American J. of Engineering and Applied Sciences 2 (2): 488-493, 2009
ISSN 1941-7020
© 2009 Science Publications

Implementing a Vector Controller Using 68k Processors

Mohammad Bagher, Bannae Sharifian, Mohammad Reza Feyzi,
Seyed Hossein Hosseini and Reza Valinia
Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

Abstract: Problem statement: This study described the design of a 3-phase AC Induction Motor (ACIM) vector control drive with position encoder coupled to the motor shaft. **Approach:** It was based on free scale's (Motorola's) 68k micro processor devices. Although the free scale 56F80x (56800 core) and 56F8300 (56800E core) families were well-suited for digital motor control and offer all things was needed, but we decided to realize a complete vector controller with a powerful 68k processor. **Results:** Obviously all 680X0 and many 683XX can overcome this task very easily, but we decided 68332 for time consuming because it combines high-performance data manipulation capabilities with powerful peripheral subsystems. All software and hardware was based on Peter J. Pinewski's nice research from Motorola. **Conclusion:** In this study the overall software algorithm and in two fellow papers the hardware schematics and performance will be described respectively.

Key word: Vector control, ACIM, PI, hi-time, MC68332, induction motor

INTRODUCTION

Traditional control methods, such as the Volts-Hertz control method, control the frequency and amplitude of the motor drive voltage. In contrast, vector control methods control the frequency, amplitude and phase of the motor drive voltage. The key to vector control is to generate a 3-phase voltage as a phasor to control the 3-phase stator current as a phasor that controls the rotor flux vector and finally the rotor current phasor.

Ultimately, the components of the rotor current need to be controlled. The rotor current cannot be measured because the rotor is a steel cage and there are no direct electrical connections. Since the rotor currents cannot be measured directly, the application program calculates these parameters indirectly using parameters that can be directly measured.

The technique described in this paper is indirect vector control because there is no direct access to the rotor currents. Indirect vector control of the rotor currents is accomplished using the following data:

- Instantaneous stator phase currents, ia, ib and ic
- Rotor mechanical velocity
- Rotor electrical time constant

The motor must be equipped with sensors to monitor the 3-phase stator currents and a rotor velocity feedback device.

By mapping the measured three phase stator currents as a vector onto a two axis (d-q) coordinate system, the stator current is broken into two component, id and iq, which are orthogonal to each other and are used to control the rotor flux and torque current respectively. From a practical point of view, the flux and torque currents are controlled through the motor voltages and slip frequency. How these are controlled is based on calculations made on the measured phase currents and speed. The hardware requires speed sensing and current sensing in addition to the six PWMs necessary to drive the inverter. Figure 1 illustrates a system which utilizes a MC68332 to implement a simple vector controller. The system uses the on-chip timer to implement the center-aligned PWMs and for the speed sensing input. A jointed MC145050 and QSPI are used by the algorithm to sense the phase currents and the torque (throttle) input. The control algorithm is executed in software and is a large factor in determining performance. Therefore, successful implementation is dependent upon how the software executes the control algorithm.

Implementing the algorithm: The critical loop of the vector control system is implemented as an interrupt service routine. This loop performs the current measurement and transformation, the speed sensing and calculation, the PI controller, slip calculation, modulation strategy, and any fault checking[1,2].

Corresponding Author: Mohammad Bagher, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

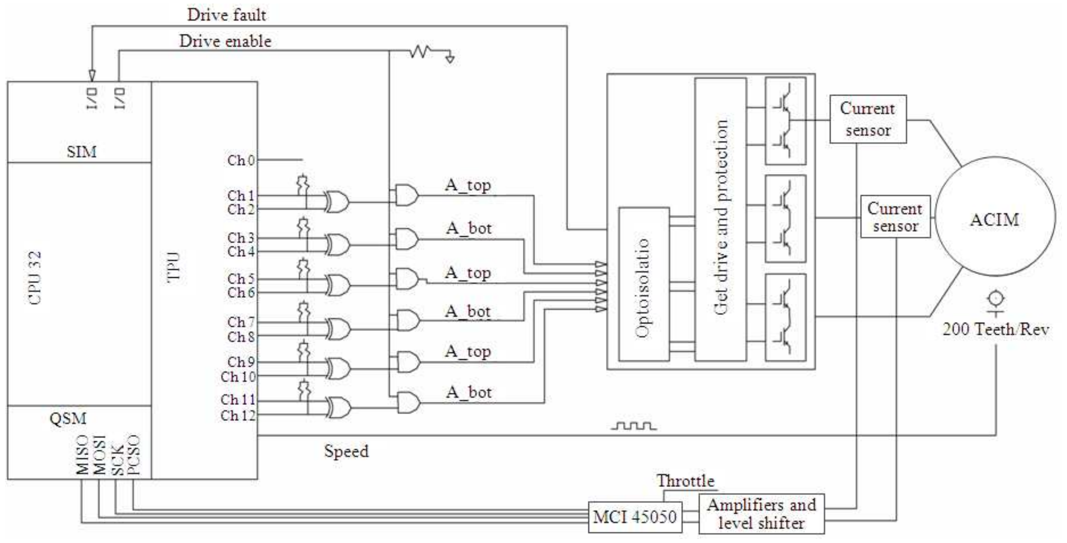

Fig. 1: Simple vector controller hardware block diagram

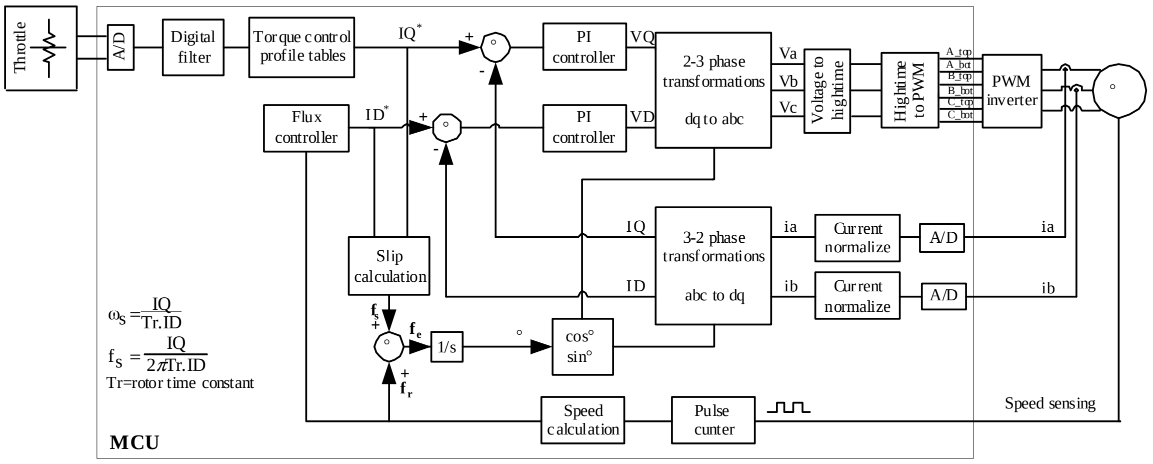

Fig. 2: Vector controller block diagram

The vector control loop is shown in Fig. 2. This system executes the interrupt service routine every 240 µs, which is two PWM period. The outer loop is used to implement the torque and flux controller, which are simple normalized lookup tables based on the A/D and speed readings, respectively. Transport and calculation delays are minimized by organizing the software so that the software code which does not depend on A/D reading is executed while the A/D conversions are taking place. Fig. 3 shows a flow diagram of the interrupt service routine.

Slip frequency calculator: The slip frequency calculator is the critical block which ensures correct field orientation between rotor flux and torque current. This is based on the IQ and ID currents as well as the rotor open circuit time constant (L_r/R_r). The equation for the slip calculator is given below:

$$f_s = \frac{1}{2\pi T_r}\frac{IQ}{ID} \tag{1}$$

where, $T_r = L_r/R_r$ is rotor time constant.

If the correct value of L_r/R_r is not known or is not achieved the vector control algorithm will operate in a “de-tuned” manner. This means that the flux and torque currents are improperly aligned with the rotor flux and that they are not truly de-coupled. In other words,

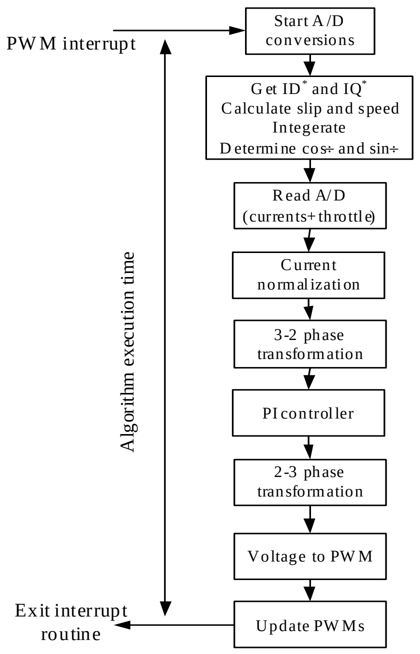

Fig. 3: Program flow diagram

a change in torque current would cause a change in rotor flux. The result is that instantaneous torque control is not achieved. This may be acceptable in applications such as an electric vehicle traction drive where the torque response is not critical and it is not worth the effort to implement more complicated techniques such as rotor temperature compensation or adaptive techniques.

Speed measurement/calculator: Speed measurement is one of the requirements for the vector controller. Hardware speed sensing can be accomplished with a toothed disk on the rotor shaft. The amount of teeth on the disk determines the resolution (the more teeth, the better the resolution). Speed measurement can be accomplished by a pulse type counter where the number of edges per time can be counted to give rotations per minute. Rotor frequency can be calculated by:

$$f_r = \frac{rpm \times p}{120} \quad (2)$$

where p is the number of poles.

In the example system, 200 tooth disk per revolution at a sample time of 5 ms equates to one measured edge ($0001) equaling 1 Hz rotor frequency. The precision of the speed sensor in the example system is 1 Hz.

Integrator: The integrator takes the frequency value and converts it to an angle for determining cosθ and sinθ. The integrator is a simple addition function of the form:

$$\theta = \text{Old\_theta} + \text{frequency} \times \text{interrupt\_rate} \times 2^{\text{Number-of-bit}}$$

In the example system, for a 16-bit angle variable and interrupt_rate = 244 µs, the above equation can be simplified to yield:

$$\theta = \text{Old\_theta} + \text{frequency (that is shifted left by four)}$$

Also the frequency resolution is 0.0625 Hz as calculated here:

$$\text{Frequency resolution} = 1/(2^{\text{Bits}} \times \text{interrupt\_rate})$$

cosè and sine: Both the vector rotators in the 2-3 phase transformation and the 3-2 phase transformations require cosè and sinè. In a microcontroller, there is no sine or cosine function. A way around this is through sine and cosine lookup tables. In actuality, only one table needs to be created since the cosine function is a 90 degree shift of the sine function. The lookup table itself is best constructed with a limited number of samples equaling a power of two (i.e., 256 point sine Table) and linearly interpolating between the sample points to produce a much larger effective table. As an example, the 332 vector controller utilizes a 16-bit angle variable. With a sine table of 256 points: 8 bits of the angle variable is used to index into the table and 8 bits are used for interpolation.

Current normalization: Current is sampled with the A/D converter. Many A/Ds, such as MC145050, are unipolar. This means they only accept a positive voltage level and the conversion result is unsigned. The currents, however, are sinusoidal in nature and have both positive and negative values. Because of this, the hardware must do some level shifting and amplifying to get the currents to be unipolar and to reach full scale of the A/D converter. The software must then convert the unipolar A/D readings into positive and negative values for use in the algorithm. This requires a simple subtraction of the zero current reading from the present phase current reading. The zero current reading can be obtained from initial startup before the drive is enabled. Another aspect of the A/D converter is that the converter may have a smaller bit size than the data size. The A/D readings should be left shifted such that full scale of the A/D is full scale of the data size. This is important for formatting the data into signed fractional numbers.

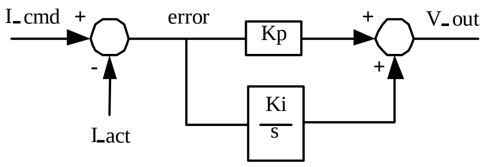

Fig. 4: Standard PI controller

3-2 Phase transformation: The three to two phase transformation is used to convert the measured AC phase currents, ia and ib, into the two DC current components, IQ and ID. The conversion is comprised of the a-b-c to d-q transformation and a vector rotator as defined as:

abc to dq transformation vector rotator

$$\text{iqs} = \text{ia} \qquad \text{IQ} = \text{iqs} \times \cos\theta - \text{ids} \times \sin\theta$$

$$\text{ids} = -\frac{1}{\sqrt{3}}\text{ia} - \frac{2}{\sqrt{3}}\text{ib} \qquad \text{ID} = \text{iqs} \times \sin\theta + \text{ids} \times \cos\theta$$

The above equations can be combined and simplified to yield:

$$\text{IQ} = \text{ia} \times \cos\theta + (0.57735 \times \text{ia} + 1.1547 \times \text{ib}) \times \sin\theta$$
$$\text{ID} = \text{ia} \times \sin\theta - (0.57735 \times \text{ia} + 1.1547 \times \text{ib}) \times \cos\theta$$

PI current controllers: The two PI regulators control the current components, IQ^* and ID^*, defined by the torque and flux controllers. These are standard PI controllers of the form that is depicted in Fig. 4:

The output voltage is the control variable which is adjusted to ensure zero error in actual and commanded current. In terms of the MCU, the PI controller is defined as a difference equation. Using $s = 1\text{-}z^{-1}$, the difference equations for the PI controllers become:

$$\text{VQ} = \text{VQ} \times z^{-1} + \text{kp}(\text{IQ\_err} - \text{IQ\_err} \times z^{-1}) + \text{ki} \times \text{IQ\_err}$$
$$\text{VD} = \text{VD} \times z^{-1} + \text{kp}(\text{ID\_err} - \text{ID\_err} \times z^{-1}) + \text{ki} \times \text{ID\_err}$$

where, z^{-1} is a delay operator. Therefore, $(\text{V}\times z^{-1})$ and $(\text{error}\times z^{-1})$ are previous values of the voltage and error respectively.

2-3 Phase transformation: The 2-3 phase transformation is used to convert the stator voltages, VQ and VD, into phase voltages va, vb, and vc. The conversion is comprised of the reverse vector rotator and the dq to abc transformation which are defined as:

Vector rotator dq-abc transformation

$$\text{vqs} = \text{VQ} \times \cos\theta + \text{VD} \times \sin\theta \qquad \text{va} = \text{vqs}$$

$$\text{vds} = \text{VD} \times \cos\theta - \text{VQ} \times \sin\theta \qquad \text{vb} = -\frac{1}{2}\text{vqs} - \frac{\sqrt{3}}{2}\text{vds}$$

$$\text{vc} = -(\text{va} + \text{vb})$$

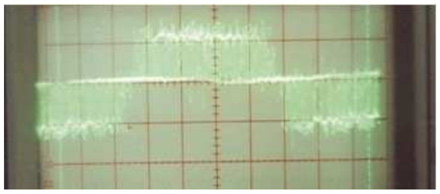

Fig. 5: Resulted line-line voltage

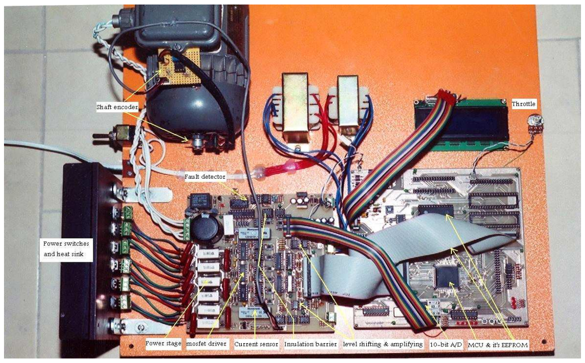

Fig. 6: Overall drive system

These equations can be combined and simplified to yield the equations:

$$\text{va} = \text{VQ} \times \cos\theta + \text{VD} \times \sin\theta$$
$$\text{vb} = -0.5\text{va} - 0.8660254 \times (\text{VD} \times \cos\theta - \text{VQ} \times \sin\theta)$$
$$\text{vc} = -(\text{va} + \text{vb})$$

High time calculations: Output of the 2-3 phase transformation block are the voltages for va, vb and vc. The final step is to convert these voltages into PWM duty cycles. The PWM duty cycles can range from 0-100% where 0% represents the negative peak of the sinewave and 100% represents the positive peak. This means the zero crossovers is at 50% duty cycle. The equations to convert the phase voltages to PWM duty cycles are given below:

$$\begin{aligned}\text{A-HIGH} &= \text{va} \times \text{Period}/2 + \text{Period}/2\\ \text{B-HIGH} &= \text{vb} \times \text{Period}/2 + \text{Period}/2\\ \text{C-HIGH} &= \text{vc} \times \text{Period}/2 + \text{Period}/2 \qquad (2)\end{aligned}$$

From practical point of view, the resulted line-line voltage may be similar to Fig. 5.

Experimental aspects: Because of the paper limitation, it is not possible to overcome evaluation of all schematic hardware and experimental results. In two follow papers, the schematic hardware and overall system performance will be described. But for a perspective, Fig. 6-8 show the overall evaluated system and its performance after gains adjusting.

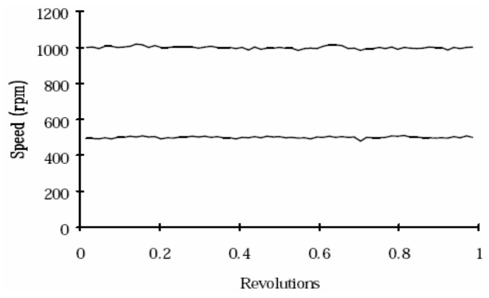

Fig. 7: Motor speed ripple in control mode

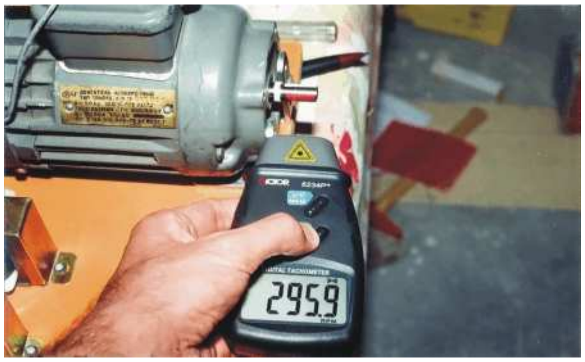

Fig. 8: Motor running during steady state

Gains adjusting are almost a hard work. The kp gain determine overall system performance and the ki gain is used to zeroes the error in steady state. At the first time these gain may set to zero. Then the kp gain increase until the response become as fast as possible with no overshoot. Then the ki will be set to a small number.

To verify the speed regulation and response time, it is better we examine the system in speed control mode rather than torque mode.

We use a simple frequency to voltage converter using lm331. The output of shaft encoder is fed to this simple circuit, so by using a storage oscilloscope we can analyze the speed tuning and response time for adjust ki, kp gains respectively (Fig. 7).

MATERIALS AND MATHODS

In general, this system consist of one processor board, one auxiliary board and a 3-phase induction motor. The processor board consist of a MC68332 as main processor and it's peripherals such as RAM, A/D converter, buffers and others. 16 bit data bus is constructed by two 32k*8 bit SRAMs. memory back-up circuitry that is combined of two Ultra-capacitor and MAX691 is sufficient for three month, so the needing for any EPROM, EEPROM or FLASH is removed. MC145050 is used for A/D conversion. This chip is well suited for MC68332 processor in 10 bit resolution.
The auxiliary board consist of some logic ICs and OP-AMPs for pulse and analog-signal conditioning respectively. It also has a insulation-barrier to completely insolate the controller-board from power-stage. Two CSNE151-100 hall-effect-sensor is used in this board for phase current sensing. The power-stage, also, is present in this board that itself consist of three IR2113 as gate-drivers, IRF840 as power switches, RCD Snubbers, over-current-protection circuitry and some others.

The induction Motor is a 3-phase, 220V/380V, 120W, two pole type, that a 200 pulse/rev shaft-encoder is coupled to it's shaft.

RESULTS

This study propose a basic vector control algorithm and it's implementation for induction motor control. It show that the control schema consist of several functional blocks. These blocks is discussed in some details and a few application tips for implementation of these blocks into hardware is offered. At last, experimental results show the validation of the proposed algorithm and it's implementation.

DISCUSSION

Today, there is a wide variety of DSPs and algorithms those are optimized for Motion-control. If our goal was only the vector control of the induction motor, then it was better that we choose other type of processors, and off-course, a great optimization in cost and developmental-time would be achieved. But our final goal was to design and construction of a very powerful Single-Board-Computer (SBC) using MC68040 that can be used for any processes. So we decided to choose one 68k family processor like 68332, because it is well suited for motion control tasks and off course the CPU32 has a great compatibility with 68040 processors.

CONCLUSION

Vector control systems, for the most part, are comprised of the same control blocks. However, some systems implement more precise calculation methods for determining and controlling rotor flux. In the next design we will use the very powerful MC68040 for this art. The proposed system in this study can be foundation to implementing more advanced and precise techniques using 68040.

ACKNOWLEDGEMENT

We thanks research center of the University of Tabriz for laboratory and material supports and foundations.

We also thank P. Pinewski and R. Valentine. from freescale (motorla) because all system hardware and software of the proposed system example are based on their nice works.

REFERENCES

1. Peter J. Pinewski, 1997. Implementing a simple vector controller. Proceeding of the American Control Conference, June 4-6, IEEE Xplore Press, NM., USA., pp: 262-266. DOI: 10.1109/ACC.1997.611798
2. Valentine, R., 1998. Motor Control Electronic Hand Book. McGraw Hill, New York, USA., ISBN: 0070668108, pp: 704.