AN2154
Application note

Space vector modulation using 8-bit ST7MC microcontroller and ST7MC-KIT/BLDC starter kit

Introduction

Due to its compatibility for use in vector control (field orientation) of AC motors, Space Vector Modulation (SVPWM) is one of the most widely utilized techniques to generate sinusoidal line-to-line voltages and currents with a three-phase inverter.

This application note first illustrates the theory of Space Vector Modulation and then describes an algorithm which can be utilized to implement SVPWM using the 8-bit microcontroller ST7MC.

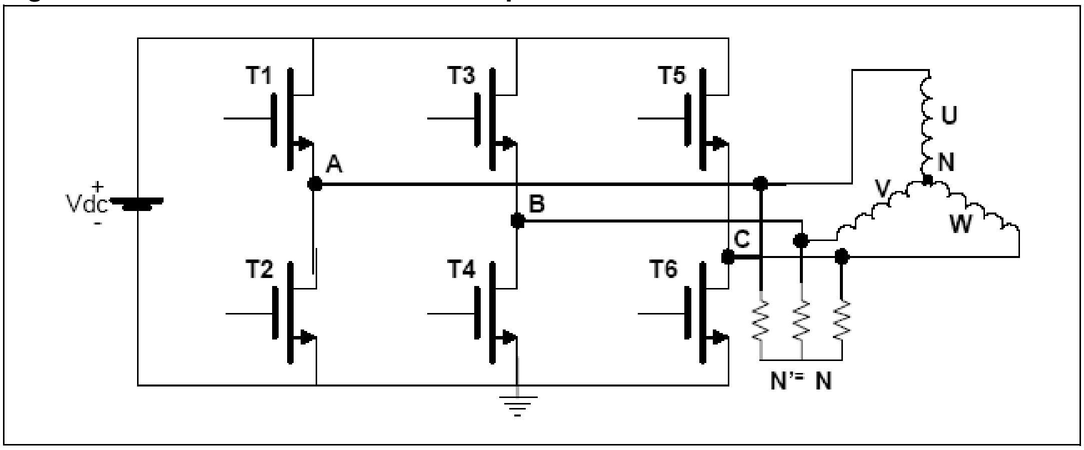
Contents

List of figures

1 Space vector modulation theory

Figure 1 shows a three-phase inverter connected to a star connected motor.

Figure 1. Motor connected to a three-phase inverter



Obviously, to avoid shorting the DC source V_{dc}, only one transistor for each leg must be switched on at one time. There are only eight possible switching states depending on the transistors that are turned on:

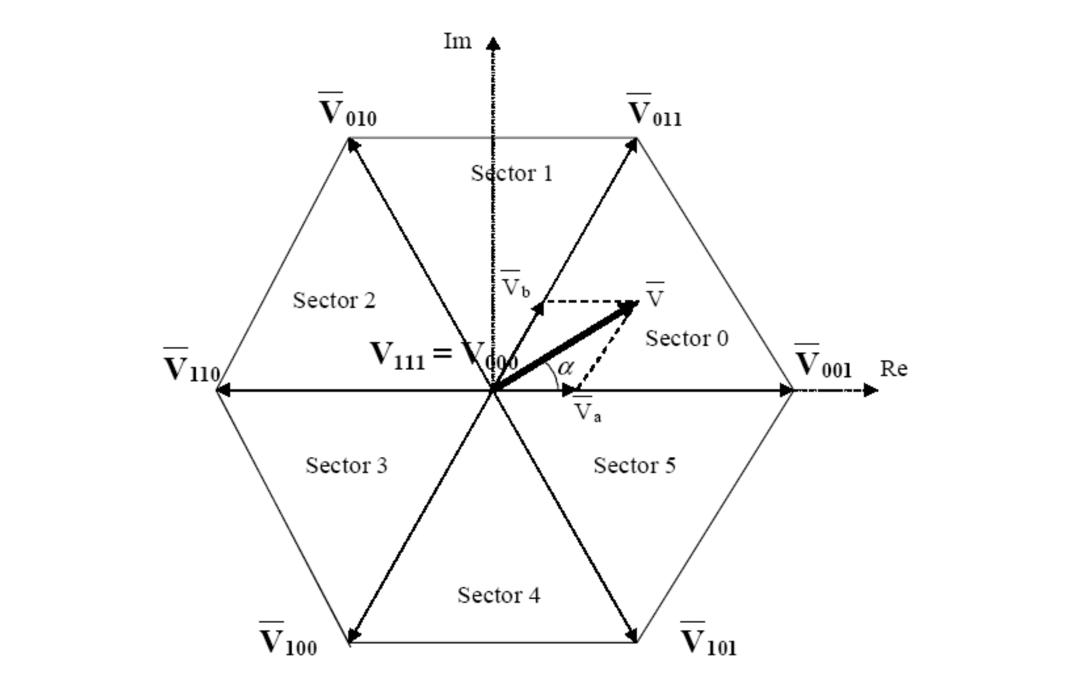
Table 1. Three-phase Inverter allowed states

State	Conducting transistor
000	T2, T4, T6
001	T2, T4, T5
010	T2, T3, T6
011	T2, T3, T5
100	T1, T4, T6
101	T1, T4, T5
110	T1, T3, T6
111	T1, T3, T5

Please note that in states 000 and 111 (null states), points A, B, and C are shortened to the negative and positive DC bus links respectively through low side and high side transistors so there are no line-to-line voltages applied to the motor. On the contrary, in the remaining six states, a voltage is present across the motor phases, causing a current to flow through statorical windings and flux to be produced by this current.

It is very useful to express the states reported in *Table 1* as space vectors. The six active vectors and the two null states can be plotted on the complex plane as shown in *Figure 2*:

Figure 2. Space vector diagram



Each one of the reference voltages V100, V110,… corresponds to one of the switching states reported in *Table 1*. In order to obtain sine wave voltages on the motor phases, a continuously rotating vector V must be produced so that one complete turn of the vector V on the complex plan represents one electrical turn of the motor (coincident with one mechanical turn in case of one pole pair motor). As it will be subsequently demonstrated, when linear modulation is applied, the circumference described by the rotating vector must stay inside the hexagon shown in *Figure 2*. The goal of space vector modulation is exactly to generate the appropriate PWM signals so that any vector V inside the hexagon can be produced by "time weighting" the two reference vectors that bound the sector in which V lies. Considering *Figure 2*, vector V can be expressed as:

Equation 1

$$\overline{V} = \overline{V}_a + \overline{V}_b$$

where V_a and V_b, are respectively components of vector $\overline{V}$, with respect to $\overline{V}_{001}$ and $\overline{V}_{011}$. As mentioned earlier, vectors $\overline{V}_a$ and $\overline{V}_b$ will be generated by applying respectively the reference vectors $\overline{V}_{001}$ and $\overline{V}_{011}$ for a well-defined percentage of PWM period T_0. Therefore, by defining t_a, t_b and t_0 as the permanency time of vectors, $\overline{V}_{001}$, $\overline{V}_{011}$ and $\overline{V}_{111}$ (or $\overline{V}_{000}$) inside the PWM period, it is possible to write:

Equation 2

$$\overline{V} = \overline{V}_a + \overline{V}_b = \frac{t_a}{T_0} \cdot \overline{V}_{001} + \frac{t_b}{T_0}\overline{V}_{011} + \frac{t_0}{T_0}(\overline{V}_{111} \mathrm{or} \overline{V}_{000})$$

with

Equation 3

$$t_a = \frac{V_a}{V_{001}} \cdot T_0$$

Equation 4

$$t_b = \frac{V_b}{V_{001}} \cdot T_0$$

Equation 5

$$t_0 = T_0 - t_a - t_b$$

Moreover, referring to *Figure 2* it is easy to demonstrate that:

Equation 6

$$V \cdot \sin\left(\frac{\pi}{3} - \alpha\right) = V_a \cdot \sin\frac{\pi}{3}$$

Equation 7

$$V \cdot \sin\alpha = V_b \cdot \sin\frac{\pi}{3}$$

from which it is possible to get:

Equation 8

$$v_a = \frac{2}{\sqrt{3}} \cdot V \cdot \sin\left(\frac{\pi}{3} - \alpha\right)$$

Equation 9

$$V_b = \frac{2}{\sqrt{3}} \cdot V \cdot \sin\alpha$$

If we now substitute *Equation 8* and *9* into *Equation 3* and *4*, it is possible to compute times t_a, t_b and t_0 as a function of angle á and of modulation index m_i=V/V_{xxx}:

Equation 10

$$t_a = \frac{2}{\sqrt{3}} \cdot T_0 \cdot m_i \cdot \sin\left(\frac{\pi}{3} - \alpha\right) \qquad 0 \leq \alpha \leq \frac{\pi}{3}$$

Equation 11

$$t_b = \frac{2}{\sqrt{3}} \cdot T_0 \cdot m_i \cdot \sin\alpha \qquad 0 \leq \alpha \leq \frac{\pi}{3}$$

Equation 12

$$t_0 = T_0 - t_a - t_b$$

Please note that when the modulation index exceeds √3/2, the value of t_0 can become negative for certain values of α. Since this does not have a physical meaning, it can be affirmed that the maximum value of m_i guaranteeing the proper working for space vector modulation in linear region is exactly √3/2. Higher values of m_i would lead to overmodulation, where *Equation 10* - *12* are no longer valid.

Graphically, this means that for space vector modulation to work properly, the magnitude of the vector V must be small enough to ensure that the vector is totally contained inside the hexagon shown in *Figure 2*.

2 The PWM generator block of ST7MC motor control peripheral for SVPWM

This section provides a basic functional description of the 12 bit PWM generator embedded into the ST7MC motor control peripheral.

Figure 3 schematizes the block diagram of the PWM generator. The 3 PWM signals are generated using a freerunning 12-bit PWM Counter and three 13-bit Compare registers for phase U, V and W: MCMPU, MCMPV and MCMPW registers. A fourth 12-bit register (MCMP0) is needed to set up the PWM carrier frequency. All three PWM signals (Phases U, V and W) are then directed to the channel manager, in which dead time may be added.

The 12-bit PWM Counter clock is supplied through a 3-bit prescaler to allow the generation of lower PWM carrier frequencies. It divides F_{mtc} by 1, 2, 3,..., 8 to get $F_{counter}$.

The PWM generator can work in center-aligned or edge-aligned mode, depending on the CMS bit setting in the MPCR register:

- Center-aligned Mode (CMS bit = 1): In this operating mode, the PWM Counter counts up to the value loaded in the 12-bit Compare 0 register (MCMP0). Subsequently it counts down until it reaches zero and restarts counting up. The PWM signals are set to '0' when the PWM Counter reaches, in counting up, the corresponding 13-bit Compare register value, and they are set to '1' when the PWM Counter reaches the 13-bit Compare value again in counting down. *Figure 4* shows a center-aligned PWM waveform, where the Compare 0 register value is equal to 8.
- Edge-aligned Mode (CMS bit = 0): In this operating mode, the PWM Counter counts up to the value loaded in the 12-bit Compare Register. Then the PWM Counter is cleared and it restarts counting up. The PWM signals are set to '0' when the PWM Counter reaches, in counting up, the corresponding 13-bit Compare register value, and they are set to '1' when the PWM Counter is cleared. *Figure 5* shows an edge-aligned PWM waveform, where the Compare 0 register value is equal to 8. Both in center-aligned and edge-aligned modes, the four Compare registers (one Compare 0 and three for the U, V and W phases) are updated (and an Update 'U' event interrupt is generated if enabled) when the PWM counter underflows or overflows, and the 8-bit Repetition countdown has reached zero.This means that data is transferred from the preload compare registers to the compare registers every N cycles of the PWM Counter, where N is the value of the 8-bit Repetition register in edge-aligned mode. When using center-aligned mode, the repetition countdown is decremented every time the PWM counter overflows or underflows. Although this limits the maximum number of repetitions to 128 PWM cycles, this makes it possible to update the duty cycle twice per PWM period.

Figure 3. The 12-bit PWM block generator of ST7MC motor control peripheral

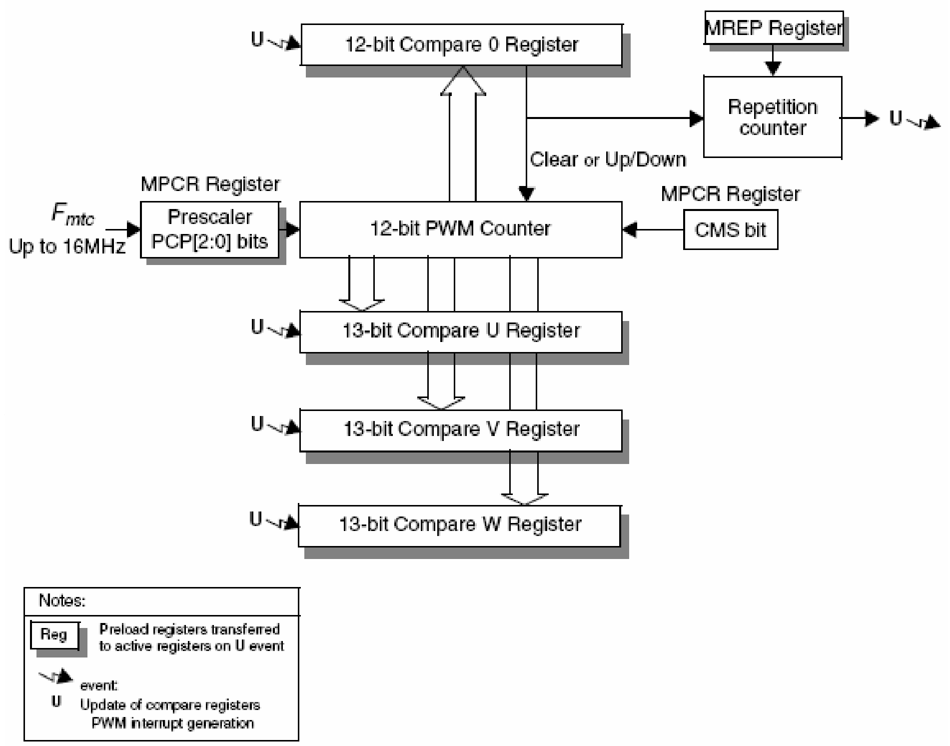


Figure 4. Center-aligned PWM waveform: compare 0 register = 8, compare register value = 4

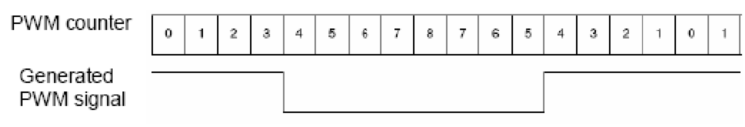
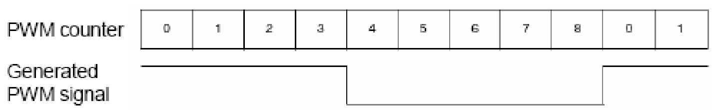


Figure 5. Edge-aligned PWM waveform: compare 0 register = 8, compare register value = 4



2.1 Symmetrical space vector modulation

As discussed in *Chapter 2* , the goal of SVPWM is to generate three appropriate PWM signals in order to create, on the phases of the motor, three vectors V shifted (by 120° between each other) and continuously rotating inside the space vector hexagon on the complex plane. Moreover, it has been also discussed that, to create vectors V inside one of the six sectors, the reference vectors which bound that sector have to be "time weighted" using *Equation 10* - *12*.

There are several possible ways to distribute time t_0 (associated to one of the two null sectors) inside the PWM period.

The first possibility is shown in *Figure 6*. It utilizes ST7MC Motor control peripheral 12-bit PWM generator in edge-aligned mode.

Please note that the edge alignment implies the presence of the passage between the null state 000 and the null state 111 and, then, the need of three contemporary switches commutation.

Literature demonstrates that, in order to reduce the number of commutation and switching losses, it is preferable to utilize a states sequence where successive states are also adjacent. This means that passing from one state to the successive one should occur with only one switch commutation.

This implies that the null state 111 must be always situated after one of the states 110, 101, 011, and null state 000 must be always situated after one of the states 100, 001, 010. *Figure 7* shows a typical switching pattern generated by ST7MC Motor control peripheral 12 bit PWM generator in "center-aligned" mode. It can be observed that the above rules are respected.

Please be aware that, to be coherent with what has been stated in the previous paragraph, the PWM period should be reported as T_0, while in *Figure 7* it has been indicated as $1/f_s$. It is convenient, in fact, from this point on, to indicate with T_0 only a half of the PWM period. It is easy to understand that *Equation 10* - *12* are still valid in the case that t_a, t_b are taken as shown in *Figure 7*.

Please observe that time t_0, associated with the null states, has been divided into two parts: t_0‘ is relative to the permanency time of state 111 and t_0“ is relative to the permanency time of state 000. Since *Equation 12* does not give any information about the values of these two times, their choice could be arbitrary. Nevertheless, literature demonstrates that the best total harmonic distortion on line to neutral voltage and on motor current is achieved when:

Equation 13

$$t'_0 = t''_0 = \frac{t_0}{2}$$

In this case the modulation is called symmetrical SVPWM.

Figure 6. Edge-aligned SVPWM patterns

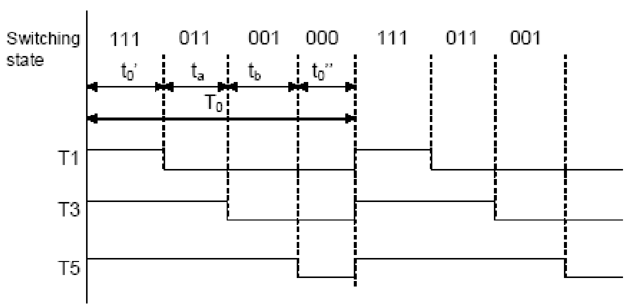
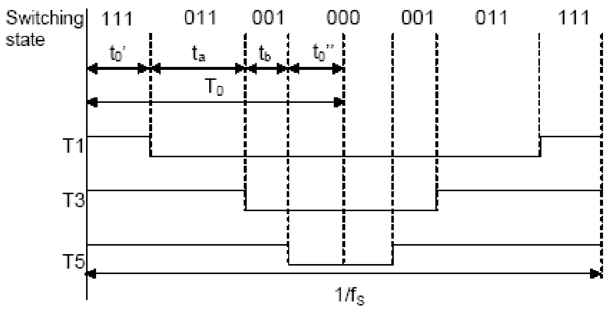


Figure 7. Center-aligned SVPWM patterns



3 ST7MC implementation of symmetrical SVPWM

A practical implementation of symmetrical SVPWM has been implemented using ST7MC, the STMicroelectronics 8-bit microcontroller dedicated to Motor Control. Thanks to its embedded Motor control peripheral, the space vector modulation can be easily implemented without excessive computational load for the CPU. The firmware developed is strongly based on the three-phase AC Induction Motor Control Software Library, downloadable from the web site: www.st.com and widely described in application note AN1904 (downloadable from the same web page). This library can handle AC asynchronous motor control in open loop and closed loop for optimum slip regulation using sinusoidal modulation with 3rd harmonic injection. Starting from the mentioned library, the symmetrical SVPWM has been generated mainly by modifying the 'U' event interrupt service routine dedicated to the sinus plus 3rd harmonic waveform generation.

3.1 Fixed settings

The aim of this paragraph is to illustrate which of the parameters defined in AC Induction Motor control library have been fixed during the symmetrical SVPWM algorithm development. All these settings are not mandatory, but a modification of one of them could require some firmware modification in order to restore the full firmware functioning. Relating to AC asynchronous motor control library, the main parameters which have been set are:

- Speed regulation: open loop operation has been selected.
- PWM pattern: center-aligned PWM must be chosen for symmetrical SVPWM.
- PWM frequency: it has been set equal to 15.625 kHz. This has been achieved by fixing MCMP0 register at 255 and by setting the prescaler to 1 ($F_{counte}r=F_{mtc}/2$). This choice allows both the elimination of the acoustical noise introduced by the switches and to perform all the necessary computations for PWM patterns generation in 8-bit format.
- Repetition counter: it has been fixed at 2 so that a U event is generated every 1.5 PWM periods.
- Accordingly with ST7MC Starter Kit schematic rev. C-6, the index modulation is adjustable from potentiometer RV2 in the range between 0 to 0.866, and the output waveform frequency is adjustable from potentiometer RV1 in the range between 3 and 225 Hz. Maximum and minimum frequency can be eventually varied by changing HIGHEST_FREQ and LOWEST_FREQ parameters values in MTC_param.h header file.

3.2 Symmetrical SVPWM algorithm description

As discussed in *Chapter 1*, in order to generate SVPWM, vector V must rotate continuously inside the space vector hexagon. This implies that time t_a, t_b and t_0 must continuously vary following *Equation 10* - *12*. Since these equations are only valid for α in the range between 0 and $\pi/3$, it could be very convenient to represent angle á by an 11-bit variable. This way, the most significant 3 bits can be used to indicate the sector, and the least significant 8-bits can contain the angle of vector V inside the identified sector. The resolution is given by $\pi/(3.256)$ radians equivalent to 0.234 degrees. In practical implementation, to keep the frequency resolution high, it is necessary to store angle α in a 16-bit variable. According to the previous discussion, the most significant 3 bits identify the sector, the successive 8 bits are used to identify the angle inside the sector and the least significant 5 bits are not used for t_a, t_b and t_0 computation.

In order to reduce the CPU computational load, a look-up table has been used for storing time t_a and t_b using *Equation 10* and *11* with the maximum admissible value of modulation index m_i (that is 0.866).

Moreover, comparing *Equation 10* and *11* it is possible to note that the t_a table is exactly the same as the t_b table in reverse. This means that only one look-up table is necessary. During runtime, the look-up times t_a and t_b must be scaled accordingly with the value of modulation index imposed from the user. This is achieved in two steps: first, times t_a and t_b are multiplied by an 8-bit variable (SineMag) read from potentiometer RV2 and, therefore, only the result of the most significant byte is considered (that is the equivalent of a division by 256).

After t_a and t_b scaling, the three values to be loaded into the compare registers must be calculated to generate the waveform shown in *Figure 7*. Let us suppose, for example, that vector V is, in a well known instant, situated in sector 0 and, therefore, it has to be "time weighted" between state 001 and 011 (this situation is exposed in *Figure 7*). The values to be loaded into the compare registers are:

Equation 14

$$\text{Phase U compare register (MCMPU)} = \left(\frac{t_0}{2}\right) \cdot F_{counter}$$

Equation 15

$$\text{Phase V compare register (MCMPV)} = \left(\frac{t_0}{2} + t_a\right) \cdot F_{counter}$$

Equation 16

$$\text{Phase W compare register (MCMPW)} = \left(\frac{t_0}{2} + t_a + t_b\right) = \left(T_0 - \frac{t_0}{2}\right) \cdot F_{counter}$$

Moreover, as it is:

Equation 17

$$T_0 = \frac{1}{(2 \cdot f_s)} = \frac{1}{(2 \cdot 15.625)} = 32\ \mu s$$

and

Equation 18

$$MCMP0 = T_0 \cdot F_{counter} - 1 = 255$$

it follows that T_0-t_0 is simply equal to ~t_0 (i.e. NOT(t_0)).

In addition, considering the equation:

Equation 19

$$\frac{t_0}{2} = \frac{(T_0 - (t_a + t_b))}{2} = \frac{(t_a \sim t_b)}{2}$$

it is possible to express Phase X compare registers (X=U, V, W) as function of only t_a and t_b ($F_{counter}$ dependency will be omitted from this point on):

Equation 20

$$\text{Phase U compare register} = \sim\left(\frac{(\sim t_a + t_b)}{2}\right)$$

Equation 21

$$\text{Phase V compare register} = \sim\left(\frac{(\sim t_a + t_b)}{2}\right) + t_a$$

Equation 22

$$\text{Phase W compare register} = \sim\left(\frac{(\sim t_a + t_b)}{2}\right)$$

A similar line of reasoning can be followed for the other sectors. *Table 2* shows the values to be loaded into the Phase U, V and W compare registers depending on the sector in which vector V is located. *Table 3* lists and shortly describes the variables utilized in SVPWM algorithm, while *Figure 8* shows the algorithm block diagram.

Table 2. Compare registers values

Sector	Phase U compare register (MCPUH)	Phase V compare register (MCPVH)	Phase W compare register (MCPWH)
0	$\frac{(\sim(t_a + t_b))}{2}$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right) + t_a$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right)$
1	$\frac{(\sim(t_a + t_b))}{2}$	$\sim\left(\frac{\sim t_a + t_b}{2}\right)$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right)$
2	$\frac{(\sim(t_a + t_b))}{2} + t_a$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right)$	$\frac{(\sim(t_a + t_b))}{2}$
3	$\sim\left(\frac{\sim t_a + t_b}{2}\right)$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right) + t_b$	$\frac{(\sim(t_a + t_b))}{2}$
4	$\sim\left(\frac{\sim t_a + t_b}{2}\right)$	$\frac{(\sim(t_a + t_b))}{2}$	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right) + t_a$
5	$\sim\left(\frac{(\sim t_a + t_b)}{2}\right) + t_b$	$\frac{(\sim(t_a + t_b))}{2}$	$\sim\left(\frac{\sim t_a + t_b}{2}\right)$

Table 3. Variables utilized in SVPWM algorithm

Variable name	Variable size	Physical meaning	Resolution
Phase	16 bit (0x00000xBFFF)	Angle α	0.234 degrees
SineFreq	16 bit	$d\alpha/dt$	0.212 Hz
T_a	8 bit	t_a	12 bit timer resolution (125 ns)
T_b	8 bit	t_b	12 bit timer resolution (125 ns)
SineMag	8 bit	m_i*256	$\approx$ 0.0039
t_0	It is not a defined variable. It is always computed as T_0- t_a- t_b		
T_0	T_0 is not a defined variable. It is constant at 15.625 kHz carrier frequency		

4 Conclusions and experimental results

Starting from the ST7MC three-phase asynchronous motor control library, an algorithm for symmetrical SVPWM has been developed. This section illustrates the measurement carried out for CPU load evaluation and shows the oscilloscope capture of obtained output waveforms.

4.1 The CPU load evaluation

The contribution of the implemented algorithm to the CPU load has been estimated measuring the U interrupt routine execution time at 8 MHz clock frequency with the usage of ST7MC emulator (ST7MDT50-EMU3). The average execution time of the above mentioned routine turned out to be equal to:

Equation 23

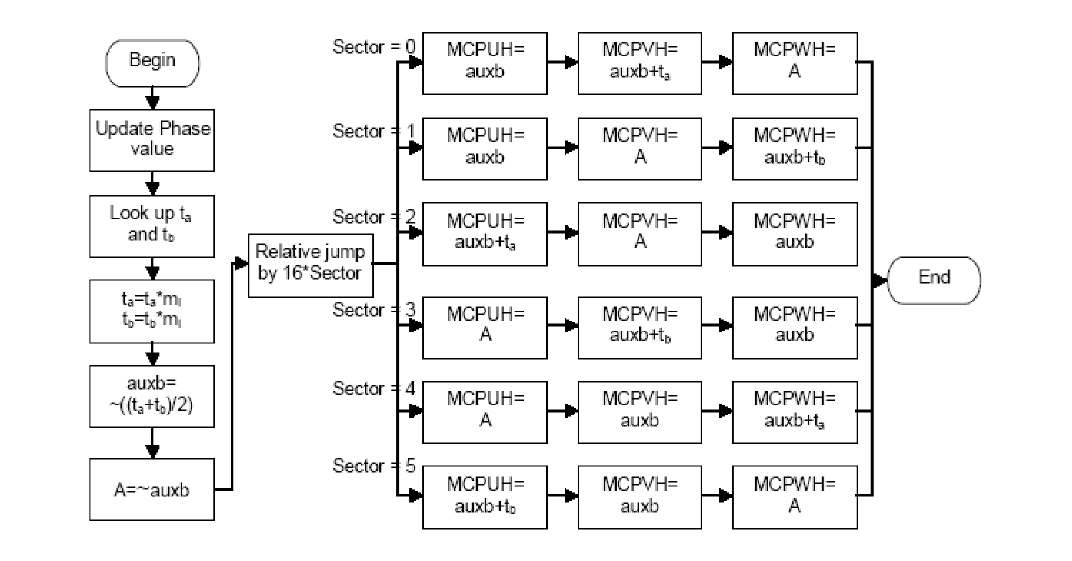
$$t_{ex} = 23.1\ \mu\text{sec}$$

The contribution of SVPWM implementation to the CPU load has been, therefore, computed in the exposed situation using relationship:

Equation 24

$$CPU_{LOAD} = \frac{t_{ex}}{0.5 \cdot t_{PWM} \cdot (REP_{COUNTER} + 1)} = 24.1\%$$

Figure 8. The update interrupt service routine block diagram



4.2 The output waveform captures

Referring to *Figure 2*, *Figure 9* illustrates the voltage of neutral point N with respect to ground (Channel 1), the voltage of point A with respect to ground (Channel 2), and the phase voltage of one of the three phases of the motor (V_{AN}, Channel 3). *Figure 10* shows the magnitude spectrum of the motor phase voltage.

Figure 9. Phase voltage, line-to-ground voltage and neutral point-to-ground voltage

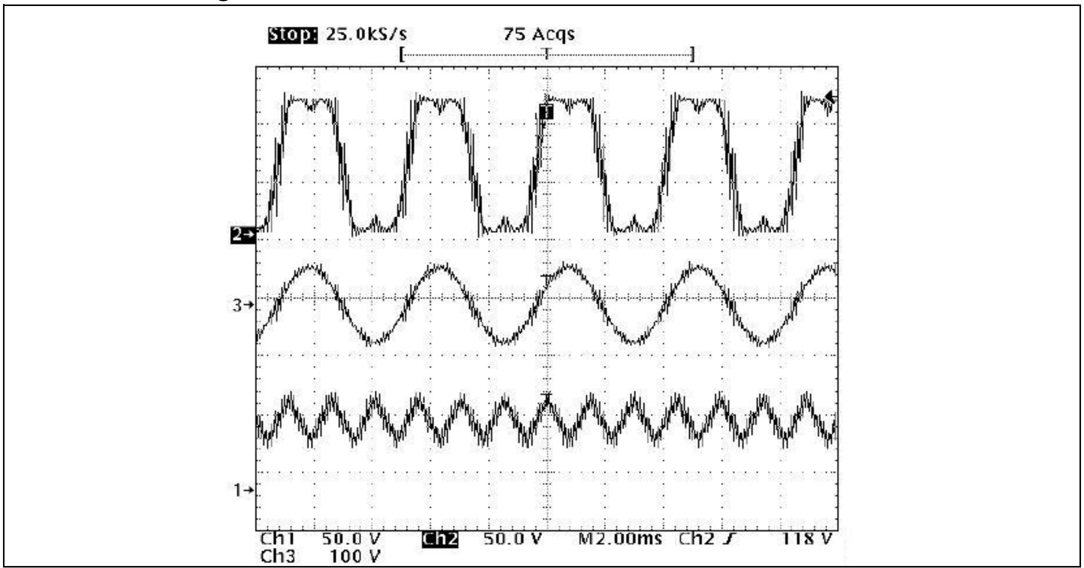
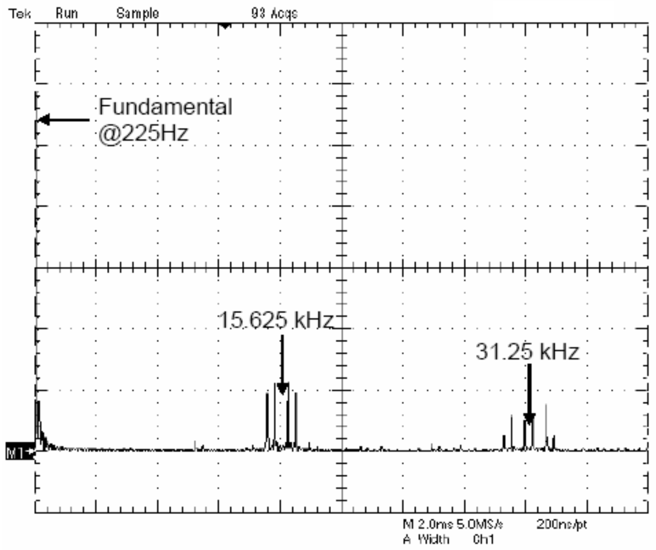


Figure 10. Magnitude spectrum of the motor phase voltage



5 Revision history

Table 4. Document revision history

Date	Revision	Changes
24-May-2005	1	Initial release
19-Nov-2007	2	Removed references to obsolete products
21-Nov-2007	3	Watermark removed, no content change