ADSP-21990: Reference Frame Conversions

AN21990-11

Table of Contents

Summary

The introduction of reference frames in the analysis of electrical machines has turned out not only to be useful in their analysis but also has provided a powerful tool for the implementation of sophisticated control techniques. This application note gives an introduction to the theory of the most commonly used reference frames and provides routines that allow for easy conversion amongst them. They are implemented in a library-like module for immediate and intuitive application.

1 Introduction to reference frame theory

1.1 Overview

As the application of ac machines has continued to increase over this century, new techniques have been developed to aid in their analysis. Much of the analysis has been carried out for the treatment of the well-known induction machine. The significant breakthrough in the analysis of three-phase ac machines was the development of reference frame theory. Using these techniques, it is possible to transform the phase variable machine description to another reference frame. By judicious choice of the reference frame, it proves possible to simplify considerably the complexity of the mathematical machine model. While these techniques were initially developed for the analysis and simulation of ac machines, they are now invaluable tools in the digital control of such machines. As digital control techniques are extended to the control of the currents, torque and flux of such machines, the need for compact, accurate machine models is obvious.

Fortunately, the developed theory of reference frames is equally applicable to the synchronous machines, such as the Permanent Magnet Synchronous Machine (PMSM). This machine is sometimes known as the sinusoidal brushless machines or the brushless ac machine and is very popular as a high-performance servo drive due to its superior torque-to-weight ratio and its high dynamic capability. It is a three-phase synchronous ac machine with permanent-magnet rotor excitation and is designed to have a sinusoidal torque-position characteristic.

The aim of this section is to introduce the essential concepts of reference frame theory and to introduce the space vector notation that is used to write compact mathematical descriptions of ac machines. Over the years, many different reference frames have been proposed for the analysis of ac machines. The most commonly used ones are the so-called stationary reference frame and the rotor reference frame.

1.2 Three-phase to two-phase transformation (Clarke transformation)

Three-phase ac machines are conventionally modeled using phase variable notation. However, for a three-phase, star-connected machine, the phase quantities are not independent variables so that:

$$\begin{aligned} i_{sA}(t) + i_{sB}(t) + i_{sC}(t) &= 0 \\ v_{sA}(t) + v_{sB}(t) + v_{sC}(t) &= 0 \\ \varphi_{sA}(t) + \varphi_{sB}(t) + \varphi_{sC}(t) &= 0 \end{aligned} \quad (1)$$

where i_S , v_S and φ_S denote stator phase currents, voltages and flux linkages, respectively.

As a result of this redundancy in the phase variable representation it is possible to transform the system to an equivalent two-phase representation. The transformation from three-phase to two-phase quantities is written in matrix form as:

$$\begin{bmatrix} i_{s\alpha}(t) \\ i_{s\beta}(t) \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & \cos(\gamma) & \cos(2\gamma) \\ 0 & \sin(\gamma) & \sin(2\gamma) \end{bmatrix}\begin{bmatrix} i_{sA}(t) \\ i_{sB}(t) \\ i_{sC}(t) \end{bmatrix} \quad (2)$$

where $\gamma = 2\pi/3$. The transformation is equally valid for the voltages and flux linkages. The **stator current space vector** is defined as the complex quantity:

$$\underline{\mathbf{i}}_s(t) = i_{s\alpha}(t) + j i_{s\beta}(t) \quad (3)$$

and it is possible to write (2) more compactly as:

$$\underline{\mathbf{i}}_s(t) = \frac{2}{3}\left[i_{sA}(t) + \underline{\mathbf{a}}\, i_{sB}(t) + \underline{\mathbf{a}}^2\, i_{sB}(t)\right] \quad (4)$$

where $\underline{\mathbf{a}}$ is a vector operator that produces a vector rotation of $\gamma = 2\pi/3$ and is defined as:

$$\underline{\mathbf{a}} = \exp(j\gamma) = \cos(\gamma) + j\sin(\gamma) \quad (5)$$

The choice of the constant in the transformations of (2) and (4) is somewhat arbitrary. Here, the value of 2/3 is adopted. Its main advantage is that magnitudes are preserved across the transformation. Therefore, sinusoidal phase currents with a peak magnitude of I_m produce a current space vector with a peak magnitude of I_m. For this transformation, the inverse relationship is written:

$$\begin{bmatrix} i_{sA}(t) \\ i_{sB}(t) \\ i_{sC}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \cos(\gamma) & \sin(\gamma) \\ \cos(2\gamma) & \sin(2\gamma) \end{bmatrix}\begin{bmatrix} i_{s\alpha}(t) \\ i_{s\beta}(t) \end{bmatrix} \quad (6)$$

As the aim of any digital current control scheme is to control current amplitude, it is felt that this form is more suitable for control purposes. Transformation (2) and its counterpart (6) are denoted hereafter as the **Forward Clarke Transformation** and the **Reverse Clarke Transformation**, respectively.

The space vector may be viewed in the complex plane as shown in Figure 1. The conventional magnetic axes of the three machine phases are separated by $\gamma = 2\pi/3$. The real or $s\alpha$ axis of the new two-axis co-ordinate system is arbitrarily chosen to coincide with the sA axis. Obviously, the imaginary or $js\beta$ axis lies in quadrature with the $s\alpha$ axis. The current space vector is shown at an arbitrary location in the complex plane. The phase currents may be obtained by projecting the current space vector onto the respective phase axis.

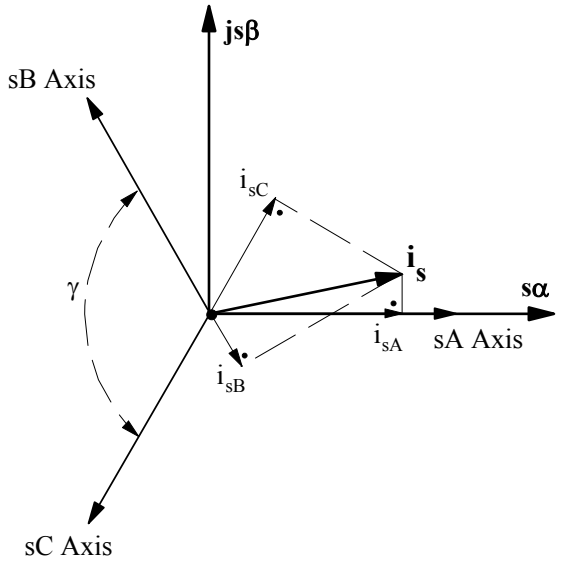

Figure 1: Relationship of stator current space vector and stator phase currents.

Consider the case of a balanced set of three-phase stator currents:

$$\begin{aligned} i_{sA} &= I_m \sin(\theta - \varphi) \\ i_{sB} &= I_m \sin(\theta - \gamma - \varphi) \\ i_{sC} &= I_m \sin(\theta - 2\gamma - \varphi) \end{aligned} \tag{7}$$

where I_m is the magnitude of the phase currents, $\theta = \omega t$ is the angular position in radians and φ is the phase angle. Using (2) the currents of (7) may be transformed to the equivalent two-phase representation to give:

$$\begin{aligned} i_{s\alpha} &= I_m \sin(\theta - \varphi) \\ i_{s\beta} &= -I_m \cos(\theta - \varphi) \end{aligned} \tag{8}$$

so that the current space vector of such a system may be written as:

$$\begin{aligned} \underline{\mathbf{i}}_s &= I_m \sin(\theta - \varphi) - jI_m \cos(\theta - \varphi) \\ &= -jI_m e^{j(\theta - \varphi)} \end{aligned} \tag{9}$$

which describes a circular trajectory in the space vector plane.

Therefore, a balanced three-phase system in phase variables transforms to a circular locus in the equivalent two-axis representation. The radius of the circle is the peak magnitude of the phase quantities. The circular locus is described at a rate equal to the angular frequency of the phase quantities.

1.3 Vector rotation (Park transformation)

The stator current, voltage and flux linkage space vectors are complex quantities defined in a reference frame whose real axis is fixed to the magnetic axis of stator winding sA. However, the corresponding quantities defined for the rotor circuit of a three-phase ac machine are similarly stated in a reference frame fixed to the rotor. In the analysis of electrical machines, it is generally necessary to adopt a common reference frame for both the rotor and the stator. For this reason, a second transformation, known as a **vector rotation**, is formulated that rotates space vector quantities through a known angle.

In the space vector diagram of Figure 1, the axes of the space vector plane are stationary. Meanwhile the space vectors of the current, voltage and flux linkages rotate about these axes at a rate equal to the angular frequencies of the corresponding phase quantities. If instead a new reference frame is defined where the axes are made to rotate at the same rate as angular frequency of the phase quantities, stationary current, voltage and flux linkage space vector result.

Consider applying the vector rotation through an angle θ:

$$\underline{\nu} = e^{-j\theta} \tag{10}$$

to the current space vector of (3). The current space vector in this new reference frame is given by:

$$\underline{\mathbf{i}}_{dq} = i_{sd} + ji_{sq} = \underline{\mathbf{i}}_s e^{-j\theta} \tag{11}$$

which may also be written in matrix form as:

$$\begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} \tag{12}$$

The real component of the current space vector in this new reference frame is the **direct axis component** while the imaginary component is called the **quadrature axis component.**

The relationship of the real and imaginary components of the current space vector in the original stationary two-axis reference frame and the new rotating reference frame is shown in Figure 2. Clearly, from the viewpoint of the stationary (αβ) frame, both the current space vector and the direct and quadrature axes are rotating at a speed ω. However, when viewed from the rotating reference frame, the current space vector is stationary. The real axis of the rotating reference frame is located at an angle θ from the real axis of the stationary reference frame. The elimination of position dependency from the machine electrical variables is the main advantage of a vector rotation. This transformation will be referred to as the **Reverse Park Transformation**.

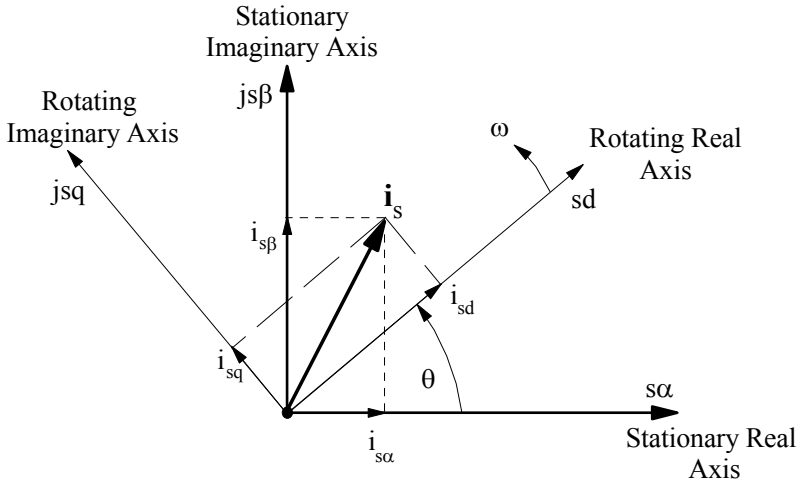

Figure 2: Relationship of current space vector components in stationary and rotating reference frames.

The inverse vector rotation, to transform from a rotating to a stationary reference frame, may be written:

$$\underline{\mathbf{i}}_s = \underline{\mathbf{i}}_{dq} e^{j\theta} \tag{13}$$

or in matrix form as:

$$\begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} \tag{14}$$

This rotation is commonly called the **Forward Park Transformation**.

Consider application of the vector rotation of (10) to the current space vector of (9) derived for the balanced set of stator currents:

$$\underline{\mathbf{i}}_{dq} = -jI_m e^{-j\varphi} \tag{15}$$

so that:

$$\begin{aligned} i_{sd} &= -I_m \cos(\varphi) \\ i_{sq} &= -I_m \sin(\varphi) \end{aligned} \tag{16}$$

which are independent of the instantaneous angular position of the phase quantities.

Therefore, a balanced three-phase system in phase variables may be transformed to an equivalent two-axis representation that is independent of the angular position by applying a three-phase to two-phase transformation followed by a vector rotation by the angular position of the phase quantities.

1.4 Practical implementation of the transformations

In this section, some considerations are made about the overall computational efficiency of the transformations.

- The Forward Clarke Transformation given by (2) may be reduced into a computationally more efficient form by explicitly substituting the (constant) cosine and sine values and applying the first equation of (1). It is easily seen that these actions lead to:

$$\begin{bmatrix} i_{s\alpha}(t) \\ i_{s\beta}(t) \end{bmatrix} = \begin{bmatrix} i_{sA}(t) \\ \frac{1}{\sqrt{3}}\left(i_{sB}(t) - i_{sC}(t)\right) \end{bmatrix} \tag{17}$$

- The Reverse Clarke Transformation given by (6) is implemented by explicitly substituting the (constant) cosine and sine values and applying the first equation of (1):

$$\begin{bmatrix} i_{sA}(t) \\ i_{sB}(t) \\ i_{sC}(t) \end{bmatrix} = \begin{bmatrix} i_{s\alpha}(t) \\ -\frac{1}{2} i_{s\alpha}(t) + \frac{\sqrt{3}}{2} i_{s\beta}(t) \\ -\frac{1}{2} i_{s\alpha}(t) - \frac{\sqrt{3}}{2} i_{s\beta}(t) \end{bmatrix} \tag{18}$$

- Unlike the Clarke transformations, both Park transformations require the real-time computation of the transforming matrix itself. This entails the evaluation of one sine and one cosine function for each. However, the forward and reverse matrices are mutually transposed. A more efficient usage of these transformation is therefore to calculate the sine and cosine function once and to build the two matrices by reversing the sign of the sine where appropriate. This will be explained more in detail in the next section.

- At last, it may be noted that the two transformation steps of (2) and (12) can be combined into a single transformation that may be written as:

$$\begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta-\gamma) & \cos(\theta-2\gamma) \\ -\sin(\theta) & -\sin(\theta-\gamma) & -\sin(\theta-2\gamma) \end{bmatrix}\begin{bmatrix} i_{sA} \\ i_{sB} \\ i_{sC} \end{bmatrix} \quad (19)$$

  This transformation converts the phase quantities to the rotating reference frame in a single step. However, it is often more desirable and computationally more efficient to apply the transformation in two steps as described previously. In this way it is necessary to compute only two sinusoidal functions of position unlike the six required for (19).

2 The Reference Frame Conversion routines

2.1 Using the conversion routines

The routines are developed as an easy-to-use library, which has to be linked to the user's application. The library consists of two files. The file "refframe.dsp" contains the assembly code for the subroutines. The user simply has to include the header file "refframe.h", which provides function-like calls to the routines. The example file in the following section will demonstrate the usage of all the conversion routines.

The following table reassumes the set of macros defined in this library.

Table 1 Implemented routines

Transformation	***Operation***	***Usage***
	Initialisation	refframe_Set_DAG_registers_for_transformations;
Clarke	Forward Clarke	refframe_Forward_Clarke(V_{abc}, $V_{\alpha\beta}$);
	Reverse Clarke	refframe_Reverse _Clarke($V_{\alpha\beta}$, V_{abc});
Park – Method 1	Calculate Sine & Cosine	refframe_Calc_SinCos(φ, V_{sincos});
	Reverse Park	refframe_Reverse_Park_SinCos($V_{\alpha\beta}$, V_{dq}, V_{sincos});
	Forward Park	refframe_Forward_Park_SinCos (V_{dq}, $V_{\alpha\beta}$, V_{sincos});
Park – Method 2	Reverse Park	refframe_Reverse_Park_angle($V_{\alpha\beta}$, V_{dq}, φ);
	Forward Park	refframe_Forward_Park_angle (V_{dq}, $V_{\alpha\beta}$, φ);

where V_{abc} is a three-element vector containing the three phase quantities, $V_{\alpha\beta}$ consists of two elements holding the components of the stator quantity vector, V_{dq} represents the two components of the rotor

quantities and φ is the angle of rotation. V_{SinCos} holds the value for sin(φ) and cos(φ) for temporary storage. The following example will clarify its usage, while section 2.2 describes the format of the quantities.

As already mentioned in section 1.4, there are two ways of implementation for the Park transformation. The three routines listed for "Method 1" in Table 1 should be used whenever both the forward and reverse transformation are required since the trigonometric functions are evaluated only once. This is computationally more efficient at the expense of two additional memory locations for V_{SinCos}. Method 2 calculates the sine and cosine functions every time the Park routine is called, but requires no memory locations for them. This is recommended whenever only one transformation is required.

2.2 Formats of inputs and outputs

The implementation of the routines is such that values for angles are expected to be in the usual scaled 1.15 format. Therefore, +1 (0x7FFF) corresponds to +π radians or 180 degrees, and –1 (0x8000) to -π radians or -180 degrees. This applies to the functions refframe_Calc_SinCos(…), refframe_Forward_Park_angle(…) and refframe_Reverse_Park_angle(…). The other inputs are vectors. That means that the values passed to the routines are addresses of contiguous memory locations. Typically, they are represented by labels, such as Valphabeta. The components of the vectors are numbers in the scaled 1.15 format. However, having the single components in this format does not necessarily entail that the input is in this format. As a simple example, having Vα and Vβ both equal to 1 leads to a vector in the stationary frame with magnitude $\sqrt{2} > 1$. Obviously, if this vector is the input to the inverse Clarke transformation, it corresponds to demand a three-phase system of amplitude bigger than one. It is up to the user to ensure that the inputs are meaningful and adequately scaled.

2.3 Usage of the DSP registers

The macros listed in Table 1 are based on three subroutines that are reported in the following table. Also, an overview is given on the usage of the core DSP registers. It may be noted that the DAG registers M0, M1, M2 must be set to 1 and L0, L1, L2 must be set 0 and that they are not modified by the conversion routines. Furthermore, the trigonometric routines require that M5 and L5 be set to 1 and 0 respectively. The already mentioned call to **refframe_Set_DAG_registers_for_transformations** prepares these registers for the conversion routines. It now becomes clear that this routine is necessary only once if none of the above-mentioned registers are modified in another part of the user's code. However, for the sake of efficiency, it may be more appropriate to use single instructions to restore eventually modified registers rather than a call to this macro that requires 8 instructions.

Table 2 Usage of DSP core registers for the subroutines

Subroutine	*Input*	*Output* [1]	*Modified registers*	*Other registers*
refframe_Forward_Clark	I1 = Input I0 = Output	N/A	MX0, MY0, MR, AR, I0, I1	M0, M1 = 1, L0, L1 = 0
refframe_Reverse_Clark	I1 = Input I2 = Output	N/A	MX0, MY0, MX1, MY1, MR, I1, I2	M1, M2 = 1, L1, L2 = 0
refframe_Rotate_Vector	I0 = Input I1 = Output my0 = cos(angle)	N/A	MX0, MX1, MR, I0, I1 I0, I1	M0, M1= 1, L0, L1 0

[1] The output values are stored in the output vector in Data Memory. No DSP core register is used.

	my1 = sin(angle)			

When using the function like macros, the modified registers are those reported below. The major difference to Table 2 is seen in the Park routines. This is because the routine refframe_Rotate_Vector only performs a vector rotation with the transforming matrix as input, whereas the overall Park transformation requires the computation of a sine and a cosine function.

Table 3 Usage of DSP core registers for the Macros

Usage	*Modified registers*
refframe_Set_DAG_registers_for_transformations;	M0, M1, M2, M5 = 1, L0, L1, L2, L5 = 0
refframe_Forward_Clarke(V_{abc}, $V_{\alpha\beta}$);	MX0, MY0, MR, AR, I0, I1
refframe_Reverse _Clarke($V_{\alpha\beta}$, V_{abc});	MX0, MY0, MX1, MY1, MR, I1, I2
refframe_Calc_SinCos(φ, V_{sincos});	AX0, AY0, AX1, AY1, AR, AF, MY0, MX1, MY1, MR, SR, I0, I6, M6, L6
refframe_Reverse_Park_SinCos($V_{\alpha\beta}$, V_{dq}, V_{sincos});	AX0, AR, MX0, MX1, MY0, MY1, MR, I0, I1, I2
refframe_Forward_Park_SinCos (V_{dq}, $V_{\alpha\beta}$, V_{sincos});	MX0, MX1, MY0, MY1, MR, I0, I1, I2
refframe_Reverse_Park_angle($V_{\alpha\beta}$, V_{dq}, φ);	AX0, AY0, AX1, AY1, AR, AF, MX0, MX1, MY0, MY1, MR, SR, I0, I1, I6, M6, L6
refframe_Forward_Park_angle (V_{dq}, $V_{\alpha\beta}$, φ);	AX0, AY0, AX1, AY1, AR, AF, MX0, MX1, MY0, MY1, MR, SR, I0, I1, I6, M6, L6

2.4 The program code: refframe.dsp

The code contained in the file "refframe.dsp" defines the three routines that are listed in Table 2. The routine **refframe_Rotate_Vector** performs a vector rotation and is the core routine for both the reverse and forward Park transformation.

In the code, equation (17) of the forward Clarke transformation is implemented. Note that the order in which the single operations are executed is changed. The reason for it is that the difference Vb-Vc in a three phase system may assume values that are bigger than one ($V_b - V_c \le \sqrt{3}$) and therefore may not be represented in 1.15 format. This is avoided by pre-multiplying both components with the constant before subtracting them. Also, before subtracting the two values, the saturation mode of the ALU has to be set in order to avoid overflow from positive to negative values and vice-versa. However, this is only necessary for input amplitude equal to one. The two instructions for enabling and disabling the saturation mode could be removed if the input system has magnitude less than one.

The reverse Clarke transformation of equation (18) is also implemented. Similarly to the case of the forward transformation, overflows of the MAC need to be avoided, for input magnitudes equal to one. Again, the corresponding **SAT MR** instructions could be removed if this never occurs.

As already mentioned the refframe_Rotate_Vector routine is used by both the forward and reverse Park transformations. It simply multiplies the input vector by the matrix of equation (14) whose elements are provided as inputs as well. Refer to the description of the header file in the next section for more details on the evaluation of this matrix. Once again, the same considerations as for the reverse Clarke

transformation regarding saturation apply. For more information regarding the program code, please refer to the comments in the "refframe.dsp" file.

2.5 Access to the library: the header file: refframe.h

The library may be accessed by including the header file "refframe.h" in the application code, as was already explained in the section 2.1. The header file is intended to provide function-like calls to the routines presented in the previous section. It defines the calls shown in Table 1. The file is mostly self-explaining. This file, after including the trigonometric library and the core routines, defines the initialisation routine **Set_DAG_registers_for_transformations** and the macros for the forward and reverse Clarke transformations. The only comment is that, since the input and output vectors are pointed to by index registers of the DM-DAG, they must be defined in data memory.

It is worth adding a few comments about the evaluation of the rotation matrix in the different forms of the Park Transformation. The sine and cosine values are calculated and the vector rotation routine is called. For the reverse transformation, the only action that is required is to complement the value of the angle. Note that, since the negate instruction ar = -angle, produces the complement except for an input of -1 (0x8000). If this occurs, the twos-complement is 0x8000 with the overflow flag set to one. The conditional instruction immediately after it detects this case and loads AR with the correct value of +1 (0x7FFF).

Method 1 of the Park transformation is implemented by the refframe_Calc_SinCos, refframe_Forward_Park_SinCos and the refframe_Reverse_Park_SinCos macros. The difference is that the matrix for the reverse transformation is not obtained by recalculating the trigonometric functions with the complemented angle, but by making use of the know symmetries of these functions. Since sin(-x)=-sin(x) and cos(-x)=cos(x), the resulting matrix is obtained by inverting the sign of only the (previously stored) sine value. Again, with a conditional instruction, the special case of –1 is correctly handled.

For more information regarding the header file, please refer to the comments in the "refframe.h" file.

3 Software Example: Testing the Conversion Routines

3.1 The main program: main.dsp

The example demonstrates how to use the routines. All it does is to generate a three-phase sine wave system. The phase components are then transformed into the stationary ($\alpha\beta$-) and rotor (dq) frame. After that, the inverse transformations are executed. The application has been adapted from a previous note[2], hence this section will only explain the few and intuitive modifications to that application.

The file "main.dsp" contains the initialisation and PWM Sync and Trip interrupt service routines. First, the general systems constants and PWM configuration constants (main.h- see the next section) are included. Also included are the PWM library, trigonometric library and the reference conversion library. Next, all the vectors for the transformation routines are declared. The initialisation of PWM block is executed and the interrupts are enabled. The program then enters a loop, which just waits for interrupts.

The interrupt service routine simply shows how to make use of the reference frame conversion routines. After implementing the sine-wave generation (see the application note mentioned above for details) the three "phase voltages" are stored into a vector (Vabc) for use with the transformation routines. Then Vabc is converted into the stationary reference frame (Valphabeta). Method 1 is then executed to transform Valphabeta into the dq-frame (Vdq) and back into Valphabeta_rev by calculating the sine and cosine of the angle once and invoking respectively the reverse and forward Park transformation. Method 2 is also

[2] AN21990-3: Generation of Three-Phase Sine-Wave PWM

shown, although the related code is commented. The user may play with them by simply removing the comment directive "//". At last, the stationary vector is transformed back into the original 3-phase description Vabc_rev. Section 3.3 will show the outputs that are produced by this example. For more information on the main program, please refer to comments in the "main.dsp" file.

3.2 Main.h

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

3.3 Example output

The graphical representation of the vector variables Vabc, Valphabeta, Vdq are shown in the following figures. The original three-phase system is described as follows:

$$\begin{bmatrix} V_A \\ V_B \\ V_C \end{bmatrix} = V \cdot \begin{bmatrix} \sin(\omega t) \\ \sin(\omega t - \frac{2\pi}{3}) \\ \sin(\omega t - \frac{4\pi}{3}) \end{bmatrix} \qquad (20)$$

where ω and V denote the angular speed and the (scaled) amplitude. Note that it is in the form of equation (7) with φ equal to zero. Figure 3 shows the produced waveforms versus the angle (-π to +π).

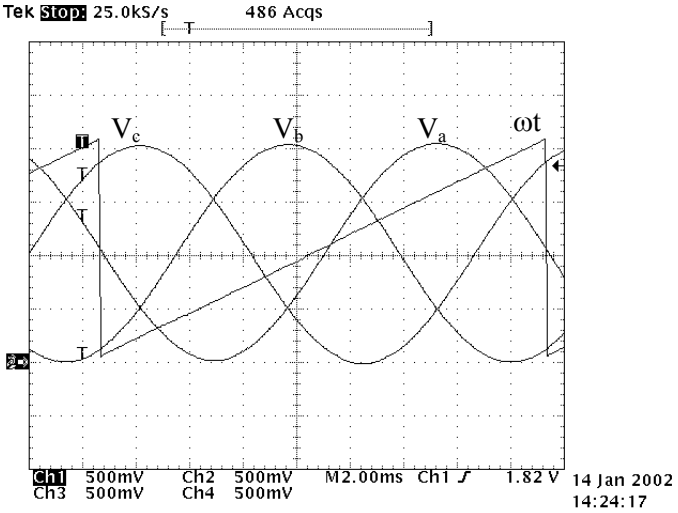

Figure 3 Vector variable Vabc (the saw-tooth is the angle representation ranging from -π to +π)

Applying the forward Clarke transformation (17) to this particular system leads to:

$$\begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix} = V \cdot \begin{bmatrix} \sin(\omega t) \\ -\cos(\omega t) \end{bmatrix} \qquad (21)$$

describing a circular locus in the αβ-plane. This is shown in the next figure.

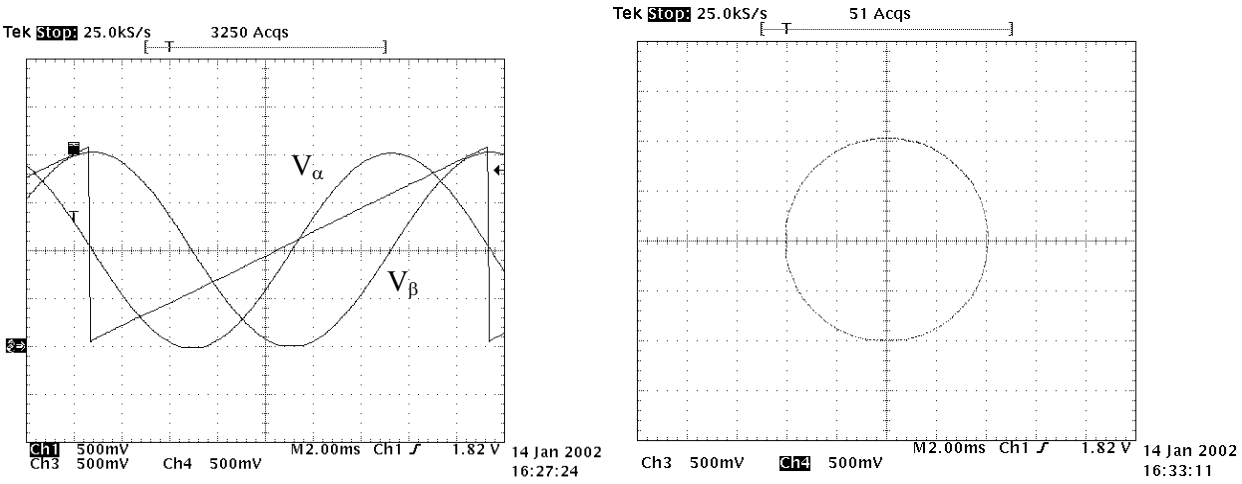

Figure 4 Vector variable Valphabeta versus theta (left) and in the αβ-plane (right)

The next step consists of applying the reverse Park transformation of (12) to (21). It is easily seen that the resulting dq-vector is constant:

$$\begin{bmatrix} V_d \\ V_q \end{bmatrix} = V \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad (22)$$

Figure 5 represents this vector graphically in the time domain. Note that in the dq-plane the vector is reduced to one point.

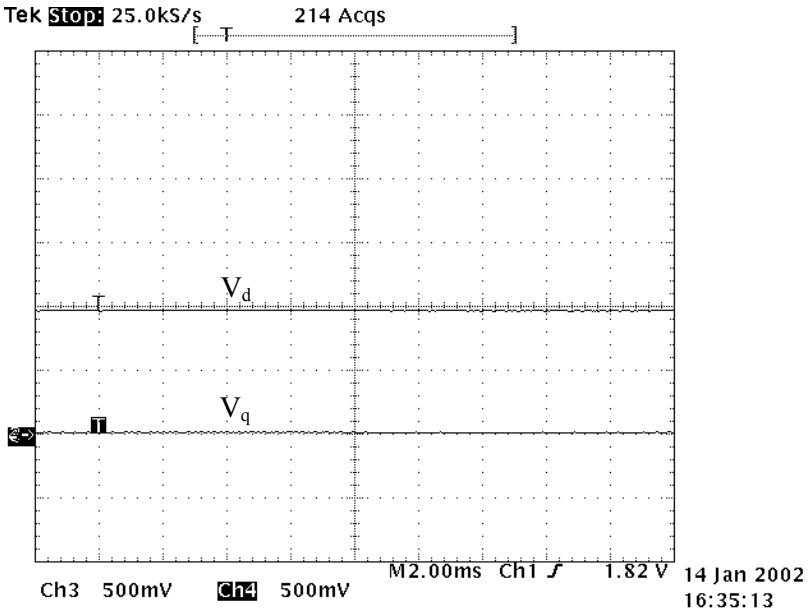

Figure 5 Vector variable Vdq versus theta

The vectors Valphabeta_rev and Vabc_rev, which represent the back-transformed quantities, obviously are, apart from eventual rounding errors, identical to Valphabeta and Vabc, respectively. As an example, the following figure shows a comparison between Va and Va_rev both in the time domain and in xy-representation.

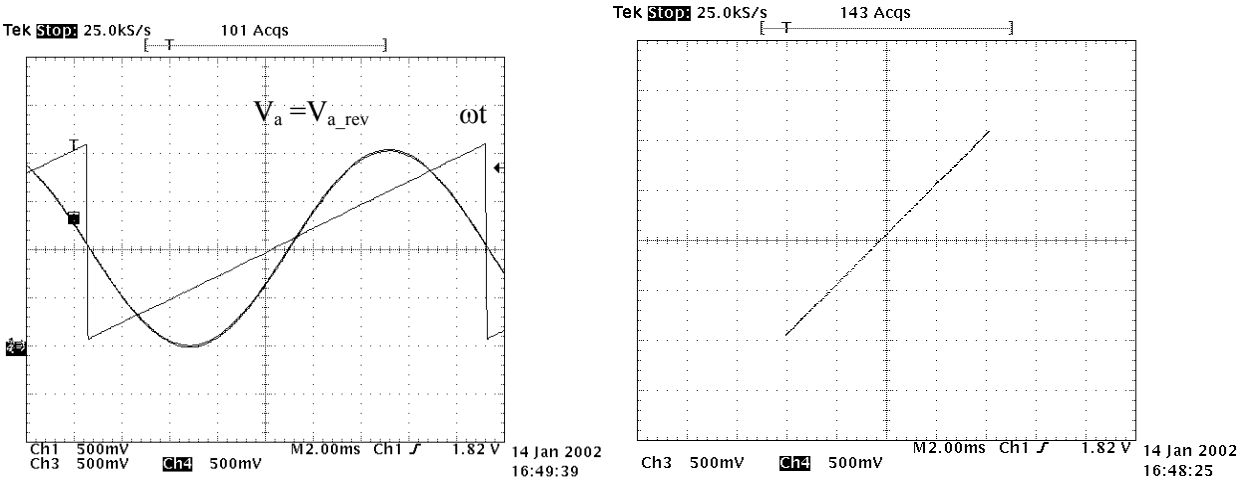

Figure 6 Va and Va_rev versus theta (left) and in xy representation (right)