min input frequency.
; User's choice
min_afrequency .set 0              ; D11
min_W_   .set  min_afrequency      ; D11

; ADC to magnitude of reference voltage conversion ratio
; 1.0/sqrt(2)/ADC(60Hz)(D0).
; Motor is rated at 60Hz meaning max duty ratio is achieved at 60Hz.
adc_to_voltage .set 11630          ; D2
A_U_  .set  adc_to_voltage         ; D2

; Max magnitude of reference voltage
; 1.0/sqrt(2)
max_voltage    .set 5792           ; D2
max_U_         .set  max_voltage   ; D2

; Min magnitude of reference voltage given by
; 1.0/sqrt(2)*min_f/60Hz
min_voltage    .set 0              ; D2
min_U_         .set  min_voltage   ; D2

; Timer 1 period which determines the PWM frequency.
T1_period_  .set  500
; Tp = 2*500*50nS=50uS => Fp = 20KHz

; Timer 2 period which determines the sampling frequency.
```

```
T2_period_  .set  500
; Ts = 2*500*50nS=50uS => Fs = 20KHz

; Max compare value
max_cmp_    .set  500

; Sampling period
T_sample_   .set  00346h   ; D-9, Ts = 50uS, Fs = 20KHz

;---------------------------------------------------------------------
; Memory resident program data
;---------------------------------------------------------------------
      .data

*********************************************************************
** Frequently used angles                           **
*********************************************************************

; The order between these angles and the decomposition matrices
; in the following must not be changed.
angles_  .WORD 010c1h   ; pi/3: D3
         .WORD 01922h   ; pi/2: D3
         .WORD 02183h   ; 2*pi/3: D3
         .WORD 03244h   ; pi
         .WORD 04305h   ; 4*pi/3: D3
         .WORD 04b66h   ; 3*pi/2: D3
         .WORD 053c7h   ; 5*pi/3: D3
         .WORD 06488h   ; 2*pi: D3

*********************************************************************
** Decomposition matrices indexed by the sector THETA (Uout) is in**
*********************************************************************
   .word       19595    ; D10
   .word       -11314
   .word       0
```

```
    .word       22627
    .word       -19595
    .word       11314
    .word       19595
    .word       11314
    .word       0
    .word       22627
    .word       -19595
    .word       -11314
    .word       0
    .word       -22627
    .word       -19595
    .word       11314
    .word       -19595
    .word       -11314
    .word       19595
    .word       -11314
    .word       19595
    .word       11314
    .word       0
    .word       -22627

********************************************************************
** Addresses of compare registers corresponding to channels to **
** toggle the 1st in a given period indexed by the sector THETA   **
** (Uout) is in.                                 **
********************************************************************
first_   .WORD CMPR1    ;
         .WORD CMPR2    ;
         .WORD CMPR2    ;
         .WORD CMPR3    ;
         .WORD CMPR3    ;
         .WORD CMPR1    ;

********************************************************************
```

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

********************************************************************
** Reset & interrupt vectors                                       **
********************************************************************
   .sect ".vectors"
RESET    B  START    ; PM 0   Reset Vector
INT1     B  PHANTOM  ; PM 2   Int level 1
INT2     B  PHANTOM  ; PM 4   Int level 2
INT3     B  EV_isr_B ; EV interrupt Group B
INT4     B  PHANTOM  ; PM 8   Int level 4
INT5     B  PHANTOM  ; PM A   Int level 5
INT6     B  PHANTOM  ; PM C   Int level 6
RESERVED B  PHANTOM  ; PM E   (Analysis Int)
SW_INT8  B  PHANTOM  ; PM 10  User S/W int
SW_INT9  B  PHANTOM  ; PM 12  User S/W int
SW_INT10 B  PHANTOM  ; PM 14  User S/W int
SW_INT11 B  PHANTOM  ; PM 16  User S/W int
SW_INT12 B  PHANTOM  ; PM 18  User S/W int
SW_INT13 B  PHANTOM  ; PM 1A  User S/W int
SW_INT14 B  PHANTOM  ; PM 1C  User S/W int
SW_INT15 B  PHANTOM  ; PM 1E  User S/W int
SW_INT16 B  PHANTOM  ; PM 20  User S/W int
TRAP     B  PHANTOM  ; PM 22  Trap vector
NMI      B  PHANTOM  ; PM 24  Non maskable Int
EMU_TRAP B  PHANTOM  ; PM 26  Emulator Trap
```

```
SW_INT20 B  PHANTOM  ; PM 28  User S/W int
SW_INT21 B  PHANTOM  ; PM 2A  User S/W int
SW_INT22 B  PHANTOM  ; PM 2C  User S/W int
SW_INT23 B  PHANTOM  ; PM 2E  User S/W int

         .text
********************************************************************
** Start of main body of code                         **
********************************************************************
START    dint           ; Set global interrupt mask

********************************************************************
** System configuration                            **
********************************************************************

; Configure system registers
; ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

; Point at Sys Module reg page 0
   LDP   #0E0h

; Disable watchdog timer if VCCP pin is at 5V
   SPLK  #06Fh, WD_CNTL

; Reset watchdog timer
   SPLK  #wd_rst_1,WD_KEY
   SPLK  #wd_rst_2,WD_KEY

; Set the source of CLKOUT to be CPUCLK
   SPLK  #0100000011000000b,SYSCR

; Clear all SYSSR register bits except HP0 (bit 5)
; FLASH programming and WD disabled allowed when bit 5 is 1.
; Note bit 5 is a read/clear bit. It can not be set.
   splk  #0000000000000100000b,SYSSR
```

```
; Configure PLL/Clocks to generate CPUCLK of 20MHz when CLKIN=10MHz
   SPLK  #0000000010110001b,CKCR1

; Disable and re-enable the PLL to make sure changes to CKCCR1
; happen
   SPLK  #0000000000000001b,CKCR0; Disable PLL
   SPLK  #0000000011000001b,CKCR0; Re-enable PLL

; Point to memory page 0 (B2)
   LDP  #0

; Configure wait state generator register so that no wait state is
; added for any off chip access
   SPLK #1000b,WSTORE
      OUT  WSTORE,0ffffh     ; WSGR <= (WSTORE)

; Point at Sys Module reg page 1
   LDP   #0E1h

; Configure i/o pins so that all pins shared by Event Manager
; are configured as Event Manager pins
; See comment lines for configuration of other pins
   SPLK  #0ff00H,OPCRA

* IOPA3/ADCIN8 => IOPA3
* IOPA2/ADCIN9 => IOPA2
* IOPA1/ADCIN1 => IOPA1
* IOPA0/ADCIN0 => IOPA0

   SPLK  #00f0H,OPCRB

* BIO_/IOPC3 => BIO_
* XF/IOPC2 => XF
```

```
* IOPC0/ADCSOC => IOPC0

; Configure the directions of all digital i/o pins to be input
   SPLK  #0000H,IOPA_DDR
   SPLK  #0000H,IOPB_DDR
   SPLK  #0000H,IOPC_DDR

********************************************************************
** Initialize peripherals                          **
********************************************************************

; Initialize and start ADC
; ~~~~~~~~~~~~~~~~~~~~~~~~

; Point at Sys Mod reg page 0
   LDP   #0E0h

; Set up ADC module with p/s=1, disable ext SOC and E.M. SOC,
; enable both ADC modules, disable ADC interrupt,
; select channel 15 (ADC1) and channel 7 (ADC0) and
; start conversion.
; ADC is re-started every time the previous conversion results are
; read.
   SPLK  #0000000000000011b,ADC_CNTL1
   SPLK  #0101100111111111b, ADC_CNTL0

; Point at Event Manager register page
   LDP   #232

;--------------------------------------------------------------------
; Initialize Event Manager
;--------------------------------------------------------------------

; Clear all Event Manager registers before proceeding further.
; Good to have even when reset works properly.
```

```
   SPLK  #0,T1CON      ;
   SPLK  #0,T2CON      ;
   SPLK  #0,T3CON      ;
   SPLK  #0,DBTCON     ;
   SPLK  #0,COMCON     ;
   SPLK  #0,CAPCON     ;
   SPLK  #0,T1CNT      ;
   SPLK  #0,T2CNT      ;
   SPLK  #0,T3CNT      ;

; Init GP Timer 1 period that determines the PWM frequency.
   SPLK  #T1_period_,T1PER

; Init GP Timer 2 period that determines the sampling frequency
; of speed loop.
      SPLK  #T2_period_,T2PER

; Init GP Timer 3 period for other use
   SPLK  #T1_period_,T3PER

; Kill all F. Comp/PWM outputs.
   SPLK  #T1_period_,CMPR1
   SPLK  #T1_period_,CMPR2
   SPLK  #T1_period_,CMPR3
; Let GP Timer compare outputs toggle (to have more things I can
; look at with an oscilloscope).
   SPLK  #200,T1CMP
   SPLK  #200,T2CMP
   SPLK  #200,T3CMP

; Define PWM output polarities.
   SPLK  #0000100110011001b,ACTR

* bits   15       0:    Dir = CCW (n/c)
* bits   14-12    000:  D2D1D0 = 000 (n/c)
```

```
* bits   11-10    10:   PWM6/CMP6 active high
* bits   9-8      01:   PWM5/CMP5 active low
* bits   7-6      10:   PWM4/CMP4 active high
* bits   5-4      01:   PWM3/CMP3 active low
* bits   3-2      10:   PWM2/CMP2 active high
* bits   1-0      01:   PWM1/CMP1 active low

; Mask PDPINT to prevent it from disabling the compare output
; enabling bits in COMCON, before I configure COMCON
   SPLK  #0h,IMRA
; Write COMCON twice to configure F&S compare units
   SPLK  #0000001100000111b,COMCON
   SPLK  #1000001100000111b,COMCON

* bit    15       1:    Enable Compare/PWM operation
* bits   14-13    00:   Load F. Comp. Registers on underflow of GPT1
* bit    12       0:    Disable Space Vector PWM Mode
* bits   11-10    00:   Load ACTR on underflow of GPt1
* bit    9        1:    Enable F Compare outputs
* bit    8        1:    Enable S Compare outputs
* bit    7        0:    Select GP Timer 1 as time base for S Comp
*                       Units
* bits   6-5      00:   Load SCMPR on Underflow of selected GP Timer
* bits   4-3      00:   Load SACTR on underflow of selected GP Timer
* bit    2        1:    F. Comp. Unit 3 in PWM mode
*                       PWM5/CMP5 & PWM6/CMP6 are PWM outputs
* bit    1        1:    F. Comp. Unit 2 in PWM mode
*                       PWM3/CMP3 & PWM4/CMP4 are PWM outputs
* bit    0        1:    F. Comp. Unit 1 in PWM mode
*                       PWM1/CMP1 & PWM2/CMP2 are PWM outputs

; Define GP Timer compare output polarities and GP Timer actions
   SPLK  #0000000001010101b,GPTCON

* bits   12-11    00:   No GP Timer 3 event starts ADC
```

```
* bits   10-9     00:   No GP Timer 2 event starts ADC
* bits   8-7      00:   No GP Timer 1 event starts ADC
* bit    6        1:    Enable GP Timer Compare outputs
* bits   5-4      01:   GP Timer 3 comp output active low
* bits   3-2      01:   GP Timer 2 comp output active low
* bits   1-0      01:   GP Timer 1 comp output active low

; Configure GP Timer 3
   SPLK  #1010100010000010b,T3CON

* bit    15       1:    FREE = 1
* bit    14       0:    SOFT = 0
* bits   13-11    101:  continuous-up/dn count mode
* bits   10-8     000:  Prescaler = /1
* bit    7        1:    Use Timer ENABLE of GP Timer 1
* bit    6        0:    Disable timer (counting operation
* bits   5-4      00:   Select internal CLK
* bits   3-2      00:   Load GP Timer comp register on underflow
* bit    1        1:    GP Timer compare enabled
* bit    0        0:    Use own PR

; Configure GP Timer 2
   SPLK  #1010100010000010b,T2CON

* bit    15       1:    FREE = 1
* bit    14       0:    SOFT = 0
* bits   13-11    101:  continuous-up/dn count mode
* bits   10-8     000:  Prescaler = /1
* bit    7        1:    Use Timer ENABLE of GP Timer 1
* bit    6        0:    Disable timer (counting operation)
* bits   5-4      00:   Select internal CLK
* bits   3-2      00:   Load GP Timer comp register on underflow
* bit    1        1:    GP Timer compare enabled
* bit    0        0:    Use own PR
```

```
; Configure GP Timer 1
   SPLK  #1010100000000010b,T1CON

* bit    15       1:    FREE = 1
* bit    14       0:    SOFT = 0
* bits   13-11    101:  continuous-up/dn count mode
* bits   10-8     000:  Prescaler = /1
* bit    7        0:    Reserved
* bit    6        1:    Disable Timer (counting operation)
* bits   5-4      00:   Select internal CLK
* bits   3-2      00:   Load GP Timer comp register on underflow
* bit    1        1:    GP Timer compare enabled
* bit    0        0:    reserved

   SPLK  #0fffh,IFRA   ; Clear all Group A interrupt flags
   SPLK  #0ffh,IFRB    ; Clear all Group B interrupt flags
   SPLK  #0fh,IFRC     ; Clear all Group C interrupt flag
   SPLK  #0h,IMRA      ; Mask all Grp A ints
   SPLK  #04h,IMRB     ; Mask all but GPT2 UF Grp B ints
   SPLK  #00h,IMRC     ; Mask all Grp C ints

********************************************************************
** Initialize variables                                 **
********************************************************************
; Point to B1 page 0
   LDP   #6

   SPLK  #1,one     ; +1 => one
   SPLK  #T_sample_,T_sample; sampling period
   SPLK  #A_W_,A_W     ; D8, ADC to set W ratio
   SPLK  #A_U_,A_U     ; D1, ADC to set U ratio
   SPLK  #min_W_,min_W ; lower limit on set W
   SPLK  #max_U_,max_U ; upper limit on set U
   SPLK  #min_U_,min_U ; lower limit on set U
```

```
   SPLK  #0,THETAL     ; theta low byte
   SPLK  #0,THETAH     ; theta high byte

   LAR   AR0,#theta_60 ; point to 1st destination
   LAR   AR1,#(32-1)   ; 32 entries
   LACC  #angles_      ; point to 1st data item
   LARP  AR0           ;
INITB
            TBLR  *+,1                ; move and point to next
destination
            ADD   one                 ; point to next data item
            BANZ  INITB,0             ;

; Init table 1st and last entries and table pointer
            SPLK  #TB_TH,theta_1stent
            SPLK  #1,SP
            SPLK  #TB_S,SIN_1stent
            SPLK  #(TB_S+180),SIN_lastent

********************************************************************
** More house keeping                        **
********************************************************************
; Set LED display on EVM
; ~~~~~~~~~~~~~~~~~~~~~~
            splk  #01h,LED_data       ;
            out   LED_data,LED_addr   ; Set LED display
            splk  #LED_freq,LED_count ; reset sub-divider counter
            splk  #1,LED_dir          ; set LED display direction

; point to memory page 0
            LDP   #0

            SPLK  #0ffh,IFR           ; Clear all core interrupt
                                      ; flags
            SPLK  #0eh,IMR            ; Unmask all EV interrupts to
```

```
                                    ; CPU

           SETC  OVM                ; Set overflow protect mode
           SETC  SXM                ; Set sign extension (allow)
                                    ; mode
           EINT                     ; Enable global interrupt

; Point at EV reg page
           LDP   #232

; Start all three GP Timers
           SPLK  #1010100001000010b,T1CON

********************************************************************
** Start of main loop                        **
********************************************************************
MAIN
; point at B1 page 0
           ldp   #6

; Wait for sampling period start.
w_sample   LACC  period_flag        ; Load sampling period start
                                    ; flag
           BZ    w_sample           ; Wait if not set

           SPLK  #0,period_flag     ; Reset the flag if it is set

; Toggle xf for debug purpose
           setc  xf                 ;

; Update LED display on EVM
           lacc  LED_count          ;
           sub   one                ; update sub_divide counter
           sacl  LED_count          ; time to update LED display?
           BNZ   LED_nc             ; no
```

```
            splk  #LED_freq,LED_count ; yes, reset subdivide counter
            bit   LED_dir,BIT0        ; left shift?
            bcnd  right_shift,NTC     ; no
            lacc  LED_data,1          ; yes
            sacl  LED_data            ; left shift one bit
            bit   LED_data,BIT7       ; time to change direction?
            bcnd  LED_update,NTC      ; no
            splk  #0,LED_dir          ; yes
            b     LED_update          ;
right_shift lacc  LED_data,15         ;
            sach  LED_data            ; right shift one bit
            bit   LED_data,BIT0       ; time to change direction?
            bcnd  LED_update,NTC      ; no
            splk  #1,LED_dir          ; yes
LED_update  out   LED_data,LED_addr   ; update LED display
LED_nc

*********************************************************************
** Read ref frequency                         **
*********************************************************************
; Point at Sys Mod reg page 0
            LDP   #0E0h

; Get ADC data and re-start ADC.
            lacl  ADC0_DATA           ; Get ADC data, 0 ACC high
            SPLK  #0101100111111111b, ADC_CNTL0; re-start ADC

; point at B1 page 0
            ldp   #6

; Right shift ADC data by one bit to get D0 representation.
            sfr                       ; shift right by one bit
            sacl  adc0_7              ; save (D0)

; Replace adc0_7 with debug data. For debug purpose only.
```

```
            SPLK  #debug_data,adc0_7  ;

*********************************************************************
** Calculate radian frequency                  **
*********************************************************************
            LT    adc0_7              ; load ADC data: D0
            MPY   A_W                 ; D0*D10=D(10+1)
            PAC                       ;
            SACH  S_W                 ; radian frequency: D11

            SUBH  min_W               ; compare W with its limit
            BGZ   W_in_limit          ; continue if within limit
            LACC  min_W               ; saturate if not
            SACL  S_W                 ;
W_in_limit

*********************************************************************
** Calculate magnitude of ref voltage Uout         **
*********************************************************************
; Note const. V/Hz is implied
            MPY   A_U                 ; D0*D1=D(1+1)
            PAC                       ;
            SACH  S_U                 ; set U: D2

            SUBH  max_U               ; compare Uout with its upper
                                      ; limit
            BLEZ  U_in_uplimit        ; continue if within limit
            LACC  max_U               ; saturate if not
            SACL  S_U                 ;
U_in_uplimit
            LACC  S_U                 ;
            SUB   min_U               ; compare Uout with its lower
                                      ; limit
            BGEZ  U_in_lolimit        ; continue if within limit
            LACC  min_U               ; saturate if not
```

```
            SACL  S_U                 ;
U_in_lolimit

********************************************************************
** Obtain theta (phase of Uout) through 32 bit integration     **
********************************************************************
            LT    S_W                 ;
            MPY   T_sample            ; D-9*D11=D(2+1)
            PAC                       ;

            ADDS  THETAL              ;
            ADDH  THETAH              ;
            SACH  THETAH              ;
            SACL  THETAL              ; accumulate: D3+D3=D3

            SUBH  theta_360           ; compare with 2*pi: D3-D3=D3
            BLEZ  Theta_in_limit      ; continue if within limit
            SACH  THETAH              ; mod(2*pi, THETA) if not
Theta_in_limit

            ZALH  THETAH              ;
            ADDS  THETAL              ;
            ADD   one,15              ;
            SACH  theta_r             ; round up to upper 16 bits

********************************************************************
** Determine quadrant                                            **
********************************************************************
; assume THETA (THETAH) is in quadrant 1
            LACC  one; assume THETA (THETAH) 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 (THETAH) is in quadrant 2 if not
            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 (THETAH) is in quadrant 3 if not
            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 (THETAH) is in quadrant 4 if not
            splk  #1,SC                 ; 1=>SC
            LACC  theta_360             ;
            SUB   theta_r               ;
            SACL  theta_m               ; 360-THETAH=>theta_m
E_Q

*********************************************************************
** Obtain theta table entry                                        **
*********************************************************************
            LACC  theta_1stent          ;
            ADD   SP                    ;
            TBLR  GPR0                  ; get table(SP)

            LACC  theta_m               ;
            SUB   GPR0                  ; compare theta_m with
                                        ; table(SP)
```

```
            BZ look_end                 ; end look-up if equal
            BGZ   inc_SP                ; increase SP if bigger

dec_SP      LACC  SP                    ; decrease SP other wise
            SUB   one                   ;
            SACL  SP                    ; SP-1=>SP

            ADD   theta_1stent          ; point to SP-1
            TBLR  GPR0                  ; get table(SP-1)

            LACC  theta_m               ;
            SUB   GPR0                  ; compare theta_m with
                                        ; table(SP-1)
            BLZ   dec_SP                ; decrease SP further if
                                        ; smaller
            B     look_end              ; jump to end if not

inc_SP      LACC  SP                    ;
            ADD   one                   ;
            SACL  SP                    ; SP+1=>SP

            ADD   theta_1stent          ; point to SP+1
            TBLR  GPR0                  ; get table(SP+1)

            LACC  theta_m               ;
            SUB   GPR0                  ; compare theta_m with
                                        ; table(SP+1)
            BGZ   inc_SP                ; increase further if bigger
look_end                                ; end if not

*********************************************************************
** Get sin(theta)                                                  **
*********************************************************************
            LACC  SIN_1stent            ;
            ADD   SP                    ;
```

```
          TBLR  sin_theta           ; get sin(THETA)

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

*********************************************************************
** Get cos(theta)                     **
*********************************************************************
          LACC  SIN_lastent         ;
          SUB   SP                  ;
          TBLR  cos_theta           ; get cos(THETA)

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

*********************************************************************
** Calculate Ud & Uq                                               **
*********************************************************************
          LT    S_U                 ;
          MPY   cos_theta           ; Uref*cos(THETA): D2*D1=D(3+1)
          PAC                       ;
          SACH  Ud                  ;
          MPY   sin_theta           ; Uref*sin(THETA): D2*D1=D(3+1)
          PAC                       ;
          SACH  Uq                  ;

*********************************************************************
** Determine sector                                                **
*********************************************************************
```

```
MAR   *,AR0              ; ARP points to AR0
LAR   AR0,#1             ; assume S=1

LACC  theta_r            ;
SUB   theta_60           ; compare with 60 (in sector
                         ; 1?)
BLEZ  E_S                ; jump to end if yes
MAR   *+                 ; assume S=2 if not

LACC  theta_r            ;
SUB   theta_120          ; compare with 120 (in sector
                         ; 2?)
BLEZ  E_S                ; jump to end if yes
MAR   *+                 ; assume S=3 if not.

LACC  theta_r            ;
SUB   theta_180          ; compare with 180 (in sector
                         ; 3?)
BLEZ  E_S                ; jump to end if yes
MAR   *+                 ; assume S=4 if not

LACC  theta_r            ;
SUB   theta_240          ; compare with 240 (in sector
                         ; 4?)
BLEZ  E_S                ; jump to end if yes
MAR   *+                 ; assume S=5 if not

LACC  theta_r            ;
SUB   theta_300          ; compare with 300 (in sector
                         ; 5?)
BLEZ  E_S                ; jump to end if yes
MAR   *+                 ; S=6 if not
SAR   AR0,S              ;
```

```
*****************************************************
** Calculate T1 & T2 based on                      **
**          Tpwm Uout = V1*T1 + V2*T2               **
** or                                              **
**          [T1 T2] = Tpwm [V1 V2]' Uout            **
**          [0.5*T1 0.5*T2] = Tp [V1 V2]' Uout      **
**                  = Mdec(S) Uout                 **
** where                                           **
**          Mdec(S) = Tp [V1 V2]'                  **
**          Uout = Trans([Ud Uq])                  **
** Mdec is obtained through table look-up.         **
** Note that timer period is half of PWM period,   **
** i.e. Tp=0.5 Tpwm.                               **
*****************************************************
            LACC  #(decpar_1stent-4)
            ADD   S,2                   ;
            SACL  GPR0                  ; get the pointer
            LAR   AR0,GPR0              ; point to parameter table

            LT    Ud                    ; calculate 0.5*T1
            MPY   *+                    ; M(1,1) Ud: D4*D10=D(14+1)
            PAC                         ;
            LT Uq                       ;
            MPY   *+                    ; M(1,2) UqP D4*D10=D(14+1)
            APAC                        ; 0.5*T1: D15+D15=D15
            BGEZ  cmp1_big0             ; continue if bigger than zero
            ZAC                         ; zero it if less than zero
cmp1_big0   SACH  cmp_1                 ;

            LT    Ud                    ; Calculate 0.5*T2
            MPY   *+                    ; M(2,1) Ud: D4*D10=D(14+1)
            PAC                         ;
            LT    Uq                    ;
            MPY   *+                    ; M(2,2) Uq: D4*D10=D(14+1)
            APAC                        ; 0.5*T2: D15+D15=D15
```

```
            BGEZ  cmp2_big0           ; continue if bigger than zero
            ZAC                       ; zero it if less than zero
cmp2_big0   SACH  cmp_2               ;

            LACC  #max_cmp_           ; Calculate 0.5*T0
            SUB   cmp_1               ;
            SUB   cmp_2               ; 0.5*T0 = Tp - 0.5*T1 –
                                      ; 0.5*T2: D15
            BGEZ  cmp0_big0           ; continue if bigger than zero
            ZAC                       ; zero it if less than zero
cmp0_big0   SACL  cmp_0               ;

            LACC  cmp_0,15            ; left shift 15 (right shift 1)
                                      ; bit
            SACH  cmp_0               ; 0.25*T0

*********************************************************************
** Determine the channel toggling sequence and load compare values**
*********************************************************************
            LACC  #(first_-1)         ;
            ADD   S                   ; point to entry in look up
                                      ; table
            TBLR  CL                  ; get the channel to toggle the
                                      ; 1st
            LAR   AR0,CL              ; point to the 1st channel
            LACC  cmp_0               ;
            SACL  *                   ; cmp_0=>the 1st channel

            LACC  #(second_-1)        ;
            ADD   S                   ; point to entry in look up
                                      ; table
            TBLR  CM                  ; get the channel to toggle the
                                      ; 2nd
            LAR   AR0,CM              ; point to the 2nd channel
            LACC  cmp_0               ;
```

```
            ADD   cmp_1               ; cmp_0+cmp_1
            SACL  *                   ; => the 2nd channel

            LACC  #CMPR3              ;
            SUB   CL                  ;
            ADD   #CMPR2              ;
            SUB   CM                  ;
            ADD   #CMPR1              ;
            SACL  GPR0                ; get the channel to toggle the
                                      ; 3rd
            LAR   AR0,GPR0            ; point to the 3rd channel
            LACC  cmp_0               ;
            ADD   cmp_1               ;
            ADD   cmp_2               ; cmp_0+cmp_1+cmp_2
            SACL  *                   ; =>the 3rd channel

**************************************
** Reset wd timer                   **
**************************************
; Point at Sys Module reg page 0
            LDP #0E0h

; Reset watchdog timer
            SPLK  #wd_rst_1,WD_KEY
            SPLK  #wd_rst_2,WD_KEY

**************************************
** Jump back to start of main loop  **
**************************************
            clrc  xf                  ; Debug signal
            B   MAIN                  ; Branch back

**************************************
** Handle interrupt                 **
**************************************
```

```
EV_isr_B
           SST #ST0,ST0_save         ; save status register ST0
           SST #ST1,ST1_save         ; save status register ST1
           LDP #0                    ; point to memory page 0
           SACH  ACCH                ;
           SACL  ACCL                ; save ACC

           LDP #232                  ; point to EV reg page
           LACC  IFRB                ; Read group B flag register
           SACL  IFRB                ; Clear all group B flags

           LDP #6                    ; Point to B1 page 0
           SPLK  #1,period_flag      ; set start flag

           LDP   #0                  ; point to memory page 0
           ZALH  ACCH                ;
           ADDS  ACCL                ; restore ACC
           LST   #ST1,ST1_save       ; restore status register ST1
           LST   #ST0,ST0_save       ; restore status register ST0
           EINT                      ;
           RET                       ;

PHANTOM
           SST   #ST0,ST0_save       ; save status register ST0
           SST   #ST1,ST1_save       ; save status register ST1
           LDP   #0                  ; point to memory page 0
           SACH  ACCH                ;
           SACL  ACCL                ; save ACC

           LDP   #0                  ; point to memory page 0
           SPLK  #00badh,B2_SADDR+15
                                     ; 6fh <= "BAD" value indicates
                                     ; error

           LDP   #0                  ; point to memory page 0
```

```
            ZALH  ACCH                  ;
            ADDS  ACCL                  ; restore ACC
            LST   #ST1,ST1_save         ; restore status register ST1
            LST   #ST0,ST0_save         ; restore status register ST0
            EINT                        ;
            RET                         ; return

;---------------------------------------------------
; Theta and sine table
;---------------------------------------------------
TB_TH       .word      0                   ; angles at 0.5 degree
                                        ; increment: D3
            .word   36
            .word      71
            .word      107
            .word      143
            .word      179
            .word      214
            .word      250
            .word      286
            .word      322
            .word      357
            .word      393
            .word      429
            .word      465
            .word      500
            .word      536
            .word      572
            .word      608
            .word      643
            .word      679
            .word      715
            .word      751
            .word      786
            .word      822
```

```
        .word       858
        .word       894
        .word       929
        .word       965
        .word       1001
        .word       1037
        .word       1072
        .word       1108
        .word       1144
        .word       1180
        .word       1215
        .word       1251
        .word       1287
        .word       1323
        .word       1358
        .word       1394
        .word       1430
        .word       1466
        .word       1501
        .word       1537
        .word       1573
        .word       1608
        .word       1644
        .word       1680
        .word       1716
        .word       1751
        .word       1787
        .word       1823
        .word       1859
        .word       1894
        .word       1930
        .word       1966
        .word       2002
        .word       2037
        .word       2073
```

```
.word       2109
.word       2145
.word       2180
.word       2216
.word       2252
.word       2288
.word       2323
.word       2359
.word       2395
.word       2431
.word       2466
.word       2502
.word       2538
.word       2574
.word       2609
.word       2645
.word       2681
.word       2717
.word       2752
.word       2788
.word       2824
.word       2860
.word       2895
.word       2931
.word       2967
.word       3003
.word       3038
.word       3074
.word       3110
.word       3146
.word       3181
.word       3217
.word       3253
.word       3288
.word       3324
```

```
        .word       3360
        .word       3396
        .word       3431
        .word       3467
        .word       3503
        .word       3539
        .word       3574
        .word       3610
        .word       3646
        .word       3682
        .word       3717
        .word       3753
        .word       3789
        .word       3825
        .word       3860
        .word       3896
        .word       3932
        .word       3968
        .word       4003
        .word       4039
        .word       4075
        .word       4111
        .word       4146
        .word       4182
        .word       4218
        .word       4254
        .word       4289
        .word       4325
        .word       4361
        .word       4397
        .word       4432
        .word       4468
        .word       4504
        .word       4540
        .word       4575
```

```
.word      4611
.word      4647
.word      4683
.word      4718
.word      4754
.word      4790
.word      4825
.word      4861
.word      4897
.word      4933
.word      4968
.word      5004
.word      5040
.word      5076
.word      5111
.word      5147
.word      5183
.word      5219
.word      5254
.word      5290
.word      5326
.word      5362
.word      5397
.word      5433
.word      5469
.word      5505
.word      5540
.word      5576
.word      5612
.word      5648
.word      5683
.word      5719
.word      5755
.word      5791
.word      5826
```

```
            .word       5862
            .word       5898
            .word       5934
            .word       5969
            .word       6005
            .word       6041
            .word       6077
            .word       6112
            .word       6148
            .word       6184
            .word       6220
            .word       6255
            .word       6291
            .word       6327
            .word       6362
            .word       6398
            .word       6434


TB_S:       .word       0  ; sin table: D1
            .word       143
            .word       286
            .word       429
            .word       572
            .word       715
            .word       857
            .word       1000
            .word       1143
            .word       1285
            .word       1428
            .word       1570
            .word       1713
            .word       1855
            .word       1997
            .word       2139
```

```
        .word       2280
        .word       2422
        .word       2563
        .word       2704
        .word       2845
        .word       2986
        .word       3126
        .word       3266
        .word       3406
        .word       3546
        .word       3686
        .word       3825
        .word       3964
        .word       4102
        .word       4240
        .word       4378
        .word       4516
        .word       4653
        .word       4790
        .word       4927
        .word       5063
        .word       5199
        .word       5334
        .word       5469
        .word       5604
        .word       5738
        .word       5872
        .word       6005
        .word       6138
        .word       6270
        .word       6402
        .word       6533
        .word       6664
        .word       6794
        .word       6924
```

```
        .word       7053
        .word       7182
        .word       7311
        .word       7438
        .word       7565
        .word       7692
        .word       7818
        .word       7943
        .word       8068
        .word       8192

        .word       8316
        .word       8438
        .word       8561
        .word       8682
        .word       8803
        .word       8923
        .word       9043
        .word       9162
        .word       9280
        .word       9397
        .word       9514
        .word       9630
        .word       9746
        .word       9860
        .word       9974
        .word       10087
        .word       10199
        .word       10311
        .word       10422
        .word       10531
        .word       10641
        .word       10749
        .word       10856
        .word       10963
```

```
.word       11069
.word       11174
.word       11278
.word       11381
.word       11484
.word       11585
.word       11686
.word       11786
.word       11885
.word       11982
.word       12080
.word       12176
.word       12271
.word       12365
.word       12458
.word       12551
.word       12642
.word       12733
.word       12822
.word       12911
.word       12998
.word       13085
.word       13170
.word       13255
.word       13338
.word       13421
.word       13502
.word       13583
.word       13662
.word       13741
.word       13818
.word       13894
.word       13970
.word       14044
.word       14117
```

```
.word       14189
.word       14260
.word       14330
.word       14399
.word       14466
.word       14533
.word       14598
.word       14663
.word       14726
.word       14788
.word       14849
.word       14909
.word       14968
.word       15025
.word       15082
.word       15137
.word       15191
.word       15244
.word       15296
.word       15346
.word       15396
.word       15444
.word       15491
.word       15537
.word       15582
.word       15626
.word       15668
.word       15709
.word       15749
.word       15788
.word       15826
.word       15862
.word       15897
.word       15931
.word       15964
```

```
.word     15996
.word     16026
.word     16055
.word     16083
.word     16110
.word     16135
.word     16159
.word     16182
.word     16204
.word     16225
.word     16244
.word     16262
.word     16279
.word     16294
.word     16309
.word     16322
.word     16333
.word     16344
.word     16353
.word     16362
.word     16368
.word     16374
.word     16378
.word     16382
.word     16383
.word     16384
```

Appendix II. Closed-loop speed control for AC induction motor based on constant V/Hz principle and space vector PWM

```
;*******************************************************************
; File Name:   SVPWM_FB.ASM
; Project:     Digital Motor Control Applications development.
;
; Originator: David Figoli
;              (Texas Instruments Ind/Auto Apps team - Houston)
;
; Target Sys: C240 EVM +
;              (ACI i/f board & Inverter from Spectrum Digital)
;
; Description:This is an implementation of a 3 phase Space vector
;              PWM control scheme for 3 phase AC Induction motor
;              driven via a power inverter. External frequency
;              control is provided by push buttons i.e. PB1 for Incr
;              & PB2 for decrement freq. This version has a fully
;              implemented Speed measurement routine based on Capture
;              input & using 25 teeth Sprocket & Hall sensor. Speed
;              (calculated from the Capture waveform) is used as the
;              feedback variable which is compared to a speed
;              setpoint value by the PI controller block when
;              maintaining closed loop speed  control.
;
; Status:      This version runs using the Full Compares, With GPT1
;              as the Time base & providing the Period control
;               - Asymmetrical PWM scheme
;                 (i.e. effective clk res = 40nS)
;               - PWM period = 41.6uS (24KHz)
;               - Effective #bits resolution per PWM period = 10
;                 (1024 counts)
;               - Effective forced "zero" vector=41.6-40.0=1.6 uS
;               - S/W has successfully run a 3 phase AC Induction
```

```
;                 motor, i.e.: GE 1/4 HP 60Hz 208-230V AC
;                 Rated speed 1725 (1800 no slip)
;
; Last Update:17-MAR 97
;*******************************************************************
;-------------------------------------------------------------------
; Debug directives
;-------------------------------------------------------------------

                 .def  GPR0       ;General purpose registers.
                 .def  GPR1
                 .def  GPR2
                 .def  ALPHA
                 .def  STEP_ANGLE
                 .def  FREQ_SETPT
                 .def  ENTRY_NEW
                 .def  ENTRY_OLD
                 .def  dx
                 .def  dy
                 .def  Ta
                 .def  Tb
                 .def  Tc
                 .def  V
                 .def  CAP_PERIOD
                 .def  CAP_NEW
                 .def  CAP_OLD
                 .def  SPEED_HI
                 .def  SPEED_LO
                 .def  SPEED_fb
                 .def  SPEED_sp
                 .def  SPD_ERROR
                 .def  BCAVG
                 .def  STEP_INTEG_H
                 .def  STEP_INTEG_L
                 .def  Kp
```

```
                  .def  Ki
                  .def  LOOP_ON_FLG

                  .include C240APP.H

;---------------------------------------------------------------------
; Constant Declarations
;---------------------------------------------------------------------
; Used by the SBIT0 & SBIT1 Macro
B15_MSK           .set  8000h      ;Bit Mask for 15
B14_MSK           .set  4000h      ;Bit Mask for 14
B13_MSK           .set  2000h      ;Bit Mask for 13
B12_MSK           .set  1000h      ;Bit Mask for 12
B11_MSK           .set  0800h      ;Bit Mask for 11
B10_MSK           .set  0400h      ;Bit Mask for 10
B9_MSK            .set  0200h      ;Bit Mask for 9
B8_MSK            .set  0100h      ;Bit Mask for 8
B7_MSK            .set  0080h      ;Bit Mask for 7
B6_MSK            .set  0040h      ;Bit Mask for 6
B5_MSK            .set  0020h      ;Bit Mask for 5
B4_MSK            .set  0010h      ;Bit Mask for 4
B3_MSK            .set  0008h      ;Bit Mask for 3
B2_MSK            .set  0004h      ;Bit Mask for 2
B1_MSK            .set  0002h      ;Bit Mask for 1
B0_MSK            .set  0001h      ;Bit Mask for 0

WSGR              .set  0FFFFh
DP_PF1            .set  0E0h       ;page 1 of peripheral file
                                   ; (7000h/80h)
DP_PF2            .set  0E1h       ;page 2 of peripheral file
                                   ; (7080h/80h)
DP_PF3            .set  0E2h       ;page 3 of peripheral file
                                   ; (7100h/80h)
DP_EV             .set  0E8h       ;EV register data mem page
                                   ; (7400h/80h)
```

```
;Space vector PWM constants
;-------------------------
F1               .set  0256       ;Low Freq point on profile  = 15Hz
F2               .set  1024       ;High Freq point on profile = 60Hz
VF_SLOPE         .set  15292      ;Volts/Hz slope 1.87 in Q13 format
INTERCEPT        .set  0546       ;Line equation intercept 0.07 in
                                  ; Q13
Vmax             .set  032767     ;0.99999..  in Q15
Vmin             .set  09830      ;0.30000..  in Q15
BCNT_MAX         .set  100        ;100x40uS=0.004 Sec depress to be
                                  ; valid
RMP_DLY_MAX      .set  100        ;100x40uS=0.004 sec between steps.
BC_SIZE          .set  25         ;Box car average size of 50
BC_BUF_STRT      .set  300h       ;Start of BC buffer
K_integ          .set  Ah         ;PI loop Integration constant
K_prop           .set  800h       ;PI loop Proportional constant
STALL_SLIP       .set  3000h      ;7FFFh = 5660 rpm
STARTUP_DELAY    .set  0FFFFh     ;FFFF x 40uS =2.6 sec
STRTUP_FREQ      .set  0250h      ;Open loop start up value
STRTUP_STP_ANGL  .set  0260h      ; "
STRTUP_INTEG     .set  02600h     ; "

;---------------------------------------------------------------------
; Variable Declarations for on chip RAM Block B0
;---------------------------------------------------------------------
                 .bss  GPR0,1        ;General purpose registers.
                 .bss  GPR1,1
                 .bss  GPR2,1
                 .bss  FREQ_SETPT,1  ;Value from 0 --> 255
                 .bss  FREQ_TRGT,1   ;Frequency Target value 0 -->
                                     ;255
                 .bss  B1_CNT,1      ;B1 button counter (Inc Freq)
                 .bss  B2_CNT,1      ;B2 button counter (Dec Freq)
                 .bss  RMP_DLY_CNT,1 ;Ramp rate in adjusting to
```

```
                        ;Target freq.
        .bss  REPRESS_DLY,1 ;Forced delay between
                        ;Represses.

        .bss  S_TABLE,1     ;Data addr to store Sine table
                        ;addr.
        .bss  ALPHA,1       ;Angle integrator
        .bss  STEP_ANGLE,1  ;Rate of change of
                        ;angle(angular speed)
        .bss  STEP_INTEG_H,1;PI integration variable 32 bit
                        ;value
        .bss  STEP_INTEG_L,1;
        .bss  ENTRY_NEW,1   ;Table lookup entry point
        .bss  ENTRY_OLD,1   ;
        .bss  SINVAL,1
        .bss  SR_ADDR,1
        .bss  SECTOR_PTR,1  ;Keeps track of current sector
                        ;(1-6)

        .bss  CAP_PERIOD,1  ;Capture period value (between
                        ;teeth)
        .bss  CAP_NEW,1
        .bss  CAP_OLD,1
        .bss  SPEED_HI,1    ;32 bit Speed calc working reg.
        .bss  SPEED_LO,1
        .bss  SPEED_fb,1    ;Speed feedback value
        .bss  SPEED_sp,1    ;Speed setpoint value
        .bss  SPD_ERROR,1   ;Error signal for PI loop
        .bss  Kp,1          ;Proportional constant
        .bss  Ki,1          ;Integral constant
        .bss  BCAVG,1       ;"Box car" average

;Space Vector PWM calc variables
        .bss  dx,1
        .bss  dy,1
```

```
                 .bss  T,1
                 .bss  Ta,1
                 .bss  Tb,1
                 .bss  Tc,1
                 .bss  V,1
                 .bss  vf_slope,1

   ;Convenience & debug feature variables
                 .bss  V_TIMER1,1    ;Virtual timer counter
                 .bss  LOOP_ON_FLG,1 ;Open/close loop control flag

   ;Stack space in B0
stk1             .usect "blk_b2", 1
stk2             .usect "blk_b2", 1
stk3             .usect "blk_b2", 1
stk4             .usect "blk_b2", 1

;---------------------------------------------------------------------
; M A C R O - Definitions
;---------------------------------------------------------------------
SBIT0            .macro   DMA, MASK   ;Clear bit Macro
                 LACC  DMA
                 AND   #(0FFFFh-MASK)
                 SACL  DMA
                 .endm

SBIT1            .macro   DMA, MASK   ;Set bit Macro
                 LACC  DMA
                 OR #MASK
                 SACL  DMA
                 .endm

KICK_DOG         .macro              ;Watchdog reset macro
                 LDP   #00E0h
                 SPLK  #05555h, WD_KEY
```

```
                  SPLK  #0AAAAh, WD_KEY
                  LDP   #0h
                  .endm

POINT_PG0         .macro
                  LDP   #00h
                  .endm

POINT_B0          .macro
                  LDP   #04h
                  .endm

POINT_PF1         .macro
                  LDP   #0E0h
                  .endm

POINT_PF2         .macro
                  LDP   #0E1h
                  .endm

POINT_EV          .macro
                  LDP   #0E8h
                  .endm

;---------------------------------------------------------------------
; Vector address declarations
;---------------------------------------------------------------------
     .sect  ".vectors"

RSVECT      B    START    ; PM 0 Reset Vector          1
INT1        B    PHANTOM  ; PM 2 Int level 1           4
INT2        B    PWM_ISR  ; PM 4 Int level 2           5
INT3        B    PHANTOM  ; PM 6 Int level 3           6
INT4        B    PHANTOM  ; PM 8 Int level 4           7
INT5        B    PHANTOM  ; PM A Int level 5           8
```

```
INT6        B    PHANTOM  ; PM C Int level 6         9
RESERVED    B    PHANTOM  ; PM E (Analysis Int)      10
SW_INT8     B    PHANTOM  ; PM 10  User S/W int      -
SW_INT9     B    PHANTOM  ; PM 12  User S/W int      -
SW_INT10    B    PHANTOM  ; PM 14  User S/W int      -
SW_INT11    B    PHANTOM  ; PM 16  User S/W int      -
SW_INT12    B    PHANTOM  ; PM 18  User S/W int      -
SW_INT13    B    PHANTOM  ; PM 1A  User S/W int      -
SW_INT14    B    PHANTOM  ; PM 1C  User S/W int      -
SW_INT15    B    PHANTOM  ; PM 1E  User S/W int      -
SW_INT16    B    PHANTOM  ; PM 20  User S/W int      -
TRAP        B    PHANTOM  ; PM 22  Trap vector       -
NMI         B    PHANTOM  ; PM 24  Non maskable Int  3
EMU_TRAP    B    PHANTOM  ; PM 26  Emulator Trap     2
SW_INT20    B    PHANTOM  ; PM 28  User S/W int      -
SW_INT21    B    PHANTOM  ; PM 2A  User S/W int      -
SW_INT22    B    PHANTOM  ; PM 2C  User S/W int      -
SW_INT23    B    PHANTOM  ; PM 2E  User S/W int      -

;===================================================================
; M A I N   C O D E  - starts here
;===================================================================
      .text
START:
      POINT_PG0
      SETC  INTM              ;Disable interrupts
      SPLK  #0h, IMR          ;Mask all Ints
      SPLK  #0FFh, IFR        ;Clear all Int Flags
      CLRC  SXM               ;Clear Sign Extension Mode
      CLRC  OVM               ;Reset Overflow Mode
      CLRC  CNF               ;Config Block B0 to Data mem.
      POINT_B0
      SPLK  #04h, GPR0        ;Set 0 wait states for XMIF
      OUT   GPR0, WSGR
```

```
      POINT_PF1
;Note: comment out if using cystal & PLL option
;     SPLK  #0041h,PLL_CNTL1 ;CLKMD=CLKIN/1, CPUCLK/2=CPUCLK/2

;Note: uncomment if using cystal & PLL option
;This sets the CPU clk to 25MHz & allows 25KHz PWM to be generated
;with full 10 Bit compare resolution.
      SPLK  #00BCh,PLL_CNTL2 ;CLKIN(XTAL)=10MHz, PLL*2.5=20MHz
      SPLK  #0081h,PLL_CNTL1 ;CLKMD=PLL Enable,SYSCLK/4=CPUCLK/2,
      SPLK  #40C0h,SYSCR     ;CLKOUT=CPUCLK

;Comment out if WD is to be active
      SPLK  #006Fh, WD_CNTL  ;Disable WD if VCCP=5V
      KICK_DOG

;---------------------------------------------------------------------
; Initialize Counter, Step parameters, & AR pointers
;---------------------------------------------------------------------
SV_PWM:
      POINT_B0
      SPLK  #STABLE, S_TABLE       ;Used only to save a cycle
      SPLK  #VF_SLOPE, vf_slope    ;Used later for multiply.
      SPLK  #K_prop, Kp            ;Init PI constants
      SPLK  #K_integ, Ki

      LACC  #0h            ;Start at 0 deg.
      SACL  ALPHA          ;Clear ANGLE integrator
      SACL  ENTRY_NEW      ;Clear Sine Table Pointer
      SACL  SECTOR_PTR     ;Init Sector table index pointer

      LACC  #1040          ;Use 41.6 uS period (1040 x 40nS)
      SACL  T              ;Init the PWM period

      LAR   AR1, #CMPR1    ;Init Timer Comp reg pointers
      LAR   AR2, #CMPR2
```

```
      LAR   AR3, #CMPR3

;---------------------------------------------------------------------
;EV Config starts here.
;---------------------------------------------------------------------
EV_CONFIG:
;Configure all I/O pins to I/O function pins
      POINT_PF2
      SPLK  #0h,OPCRA
      SPLK  #0h,OPCRB
EV_LP    SPLK  #0C0Ch,PADATDIR  ;A3,A2=O/P,  A1,A0=I/P, A3,A2=1,1

; Mask all EV interrupts
; (prevent stray PDPINTs from disabling compare outputs)
      POINT_EV                 ;DP => EV Registers
      SPLK  #00000h,IMRA       ;Mask all Group A interrupt flags
      SPLK  #00000h,IMRB       ;Mask all Group B interrupt flags
      SPLK  #00000h,IMRC       ;Mask all Group C interrupt flags

; Clear EV control registers
      SPLK  #00000h,T1CON      ;GP Timer 1 control
      SPLK  #00000h,T2CON      ;GP Timer 2 control
      SPLK  #00000h,T3CON      ;GP Timer 3 control
      SPLK  #00000h,DBTCON     ;Dead band control register
      SPLK  #00000h,COMCON     ;Compare control
      SPLK  #00000h,CAPCON     ;Capture control
      SPLK  #000FFh,CAPFIFO    ;Capture FIFO status bits

;---------------------------------------------------------------------
; Setup GP Timers
;---------------------------------------------------------------------
; Initialize period registers for Timers T1, T2
; T1 used for PWM, T2 used for CAP1
      POINT_B0
      LACC  T
```

```
      POINT_EV
      SACL  T1PER                ;GP Timer 1 period
      SPLK  #07FFFh, T2PER       ;Limit counter values to +ve only

;-----------------------------------------------------------
; Configure GP Timer registers
;-----------------------------------------------------------
;            5432109876543210
;            ||||!!!!||||!!!!
*     SPLK  #1001010101000010b,T2CON  ;Cont Up, /32,
;     SPLK  #1010100001000000b,T1CON  ;Sym
      SPLK  #1001000001000000b,T1CON  ;Asym

;-----------------------------------------------------------
; Configure Capture 1 registers
;-----------------------------------------------------------
;            5432109876543210
;            ||||!!!!||||!!!!
      SPLK  #1010000001000000b,CAPCON
      LACC  FIFO1
      LACC  FIFO1

      POINT_PF2
      SPLK  #0F0h, OPCRB            ;Enable Capture function on pins

;--------------------------------------------------------------------
; Configure Simple & Full Compare registers
;--------------------------------------------------------------------
   ;Start the "Ball rolling" with Timers & Compare units.
      POINT_EV
      SPLK  #0000011001100110b,ACTR     ;Full Action Cntl
      SPLK  #1010001000000111b,COMCON   ;Compare Cntl
      SPLK  #1010001000000111b,COMCON   ;Compare Cntl
;            ||||!!!!||||!!!!
;            5432109876543210
```

```
;-------------------------------------------------------------------
; Enable appropriate Interrupts - EV & DSP core
;-------------------------------------------------------------------
      POINT_EV
      SPLK  #0000001000000000b,IMRA     ;Enable Underflow Int
;            ||||!!!!||||!!!!           ; for Evt Mgr
;            5432109876543210

      SPLK  #0FFFFh,IFRA           ;Clear all Group A interrupt flags
      SPLK  #0FFFFh,IFRB           ;Clear all Group B interrupt flags
      SPLK  #0FFFFh,IFRC           ;Clear all Group C interrupt flags

      POINT_PG0
      SPLK  #0000000000000010b,IMR ;Enable Int lvl 2 for
;            ||||!!!!||||!!!!      ;DSP core  & Emu Int.
;            5432109876543210

      SPLK  #0FFFFh, IFR        ;Clear any pending Ints
      CLRC  INTM                ;Enable global Ints

   ;Init for Capture
      LAR   AR4, #FIFO1         ;Point to Capture FIFO reg
      LAR   AR5, #BC_BUF_STRT   ;Point to start of BC buffer

   ;Clear virtual timer & s/w Flags
      POINT_B0
      LACC  #0h
      SACL  V_TIMER1            ;Reset V_Timer
      SACL  STOP                ;Set to 0 for normal operation
      SACL  GO                  ;Set to 0 for normal operation
      SACL  LOOP_ON_FLG         ;Start with open loop

   ;Init for open loop start-up
      LACC  #STRTUP_FREQ        ;Use open loop freq initially
```

```
      SACL  FREQ_SETPT          ;Init the angular speed
      LACC  #STRTUP_STP_ANGL    ;Use appropriate value for STEP_ANGLE
      SACL  STEP_ANGLE          ;to give ~20Hz also.
      LACC  #STRTUP_INTEG
      SACL  STEP_INTEG_H        ;Init Step integrator

;---------------------------------------------------------------------
-
;Main background loop
;---------------------------------------------------------------------
-
MAIN     POINT_B0
      LACC  V_TIMER1
      SUB   #STARTUP_DELAY
      BCND  MAIN, NEQ
      SPLK  #0Fh, LOOP_ON_FLG   ;On time-out set Flag
      B     MAIN
;---------------------------------------------------------------------



;=====================================================================
; Routine Name: P W M _ I S R             Routine Type: ISR
;
; Description:
;   - Calculate new speed set point
;   - Pass once through PI loop
;   - Load Timer compare regs with previously calculated Ta, Tb, Tc
;   - Calculate new Angle (alpha)
;   - Deduce dx & dy
;   - Determine current Sector Pointer
;   - Do Calculated Branch to Sector Subroutine
;   - Get a speed reading from the Hall sensor using the Capture
;     unit.
;
; Originator: David Figoli
;
```

```
; Last Update:13 MAR 97
;=====================================================================
PWM_ISR:
   ;save regs
      SST   #0, stk1            ;save ST0 - Forced Page 0
      SST   #1, stk2            ;save ST1 - Forced Page 0
      POINT_PG0
      SACL  stk3                ;save ACCL
      SACH  stk4                ;save ACCH

      POINT_EV
      SPLK  #0FFFFh,IFRA        ;Clear all Group A interrupt flags

   ;Increment Virtual Timers
      POINT_B0
      LACC  V_TIMER1
      ADD   #1
      SACL  V_TIMER1

;---------------------------------------------------------------------
; PI controller section
;---------------------------------------------------------------------
;Calculate Speed Setpoint - i.e. the Setpoint for the system.
;This variable is used in the error calculation when closing
;the loop.
      LT FREQ_SETPT             ;SPEED_sp = FREQ_SETPT x 22
      MPY   #22                 ;22 is a scaling factor
      PAC
      SACL  SPEED_sp

   ;The PI controller uses STEP_ANGLE to control the Stator MMF
   ;rotation speed. STEP_ANGLE has a range of
   ; 1(0.06Hz) --> 2048(125Hz)

PI_LOOP  POINT_B0
      SPM   1                   ;Allow shift left of 1
```

```
      LACC  SPEED_sp
      SUB   SPEED_fb              ;Calculate error term E
      SACL  SPD_ERROR             ;Save it

   ;-----------------------------------------------------------------
   ;Check if it's time to close the PI loop. PI loop is closed only
   ;after a start-up delay in which motor starts to spin & the Speed
   ;measurement from the Hall sensor is valid
      LACC  LOOP_ON_FLG           ;Skip PI loop if Flag not
      SUB   #0Fh                  ;set.
      BCND  PROFILE1, NEQ
   ;-----------------------------------------------------------------

PL_1  LT SPD_ERROR
      MPY   Ki
      PAC                         ;ACC = E*Ki
      ADD   STEP_INTEG_L          ;Keep a 32 bit integ value I
      ADDH  STEP_INTEG_H          ;I = I + Ki*E (E=SPD_ERROR)
      SACL  STEP_INTEG_L          ;Store Low word
      SACH  STEP_INTEG_H          ;Store Hi word

      LT SPD_ERROR
      MPY   Kp                    ;
      APAC                        ;ACC  = I + Ki*E + Kp*E
      SACH  GPR0                  ;GPR0 = I + Ki*E + Kp*E

      LT GPR0                     ;GPR0 is in Q15
      MPY   #2048                 ;2048 is in Q0
      PAC                         ;Result is Q15 in 32bit format
      SACH  STEP_ANGLE            ;Store as Q0

   ;Check for Out of range STEP_ANGLE - i.e. 1 < STEP_ANGLE < 2048
      LACC  STEP_ANGLE
      SUB   #2048
      BCND  OVER_RANGE, GT           ;Check if "over-speed"
```

```
      LACC  STEP_ANGLE
      BCND  UNDER_RANGE, LT        ;Check if "under-speed"
      B     PROFILE1               ;If not continue

OVER_RANGE  SPLK  #2048, STEP_ANGLE;Force upper limit
      B  PROFILE1                  ;Continue

UNDER_RANGE SPLK  #1, STEP_ANGLE   ;Force lower limit

;---------------------------------------------------------------------
;Calculate new Voltage V based on Volts/Freq (V/Hz) profile
;---------------------------------------------------------------------
PROFILE1 SPM   0                   ;Set back to default mode
      LACC  STEP_ANGLE
      SUB   #F1                    ;Is Freq<=F1
      BCND  PROFILE2, GT
      LACC  #Vmin
      SACL  V                      ;V is in Q15
      B     NEW_ALPHA

PROFILE2 LACC  STEP_ANGLE
      SUB   #F2
      BCND  PROFILE3, GT

      LACC  STEP_ANGLE,4           ;Convert FCV to Q15 format
      SACL  GPR0
      LT    GPR0
      MPY   vf_slope               ;P = vf_slope * FCV
      PAC                          ;Q13 * Q15 --> Q28
      SACH  V,1                    ;convert result to Q15 format
      LACC  V
      ADD   #INTERCEPT             ;INTERCEPT is in Q13
      SACL  V,2                    ;result V is in Q15
      B     NEW_ALPHA
```

```
PROFILE3 LACC  #Vmax
      SACL  V                       ;V is in Q15

;---------------------------------------------------------------------
;Calculate angle ALPHA for the new position of the Space vector
;& perform decomposition of the Basic unit vectors.
;---------------------------------------------------------------------
NEW_ALPHA   LACC  ENTRY_NEW
      SACL  ENTRY_OLD
      LACC  ALPHA
      ADD   STEP_ANGLE              ;Inc angle.
      SACL  ALPHA                   ;Save
      LACC  ALPHA,8                 ;Prepare pointer for Sine table
      SACH  ENTRY_NEW
      LACC  S_TABLE
      ADD   ENTRY_NEW               ;ACC = actual table pointer value
      TBLR  dy                      ;dy=Sin(ALPHA)

      LT    dy                      ;dy is in Q15
      MPY   V                       ;V  is in Q15
      PAC                           ;P = V * dy
      SACH  dy,1                    ;shift 1 to restore Q15 format

      LACC  dy,11                ;scale for 10 bit integer resolution
      SACH  dy                   ;Save in Q0 format
      LACC  #0FFh                ;ACC=60 deg
      SUB   ENTRY_NEW
      ADD   S_TABLE
      TBLR  dx                   ;dx=Sin(60-ALPHA)

      LT    dx
      MPY   V
      PAC                        ;P = V * dx
      SACH  dx,1                 ;shift 1 to restore Q15 format
```

```
      LACC  dx,11               ;scale for 10 bit integer resolution
      SACH  dx                  ;Save in Q0 format

   ;Determine which Sector Space vector is in
      LACC  ENTRY_NEW
      SUB   ENTRY_OLD
      BCND  BRNCH_SR, GEQ       ;If negative need to change Sector
                                ;If positive continue
MODIFY_SEC_PTR LACC  SECTOR_PTR ;
      SUB   #05h                ;Check if at last sector (S6)
      BCND  PISR1,EQ            ;If yes, re-init AR1= 1st Sector (S1)
      LACC  SECTOR_PTR          ;If no, select next Sector (Sn->Sn+1)
      ADD   #01h
      SACL  SECTOR_PTR          ;i.e. inc SECTOR_PTR
      B     BRNCH_SR
PISR1 SPLK  #00, SECTOR_PTR     ;Reset Sector pointer to 0

BRNCH_SR:
      LACC  #SECTOR_TBL
      ADD   SECTOR_PTR
      TBLR  SR_ADDR
      LACC  SR_ADDR
      BACC

;-------------------------------------------------------------------
;Calculate resultant PWM compare values from dx & dy & perform
;appropriate phase "scrambling" to maintain correct switching
;sequence.
;Note: only one of the following Sector calculations are done per
;ISR call.
;-------------------------------------------------------------------
;-------------------------------------------------------------------
;Sector 1 calculations - a,b,c --> a,b,c
;-------------------------------------------------------------------
SECTOR_SR1:
```

```
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = T-dx-dy
      SFR                       ;Acc = Ta = 1/2(T-dx-dy)  <A>
      SACL  Ta

      ADD   dx                  ;Acc = Tb = dx+Ta   <B>
      SACL  Tb

      LACC  T                   ;ACC = T
      SUB   Ta                  ;ACC = T-Ta
      SACL  Tc                  ;ACC = Tc = T-Ta   <C>
      B     LOAD_COMPARES

;--------------------------------------------------------------------
;Sector 2 calculations - a,b,c --> b,a,c  &  dx <--> dy
;--------------------------------------------------------------------
SECTOR_SR2:
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = T-dx-dy
      SFR                       ;Acc = Tb = 1/2(T-dx-dy)  <A>
      SACL  Tb

      ADD   dy                  ;Acc = Ta = dy+Tb   <B>
      SACL  Ta

      LACC  T                   ;ACC = T
      SUB   Tb                  ;ACC = T-Tb
      SACL  Tc                  ;ACC = Tc = T-Tb   <C>
      B     LOAD_COMPARES

;--------------------------------------------------------------------
;Sector 3 calculations - a,b,c --> c,a,b
;--------------------------------------------------------------------
```

```
SECTOR_SR3:
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = T-dx-dy
      SFR                       ;Acc = Tc = 1/2(T-dx-dy)  <A>
      SACL  Tb

      ADD   dx                  ;Acc = Ta = dx+Tc   <B>
      SACL  Tc

      LACC  T                   ;ACC = T
      SUB   Tb                  ;ACC = T-Tc
      SACL  Ta                  ;ACC = Tb = T-Tc   <C>
      B     LOAD_COMPARES

;---------------------------------------------------------------------
;Sector 4 calculations - a,b,c --> c,b,a  &  dx <--> dy
;---------------------------------------------------------------------
SECTOR_SR4:
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = T-dx-dy
      SFR                       ;Acc = Tc = 1/2(T-dx-dy)  <A>
      SACL  Tc

      ADD   dy                  ;Acc = Tb = dx+Ta   <B>
      SACL  Tb

      LACC  T                   ;ACC = T
      SUB   Tc                  ;ACC = T-Tc
      SACL  Ta                  ;ACC = Ta = T-Tc   <C>
      B     LOAD_COMPARES
```

```
;-----------------------------------------------------------------------
;Sector 5 calculations - a,b,c --> b,c,a
;-----------------------------------------------------------------------
SECTOR_SR5:
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = Tb = 1/2(T-dx-dy)  <A>
      SACL  Tc

      ADD   dx                  ;Acc = Tc = dx+Ta   <B>
      SACL  Ta

      LACC  T                   ;ACC = T
      SUB   Tc                  ;ACC = T-Tb
      SACL  Tb                  ;ACC = Ta = T-Tb   <C>
      B     LOAD_COMPARES

;-----------------------------------------------------------------------
;Sector 6 calculations - a,b,c --> a,c,b  &  dx <--> dy
;-----------------------------------------------------------------------
SECTOR_SR6:
      LACC  T                   ;Acc = T
      SUB   dx                  ;Acc = T-dx
      SUB   dy                  ;Acc = T-dx-dy
      SFR                       ;Acc = Ta = 1/2(T-dx-dy)  <A>
      SACL  Ta

      ADD   dy                  ;Acc = Tc = dx+Ta   <B>
      SACL  Tc

      LACC  T                   ;ACC = T
      SUB   Ta                  ;ACC = T-Ta
      SACL  Tb                  ;ACC = Tb = T-Ta   <C>

   ;Transfer new Compare values for this PWM period
```

```
LOAD_COMPARES POINT_B0
      MAR   *, AR1
      LACC  Ta
      SACL  *,0,AR2             ;Load Compare2 Register with Ta
      LACC  Tb
      SACL  *,0,AR3             ;Load Compare3 Register with Tb
      LACC  Tc
      SACL  *,0,AR1             ;Load Compare4 Register with Tc

;---------------------------------------------------------------------
;Check Push-Button I/Ps for User Frequency (Speed) change.
;---------------------------------------------------------------------
      POINT_PF2
      LACC  PADATDIR
      POINT_B0
      SACL  GPR0
      AND   #0001h
      BCND  B1_DWN, EQ          ;Has PB1 been pressed?
      LACC  GPR0                ;No! - check B2
      AND   #0002h
      BCND  B2_DWN, EQ          ;Has PB2 been pressed?

      SPLK  #0,B1_CNT           ;No! - Clear B1 button counter
      SPLK  #0,B2_CNT           ;No! - Clear B2 button counter
      B     PB_END              ; & return

B1_DWN   LACC  B1_CNT           ;Yes! - Inc B1 button counter
      ADD   #1
      SACL  B1_CNT
      SUB   #BCNT_MAX
      BCND  PB_END, LEQ         ;If max not reached go back
      SPLK  #0,B1_CNT           ;Clear B1 button counter
B1_ACTION   LACC  FREQ_SETPT    ;If max then Inc Frequency
      ADD   #1
      SACL  FREQ_SETPT
```

```
      SUB   #MAX_SPEED_LMT
      BCND  B1_END, LEQ
      SPLK  #MAX_SPEED_LMT, FREQ_SETPT
B1_END   B  PB_END              ;return

B2_DWN   LACC  B2_CNT           ;Yes! - Inc B2 button counter
      ADD   #1
      SACL  B2_CNT
      SUB   #BCNT_MAX
      BCND  PB_END, LEQ         ;If max not reached go back
      SPLK  #0,B2_CNT           ;Clear B2 button counter

B2_ACTION   LACC  FREQ_SETPT    ;If max then Dec Frequency
      SUB   #1
      SACL  FREQ_SETPT
      SUB   #MIN_SPEED_LMT
      BCND  B2_END, GT
      SPLK  #MIN_SPEED_LMT, FREQ_SETPT
B2_END      B  PB_END           ;return

PB_END:

;---------------------------------------------------------------------
;Use Capture to get a "speed" reading from Hall sensor & sprocket
;---------------------------------------------------------------------
      POINT_EV
CHECK_CAP_FLG BIT   IFRC, BIT0
      BCND  FD_END2, NTC        ;If no edge present exit ISR

      POINT_B0
      MAR   *, AR4              ;Point to FIFO1 reg
      LACC  CAP_NEW
      SACL  CAP_OLD
      LACC  *                   ;Load FIFO1
      SACL  CAP_NEW
```

```
      SUB   CAP_OLD
      BCND  NEG_DELTA, LT       ;If negative

POS_DELTA   SACL  CAP_PERIOD    ;Delta = f(t2) - f(t1)
      B     BOX_CAR

NEG_DELTA   ADD #7FFFh          ;Add 1 to Delta
      SACL  CAP_PERIOD          ;Delta = 1 + f(t2) - f(t1)

   ;Perform "Box-Car" average
BOX_CAR  MAR   *, AR6
      LAR   AR6, #BC_BUF_STRT   ;Prepare to sum values
      LACC  #0
      RPT   #(BC_SIZE-1)        ;Sum all Capture values in buffer
      ADD   *+
      SFL
      SFL
      SFL
      SFL
      SACH  BCAVG, 7            ;Scale for max +ve Q15 result.

   ;Update circular buffer pointer
      MAR   *, AR5
      LACC  CAP_PERIOD
      SACL  *+                  ;Replace oldest with newest.
      LAR   AR0, #(BC_SIZE+BC_BUF_STRT)
      CMPR  2                   ;Is AR5 > BC (i.e. box car size) ?
      BCND  CAP_EXIT, NTC       ;If not continue
      LAR   AR5, #BC_BUF_STRT   ;If yes wrap pointer back to start

   ;Calculate the Speed of rotation, i.e. speed = 1/period
      ;Phase 1
      LACC  #07FFFh             ;Load Numerator Hi
      RPT   #15
      SUBC  BCAVG
```

```
      SACL  SPEED_HI
      XOR   SPEED_HI
      OR    #0FFFFh             ;Load Numerator Lo
      ;Phase 2
      RPT   #15
      SUBC  BCAVG
      SACL  SPEED_LO            ;Result is in Q16 in 32 bit fmt

   ;Scale 32 bit value to fit in 16 bits (i.e. Q15)
      LACC  SPEED_LO
      ADDH  SPEED_HI
      SFL
      SACH  SPEED_fb, 7         ;SPEED_fb = SPEED_HI:LO x 256

CAP_EXIT POINT_EV
      LACC  FIFO1               ;Dummy read only
      SPLK  #0FFFFh, IFRC       ;Clear all CAP flags

;---------------------------------------------------
;Exit ISR & Restore regs
;---------------------------------------------------
FD_END2  POINT_PG0
      ZALH      stk4            ; restore ACCH
      ADDS      stk3            ; restore ACCL
      LST   #1, stk2            ; restore ST1
      LST   #0, stk1            ; restore ST0
      CLRC  INTM
      RET
;---------------------------------------------------------------------
;Sector routine jump table - used with BACC inst.
;---------------------------------------------------------------------
SECTOR_TBL:
SR0   .word SECTOR_SR1
SR1   .word SECTOR_SR2
SR2   .word SECTOR_SR3
```

```
SR3   .word SECTOR_SR4
SR4   .word SECTOR_SR5
SR5   .word SECTOR_SR6
;---------------------------------------------------------------------
;No. Samples: 256,  Angle Range: 60, Format: Q15
;---------------------------------------------------------------------
;                 SINVAL; Index    Angle  Sin(Angle)
;---------------------------------------------------------------------
STABLE   .word    0     ;  0        0        0.00
         .word    134   ;  1        0.23     0.00
         .word    268   ;  2        0.47     0.01
         .word    402   ;  3        0.70     0.01
         .word    536   ;  4        0.94     0.02
         .word    670   ;  5        1.17     0.02
         .word    804   ;  6        1.41     0.02
         .word    938   ;  7        1.64     0.03
         .word    1072  ;  8        1.88     0.03
         .word    1206  ;  9        2.11     0.04
         .word    1340  ;  10       2.34     0.04
         .word    1474  ;  11       2.58     0.04
         .word    1608  ;  12       2.81     0.05
         .word    1742  ;  13       3.05     0.05
         .word    1876  ;  14       3.28     0.06
         .word    2009  ;  15       3.52     0.06
         .word    2143  ;  16       3.75     0.07
         .word    2277  ;  17       3.98     0.07
         .word    2411  ;  18       4.22     0.07
         .word    2544  ;  19       4.45     0.08
         .word    2678  ;  20       4.69     0.08
         .word    2811  ;  21       4.92     0.09
         .word    2945  ;  22       5.16     0.09
         .word    3078  ;  23       5.39     0.09
         .word    3212  ;  24       5.63     0.10
         .word    3345  ;  25       5.86     0.10
         .word    3479  ;  26       6.09     0.11
```

```
        .word    3612  ;  27        6.33     0.11
        .word    3745  ;  28        6.56     0.11
        .word    3878  ;  29        6.80     0.12
        .word    4011  ;  30        7.03     0.12
        .word    4144  ;  31        7.27     0.13
        .word    4277  ;  32        7.50     0.13
        .word    4410  ;  33        7.73     0.13
        .word    4543  ;  34        7.97     0.14
        .word    4675  ;  35        8.20     0.14
        .word    4808  ;  36        8.44     0.15
        .word    4941  ;  37        8.67     0.15
        .word    5073  ;  38        8.91     0.15
        .word    5205  ;  39        9.14     0.16
        .word    5338  ;  40        9.38     0.16
        .word    5470  ;  41        9.61     0.17
        .word    5602  ;  42        9.84     0.17
        .word    5734  ;  43        10.08    0.17
        .word    5866  ;  44        10.31    0.18
        .word    5998  ;  45        10.55    0.18
        .word    6130  ;  46        10.78    0.19
        .word    6261  ;  47        11.02    0.19
        .word    6393  ;  48        11.25    0.20
        .word    6524  ;  49        11.48    0.20
        .word    6655  ;  50        11.72    0.20
        .word    6787  ;  51        11.95    0.21
        .word    6918  ;  52        12.19    0.21
        .word    7049  ;  53        12.42    0.22
        .word    7180  ;  54        12.66    0.22
        .word    7310  ;  55        12.89    0.22
        .word    7441  ;  56        13.13    0.23
        .word    7571  ;  57        13.36    0.23
        .word    7702  ;  58        13.59    0.24
        .word    7832  ;  59        13.83    0.24
        .word    7962  ;  60        14.06    0.24
        .word    8092  ;  61        14.30    0.25
```

```
.word    8222  ;  62       14.53    0.25
.word    8351  ;  63       14.77    0.25
.word    8481  ;  64       15.00    0.26
.word    8610  ;  65       15.23    0.26
.word    8740  ;  66       15.47    0.27
.word    8869  ;  67       15.70    0.27
.word    8998  ;  68       15.94    0.27
.word    9127  ;  69       16.17    0.28
.word    9255  ;  70       16.41    0.28
.word    9384  ;  71       16.64    0.29
.word    9512  ;  72       16.88    0.29
.word    9640  ;  73       17.11    0.29
.word    9768  ;  74       17.34    0.30
.word    9896  ;  75       17.58    0.30
.word    10024 ;  76       17.81    0.31
.word    10151 ;  77       18.05    0.31
.word    10279 ;  78       18.28    0.31
.word    10406 ;  79       18.52    0.32
.word    10533 ;  80       18.75    0.32
.word    10660 ;  81       18.98    0.33
.word    10786 ;  82       19.22    0.33
.word    10913 ;  83       19.45    0.33
.word    11039 ;  84       19.69    0.34
.word    11165 ;  85       19.92    0.34
.word    11291 ;  86       20.16    0.34
.word    11417 ;  87       20.39    0.35
.word    11668 ;  89       20.86    0.36
.word    11793 ;  90       21.09    0.36
.word    11918 ;  91       21.33    0.36
.word    12043 ;  92       21.56    0.37
.word    12167 ;  93       21.80    0.37
.word    12292 ;  94       22.03    0.38
.word    12416 ;  95       22.27    0.38
.word    12540 ;  96       22.50    0.38
.word    12664 ;  97       22.73    0.39
```

```
.word    12787 ;  98      22.97   0.39
.word    12910 ;  99      23.20   0.39
.word    13033 ;  100     23.44   0.40
.word    13156 ;  101     23.67   0.40
.word    13279 ;  102     23.91   0.41
.word    13401 ;  103     24.14   0.41
.word    13524 ;  104     24.38   0.41
.word    13646 ;  105     24.61   0.42
.word    13767 ;  106     24.84   0.42
.word    13889 ;  107     25.08   0.42
.word    14010 ;  108     25.31   0.43
.word    14131 ;  109     25.55   0.43
.word    14252 ;  110     25.78   0.43
.word    14373 ;  111     26.02   0.44
.word    14493 ;  112     26.25   0.44
.word    14613 ;  113     26.48   0.45
.word    14733 ;  114     26.72   0.45
.word    14852 ;  115     26.95   0.45
.word    14972 ;  116     27.19   0.46
.word    15091 ;  117     27.42   0.46
.word    15210 ;  118     27.66   0.46
.word    15328 ;  119     27.89   0.47
.word    15447 ;  120     28.13   0.47
.word    15565 ;  121     28.36   0.48
.word    15683 ;  122     28.59   0.48
.word    15800 ;  123     28.83   0.48
.word    15917 ;  124     29.06   0.49
.word    16035 ;  125     29.30   0.49
.word    16151 ;  126     29.53   0.49
.word    16268 ;  127     29.77   0.50
.word    16384 ;  128     30.00   0.50
.word    16500 ;  129     30.23   0.50
.word    16616 ;  130     30.47   0.51
.word    16731 ;  131     30.70   0.51
.word    16846 ;  132     30.94   0.51
```

```
.word   16961 ;  133      31.17    0.52
.word   17075 ;  134      31.41    0.52
.word   17190 ;  135      31.64    0.52
.word   17304 ;  136      31.88    0.53
.word   17417 ;  137      32.11    0.53
.word   17531 ;  138      32.34    0.53
.word   17644 ;  139      32.58    0.54
.word   17757 ;  140      32.81    0.54
.word   17869 ;  141      33.05    0.55
.word   17981 ;  142      33.28    0.55
.word   18093 ;  143      33.52    0.55
.word   18205 ;  144      33.75    0.56
.word   18316 ;  145      33.98    0.56
.word   18427 ;  146      34.22    0.56
.word   18538 ;  147      34.45    0.57
.word   18648 ;  148      34.69    0.57
.word   18758 ;  149      34.92    0.57
.word   18868 ;  150      35.16    0.58
.word   18978 ;  151      35.39    0.58
.word   19087 ;  152      35.63    0.58
.word   19195 ;  153      35.86    0.59
.word   19304 ;  154      36.09    0.59
.word   19412 ;  155      36.33    0.59
.word   19520 ;  156      36.56    0.60
.word   19627 ;  157      36.80    0.60
.word   19735 ;  158      37.03    0.60
.word   19841 ;  159      37.27    0.61
.word   19948 ;  160      37.50    0.61
.word   20054 ;  161      37.73    0.61
.word   20160 ;  162      37.97    0.62
.word   20265 ;  163      38.20    0.62
.word   20371 ;  164      38.44    0.62
.word   20475 ;  165      38.67    0.62
.word   20580 ;  166      38.91    0.63
.word   20684 ;  167      39.14    0.63
```

```
.word   20788 ;  168     39.38   0.63
.word   20891 ;  169     39.61   0.64
.word   20994 ;  170     39.84   0.64
.word   21097 ;  171     40.08   0.64
.word   21199 ;  172     40.31   0.65
.word   21301 ;  173     40.55   0.65
.word   21403 ;  174     40.78   0.65
.word   21504 ;  175     41.02   0.66
.word   21605 ;  176     41.25   0.66
.word   21706 ;  177     41.48   0.66
.word   21806 ;  178     41.72   0.67
.word   21906 ;  179     41.95   0.67
.word   22006 ;  180     42.19   0.67
.word   22105 ;  181     42.42   0.67
.word   22204 ;  182     42.66   0.68
.word   22302 ;  183     42.89   0.68
.word   22400 ;  184     43.13   0.68
.word   22498 ;  185     43.36   0.69
.word   22595 ;  186     43.59   0.69
.word   22692 ;  187     43.83   0.69
.word   22788 ;  188     44.06   0.70
.word   22884 ;  189     44.30   0.70
.word   22980 ;  190     44.53   0.70
.word   23075 ;  191     44.77   0.70
.word   23170 ;  192     45.00   0.71
.word   23265 ;  193     45.23   0.71
.word   23359 ;  194     45.47   0.71
.word   23453 ;  195     45.70   0.72
.word   23546 ;  196     45.94   0.72
.word   23640 ;  197     46.17   0.72
.word   23732 ;  198     46.41   0.72
.word   23824 ;  199     46.64   0.73
.word   23916 ;  200     46.88   0.73
.word   24008 ;  201     47.11   0.73
.word   24099 ;  202     47.34   0.74
```

```
.word   24189 ;  203     47.58   0.74
.word   24279 ;  204     47.81   0.74
.word   24369 ;  205     48.05   0.74
.word   24459 ;  206     48.28   0.75
.word   24548 ;  207     48.52   0.75
.word   24636 ;  208     48.75   0.75
.word   24724 ;  209     48.98   0.75
.word   24812 ;  210     49.22   0.76
.word   24900 ;  211     49.45   0.76
.word   24986 ;  212     49.69   0.76
.word   25073 ;  213     49.92   0.77
.word   25159 ;  214     50.16   0.77
.word   25245 ;  215     50.39   0.77
.word   25330 ;  216     50.63   0.77
.word   25415 ;  217     50.86   0.78
.word   25499 ;  218     51.09   0.78
.word   25583 ;  219     51.33   0.78
.word   25667 ;  220     51.56   0.78
.word   25750 ;  221     51.80   0.79
.word   25833 ;  222     52.03   0.79
.word   25915 ;  223     52.27   0.79
.word   25997 ;  224     52.50   0.79
.word   26078 ;  225     52.73   0.80
.word   26159 ;  226     52.97   0.80
.word   26239 ;  227     53.20   0.80
.word   26320 ;  228     53.44   0.80
.word   26399 ;  229     53.67   0.81
.word   26478 ;  230     53.91   0.81
.word   26557 ;  231     54.14   0.81
.word   26635 ;  232     54.38   0.81
.word   26713 ;  233     54.61   0.82
.word   26791 ;  234     54.84   0.82
.word   26868 ;  235     55.08   0.82
.word   26944 ;  236     55.31   0.82
.word   27020 ;  237     55.55   0.82
```

```
        .word   27096 ;  238     5.78    0.83
        .word   27171 ;  239     56.02   0.83
        .word   27246 ;  240     56.25   0.83
        .word   27320 ;  241     56.48   0.83
        .word   27394 ;  242     56.72   0.84
        .word   27467 ;  243     56.95   0.84
        .word   27540 ;  244     57.19   0.84
        .word   27612 ;  245     57.42   0.84
        .word   27684 ;  246     57.66   0.84
        .word   27756 ;  247     57.89   0.85
        .word   27827 ;  248     58.13   0.85
        .word   27897 ;  249     58.36   0.85
        .word   27967 ;  250     58.59   0.85
        .word   28037 ;  251     58.83   0.86
        .word   28106 ;  252     59.30   0.86
        .word   28243 ;  254     59.53   0.86
        .word   28311 ;  255     59.77   0.86
;==================================================
; I S R  -  PHANTOM
;
; Description: Dummy ISR, used to trap spurious interrupts.
;
; Modifies:
;
;==================================================
PHANTOM  B  PHANTOM
```