Implementation of a Sensorless Speed Controlled Brushless DC drive using TMS320F240

Literature Number: BPRA072
Texas Instruments Europe
November 1997

Contents

List of Figures

Implementation of a Sensorless Speed Controlled Brushless DC Drive using the TMS320F240

ABSTRACT

The DSP Controller TMS320F240 from Texas Instruments is suitable for a wide range of motor drives. TMS320F240 provides a single chip solution by integrating on-chip not only a high computational power but also all the peripherals necessary for electric motor control. The present application note describes how a sensorless, speed controlled, brushless DC drive can be implemented using TMS320F240. This document shows that this sensorless solution is single chip and really cost effective in comparison with the sensored solution.

1. Introduction

Controlling a synchronous permanent magnet trapezoidal Bemf shape motor by direct current results in a simple low cost drive with a high mechanical power density. In order to reduce the system cost, it is desirable to eliminate the rotor position sensor used to operate the Brushless Direct Current (BLDC) motor. This document describes a method for commutation of a BLDC motor requiring no rotor position sensor and suppressing the position sensor cost at no cost other than software. The control described here operates in closed loop from zero speed. The initial rotor position is fixed by energizing one, arbitrarily selected phase. This application report covers the TMS320F240 DSP Controller, provides the sensorless control DSP Controller software and demonstrates some really effective benefits.

2. Sensorless Brushless DC Drive Presentation

2.1 The motor

The synchronous machine with permanent magnets used in this application note is a three-phase Y-connected motor. It has one magnetic non-salient pole pair on the rotor. The thooless stator is made of ironless windings. The stator phase inductance is 0.045mH (measured at 1kHz) and the phase resistance is 300mΩ. The maximum permissible current at 5000rpm is 2.9A and the torque constant is equal to 11.8mNm/A. It is assumed to have trapezoidal back electromotive force (Bemf) wave form shapes and is supplied with direct current. The DC bus voltage is 18V.

2.2 The control hardware

The control hardware can be either the TMS320F240 Evaluation Module introduced by Texas Instruments or the MCK240 developed by Portescap/Technosoft. In this application, the second board can be plugged directly onto the power electronics board. The two boards contain a DSP controller TMS320F240 and its oscillator, a JTAG, a RS232 link and the necessary output connectors. See the figure below depicting the EVM board.

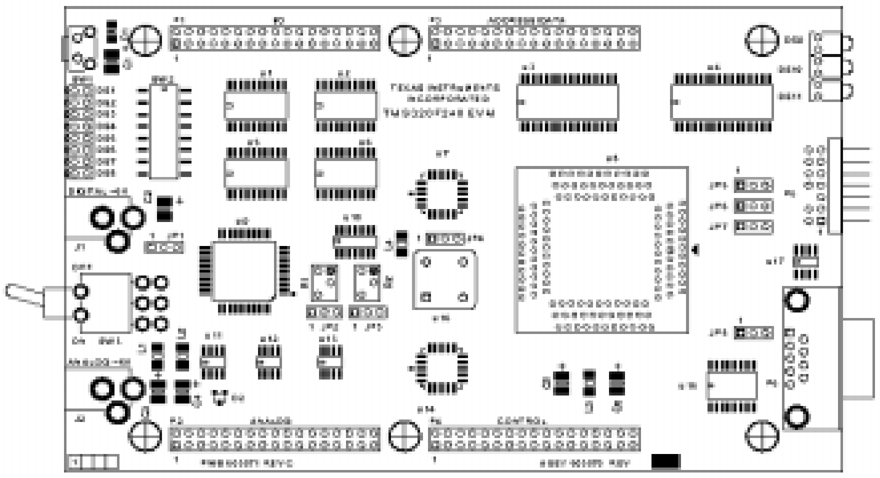


Figure 1: Top View of TMS320F240 EVM Board

2.3 The Power Electronics Hardware

2.3.1 The Power Conversion Dedicated Hardware

The power board is designed to support an 18V DC voltage supply and a 300W power range. The reader can find a complete detailed technical description of this inverter in a separate dedicated report [5]. The converter topology supports either sinusoidal currents (Three phases ON operation) or direct currents (two phases ON operation). The latter control is implemented in this application note. The figure below shows the converter which is used here.

The power switches used are power MOSFETs IRFP054. The pre-driver component selected is the IR2131 and the TMS320F240 PWM output signals are directly connected (no buffers). The pre-driver output signals go through a resistor and then directly to the power switches. The relative ground of the upper half-bridge is realized with bootstrap capacitors. This hardware configuration allows hard chopping as well as soft chopping operation. The chosen PWM strategy is the soft chopping mode with symmetrical modulation. All the power device security features are hard-wired (Shutdown, Fault, Clearfault, Itrip, reverse battery diode, varistor peak current protection).

Current sensing is achieved by a low cost shunt resistor. Its voltage drop is interfaced with TMS320F240 as shown in Figure 3. The sizing of the shunt resistor is such that, at maximum allowed current, the voltage drop across the shunt is equal to 0.5V (Voltage threshold to enable the driver’s over-current protection). The RC cell is designed to filter not only the commutation noises but also the chopping noise. A feedback gain is set so that the shunt voltage drop range utilizes the whole ADC input voltage range. So if the ADC voltage reference is set to five volts, the feedback gain should be set to ten.

A break feature is realized with another MOSFET and a power resistor. This feature is not used in the control software described because of the very small electrical time constant and hence the small amount of time necessary to get the Direct Current out of the windings. Also due to this small electrical time constant, the chopping frequency is set at 80kHz.

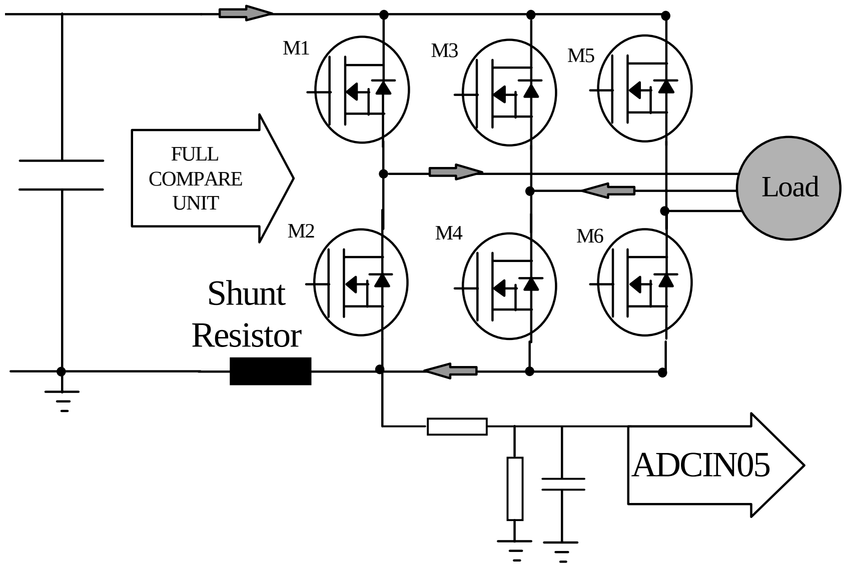


Figure 2: Power Electronics Topology

The bold arrows on the wires depict the Direct Current flowing into two motor phases.

2.3.2 The Sensorless Dedicated Hardware

The figure below shows the basic hardware necessary to run the sensorless control described. This shows that the system cost saving is highly efficient as the mechanical position sensor is fully replaced in terms of hardware by very low cost resistors.

In fact the sensorless BLDC drive presented here is much more cost effective than the ASIC or MCU based ones as these must include several additional operational amplifiers and/or ICs on the power board.

Note also that this hardware may be easily and freely modified to fit different motor characteristics. This makes this sensorless dedicated hardware more upgradable and versatile than ASIC and MCU based solutions, which require a new ASIC design and/or new op amp adjustments.

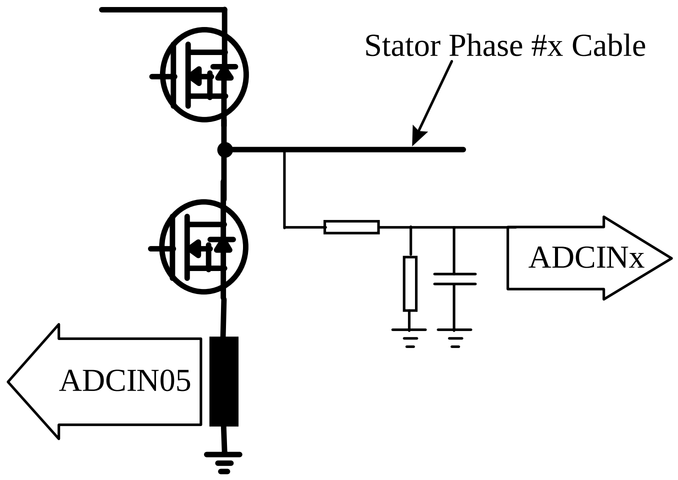


Figure 3: Basic Sensorless Additional Hardware

The resistance bridge is specified such that the maximum bridge output utilizes the full ADC conversion range. So if the ADC reference voltages are set to zero and five volts and if the DC bus voltage is 18V, the bridge ratio should be equal to 0.27. Note that the DSP Controller F240 ADC Unit requires a 1μs sample time per kΩ on the analog line. The filtering capacitor should filter the chopping frequency, so only very small values are necessary (in the range of nF). The sensorless algorithm is based only on the three terminal voltage measurements and thus requires only four ADC input lines out of the 16 available.

2.4 The control algorithm

According to [3], achieving a BLDC speed control requires **three control layers** to be performed. The innermost one is to get the **rotor position** in order to correctly commutate the stator flux. Once the rotor position is known, the magnitude of the stator flux has to be generated and controlled. Assuming that the stator flux is proportional to the current flowing in the stator coils, the control of the **stator flux magnitude** is equivalent to the control of the input current. The outermost control loop is the **speed regulation loop**. As the motor and the power electronics board used in this application are the same as the ones used in [3], to get more precise information about speed and current regulation please refer to [3]. Nevertheless, the corresponding software is included with the software given in the appendix. By contrast, the means of **obtaining the rotor position** and the corresponding software will be fully explained.

3. Sensorless Principles

The aim of this document is to present a DSP based solution able to deliver to the classic BLDC drive control the necessary rotor position without any help other than the F240 DSP Controller capabilities (no additional ICs; no star connection wired out of the motor housing). The following description gives a succinct theoretical background and covers every practical aspect.

3.1 Theoretical Background

According to [4] in the sensored control structure, the phases are commutated once every 60º mechanical rotation of the rotor. This implies that only six commutation signals are sufficient to drive a BLDC motor. Furthermore, an efficient control implies a synchronization between the phase Bemf and the phase supply so that the Bemf crosses zero once during the non-fed 60º sector. The next paragraph shows how it is possible to get the three Bemfs and their zero crossings.

The figure below depicts the motor terminal model, where L is the phase inductance, R is the phase resistance, E is the back electromotive force, Vn is the star connection voltage referenced to ground and Vx is the phase voltage referenced to ground. Vx voltages are measured by means of the DSP controller ADC Unit and via the resistance bridge depicted in figure 3.

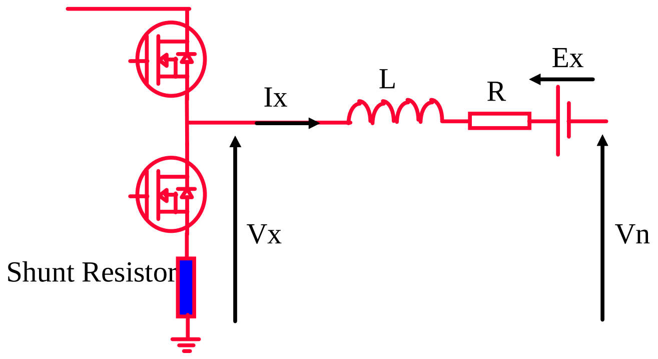


Figure 4: Stator Terminal Electrical Model

Each stator terminal voltage can be modelled as follows:

$$Vx = RIx + L\frac{dIx}{dt} + Ex + Vn\,.$$

As only two currents flow in the stator windings at any one time, two phase currents are opposite and the third one is equal to zero. Furthermore, knowing that the sum of the three stator currents is equal to zero (star wound stator) and given the following instantaneous Bemf wave forms:

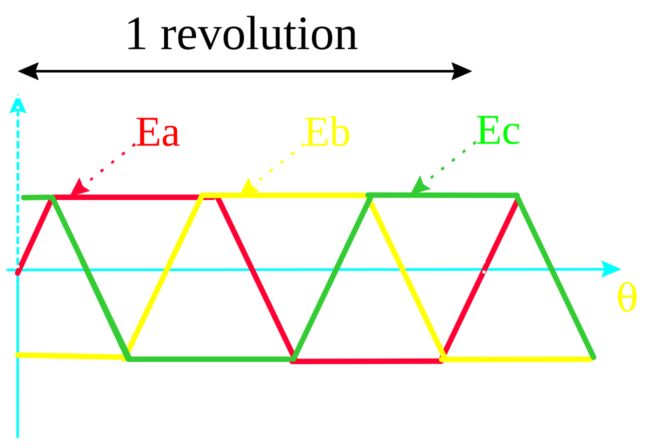


Figure 5: Bemf Wave-form Shapes

the sum of the three stator terminal voltages is equal to three times the neutral point voltage (Vn). So we have

$$Vn = \frac{1}{3} * \sum_{x=1}^{3} Vx\,.$$

For the non-fed phase (zero current flowing), the stator terminal voltage can be rewritten as follows:

$$Enon\ fed = Vnonfed - Vn = Vnonfed - \frac{1}{3} * \sum_{x=1}^{3} Vx\,.$$

As each of the Bemfs crosses zero twice per mechanical revolution, and as the Bemfs are numerically easy to compute, thanks to the DSP Controller, it is possible to derive the six required items of information regarding the commutation. The above mentioned equations are implemented in the code given below.

3.2 Practical Point of View

This chapter presents the practical implementation of the above theory. It also deals with the theory limitations and the practical solutions used to overcome them. Each software module will be addressed by one dedicated sub chapter.

3.2.1 Neutral Point Voltage Computation

As shown above, the neutral point voltage computation requires that we should know the three instantaneous terminal voltages referenced to ground. Let us explain how the ADC is managed in order to perform this real time task.

Due to the very small phase electrical time constant the chopping frequency has been set at 80kHz. The PWM period is then equal to 12.5µs. The current regulation loop period has been set to 50µs. So four PWM period flags occur in one current loop period. They are all acknowledged by the DSP Controller Core and the corresponding interrupt subroutine is initiated. Only when the current PWM period ISR is the first or the second (among the four available during one current control loop) is a conversion is started by software. At the End Of Conversion an interrupt to the core is requested, acknowledged and initiates processing of the two results. The following plot depicts the organization of the interrupts.

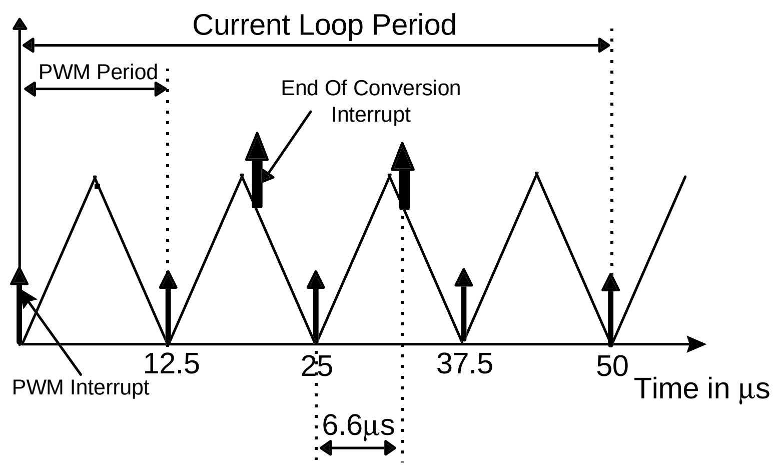


Figure 6: PWM Strategy and Operating System

The first conversion gives the phase current, and the second the phase terminal voltage information. Once these two results have been processed, the ADC input channels dedicated to the two other phase terminal voltage measurements are selected. The second End Of Conversion subroutine handles two tasks: firstly, the two conversion results and reloading of the current and second terminal voltage channels; secondly, the computation of the neutral point voltage according to the formula given above.

This solution assumes that the variation of the phase terminal voltage is negligible during one PWM period (12.5µs). Note that this real time sampling structure fits any chopping and regulation frequencies: for a motor requiring a 20kHz chopping frequency we can easily imagine a current regulation loop frequency at 10kHz and still having the three terminal voltages.

3.2.2 Bemf Zero Crossing Point Computation

Once the neutral point voltage is available it is necessary to get the Bemf of the non-fed phase. These is achieved by subtracting the computed neutral voltage from the non-fed phase terminal voltage. As we are interested in the zero crossing of the Bemf it is possible to look solely at the Bemf sign change; this only assumes that the Bemf scanning loop period is much shorter than the mechanical time constant. This function is computed after the three terminal voltage samples, once every 50µs and out of any ISR during the PWM duty cycle update.

3.2.3 Electrical Behaviour at Commutation Points

At the instants of phase commutation, high dV/dt and dI/dt glitches may occur due, for example, to the direct current level or to the parasitic inductance and capacitance of the power board. In addition to the Electro-Magnetic Interference troubles that they might generate, these glitches, as well as the non-symmetrical three phase system created by the power stage, can lead to a misreading of the computed neutral voltage. This problem is solved by discarding the first scans of the Bemf once a new phase commutation occurs. The discard duration is fully customizable as this is a software module. The duration depends on the power switches, the power board design, the phase inductance and the driven direct current. This parameter is system-dependent and should be set to a large value in the early development stages. Later on it is possible to tune to a very small discard duration in order to achieve a better control.

3.2.4 Some Bemf Zero Crossing Results

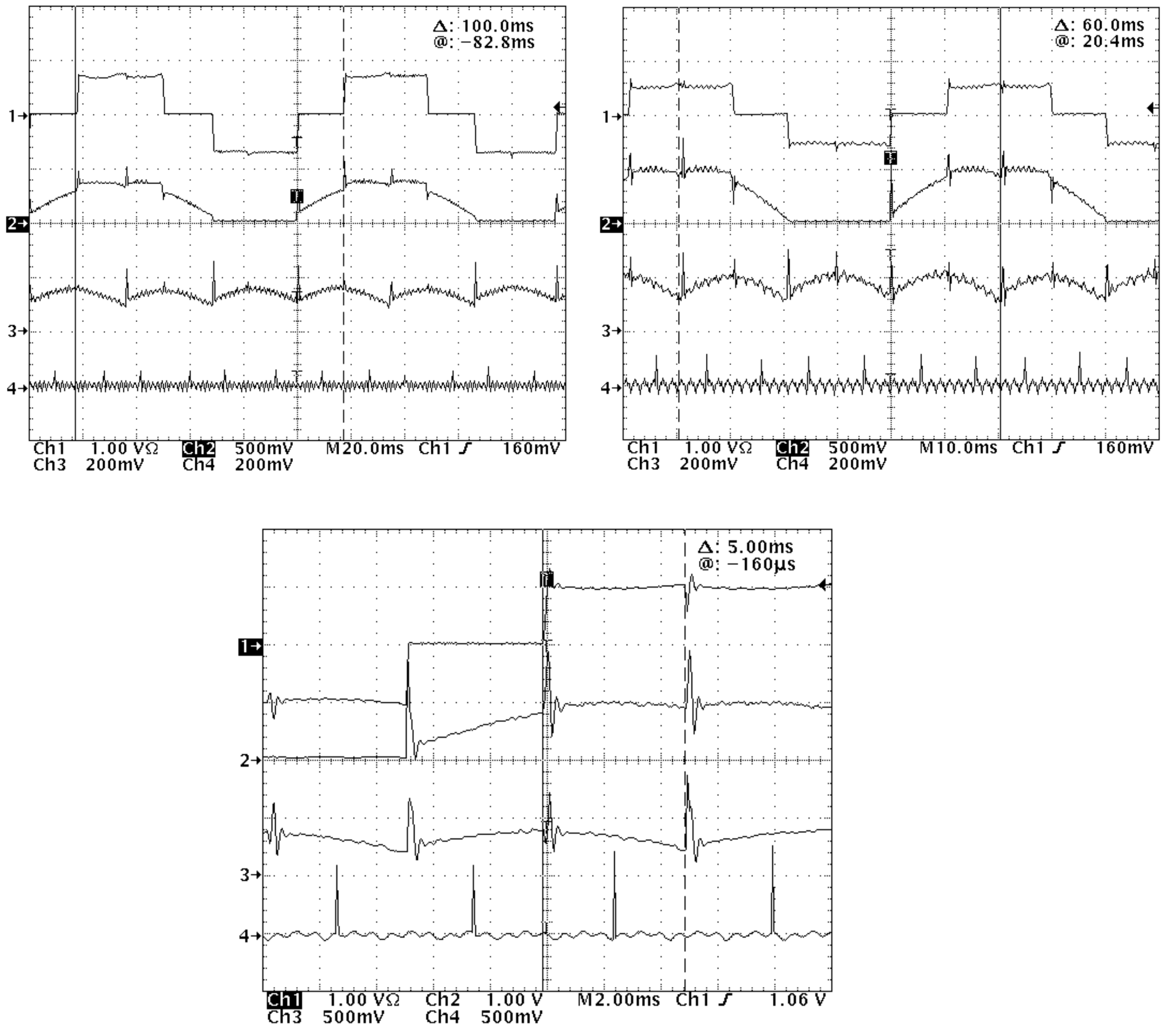


Figure 7: Zero Crossing Results at Different Running Speeds

On each oscilloscope picture the channel #1 represents one phase current, channel #2 is the corresponding half- bridge voltage, channel #3 is the computed neutral point voltage and the channel #4 gives one peak each time the software detects the zero crossing of the scanned Bemf. The different speeds tested here are respectively 600, 1000 and 2000rpm. The Bemf zero crossing algorithm has been successfully tested down to 30rpm. The most interesting information is given by the neutral voltage computation: even if there are very high commutation glitches and/or measurement noises the software recognizes them as glitches thus making it possible to detect the right Bemf sign change. This is achievable because of the high frequency of neutral point voltage computation loop. This high frequency can be reached thanks to the autonomous ADC peripheral and to the high DSP Controller CPU power. The channel #4 spikes (representing the Bemf zero crossing) find themselves in the middle of the non-fed sector. This synchronization between the Bemf and the 60º sector is the main result to achieve in order to get the highest motor performance.

3.2.5 Commutation Instants Computation

In an efficient sensored control the Bemf zero crossing events are displaced 30º from the instants of phase commutation. So before running the sensorless BLDC motor with help of the six zero crossing events it is necessary to compute the 30º shift of commutation points. In fact, depending on the different desired speed ranges, the angle shift might be to any angle. So a position interpolation function should be realized. In this control software it is implemented as follows: let T be the time that the rotor spent to complete the previous revolution and α be the desired shift angle. By dividing α by 360º and multiplying the result by T we obtain the amount of time (let us call it ‘shift time’) to be spent before commutating the next phase pair.

A question might be asked regarding the transient response of the system. Let us, therefore, assume that the motor slows down and see how the control reacts. The computed shift time will be too short in comparison with the actual shift time necessary; this early arrival of the commutation point will tend to accelerate the motor. There will be an opposite controller reaction if the motor accelerates. So a kind of natural robustness, arising from the control algorithm, is added to the speed control loop.

In spite of this advantage this shift time solution assumes that the motor speed variation between two revolutions is not too critical in comparison with the system dynamic behaviour.

In the software described, α is fixed to 30º. We can imagine an additional software function giving the shift angle as output and taking the mechanical speed as input. Two charts are presented below which show the computation of the commutation point in comparison with the three Hall effect sensor outputs. Channel #1 spikes represent the computed commutation points, channel #2, 3, 4 represent the Hall sensor output. The left chart shows results at low speed (380rpm) and the right chart the results at high speed (2030 rpm).

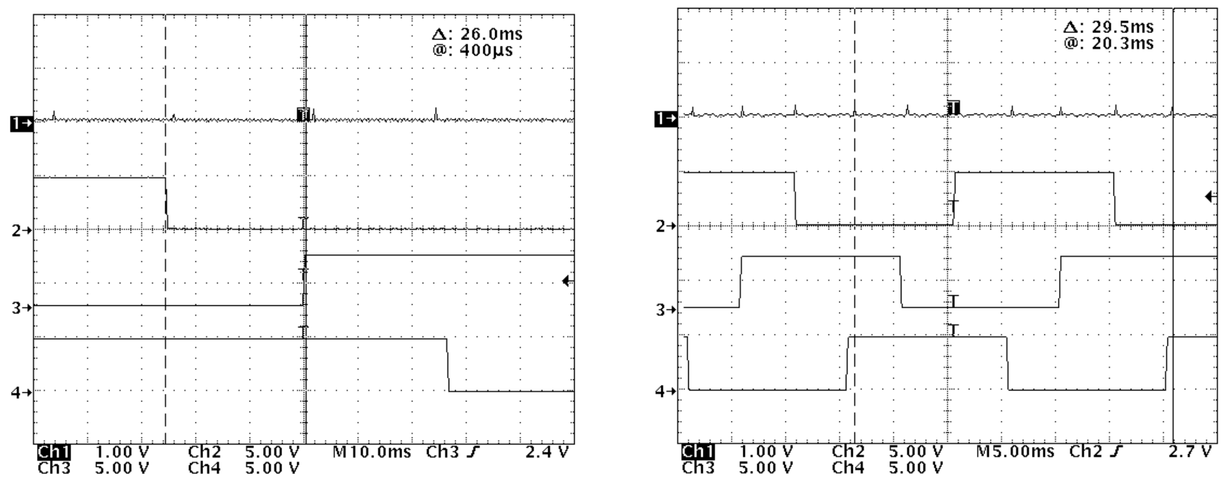


Figure 8: Computed Commutation Instants vs Hall Effect Sensor Outputs

These charts show that very good results can be achieved at low or high speed. A slight shift between the Hall sensor's output and the computed commutation is noticeable; this is compensated by means of a phase use imbalance corrector.

This algorithm for determining the commutation point is based on the time spent by the rotor to make the previous revolution. For noise or system dynamics problems it is possible to design some slightly more complex algorithms: for example a low pass filter on the speed feedback could improve the transient behaviour. It can simply be implemented by saving not only the revolution time at stage k but also the revolution time at stage k-1, then the computation of

$$\frac{T_k - T_{k-1}}{2}$$

gives a smoother change in the shift time, thus improving the transient behaviour. Another possible algorithm which could improve the dynamic behaviour is to compute the time spent by the rotor to achieve one third of a revolution. This last algorithm has a quicker reaction to speed variation but is much more sensitive to measurement noise.

3.2.6 Self Correction of Phase Use Imbalance

Some imbalance in the phases' use might be present when we simply apply the above algorithm (detection of the zero crossing point and waiting for the shift time to elapse before commutating the phases). By 'imbalance' we understand that instead of having six equal 60º sectors per revolution we can have 40º wide and 80º wide sectors. This is depicted below:

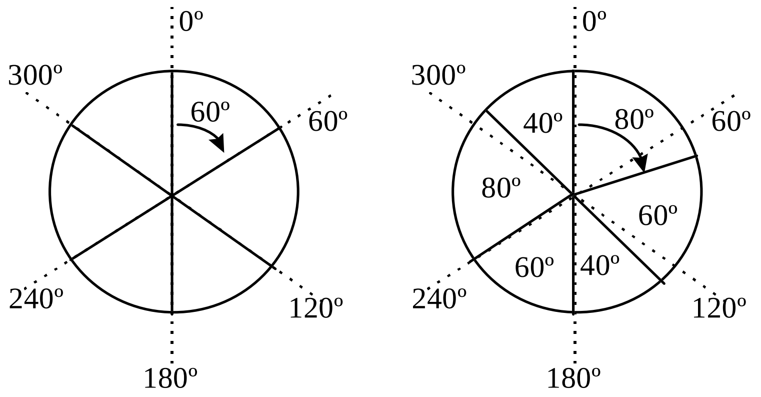


Figure 9: Balanced and Unbalanced Phase Use

This imbalance is created by the asymmetrical behaviour of the three phase system and of the terminal voltage measurement resistor bridges. The imbalance is application dependent but can be approximated as a linear function of the mechanical speed and can be corrected thanks to software offset on the Bemf computation. The following structure solves this problem and is implemented in the code given below.

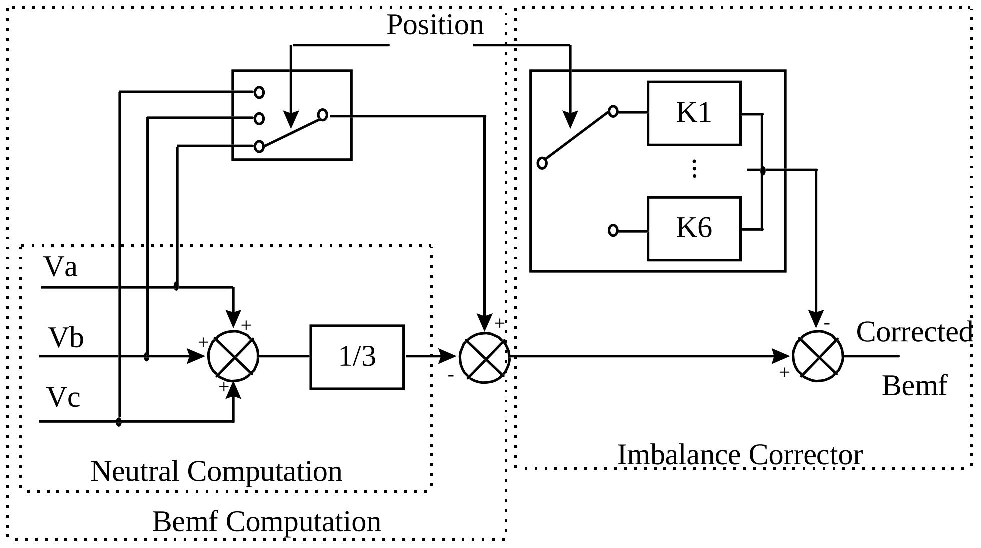
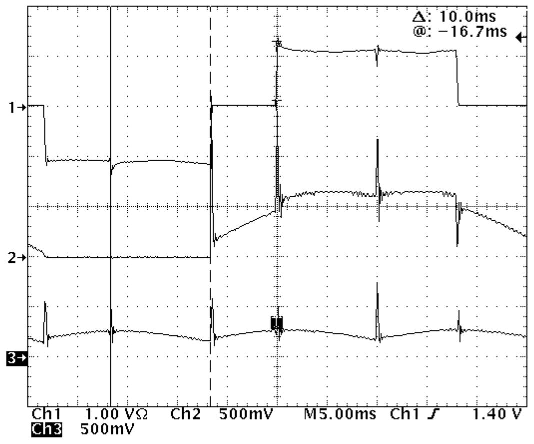


Figure 10: Phase Use Imbalance Correction Module

The two 'scope pictures below show the phase use without (left chart) and with (right chart) this additional correction module.



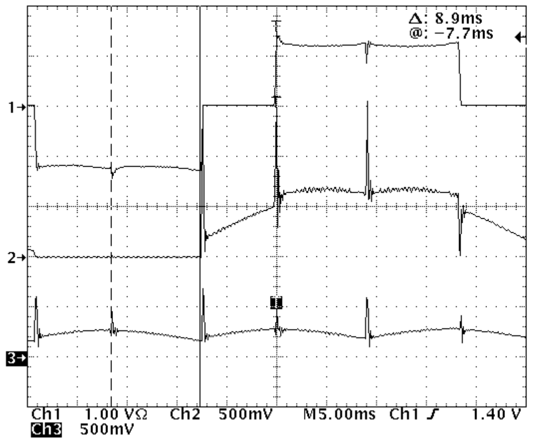


Figure 11: Phase Use without and with Imbalance Corrector

The imbalance corrector code might be improved by setting sector specific coefficients and by correcting the Bemf in each commutation sector (in the code below only one offset is computed and this offset is applied only on three of the six sectors). These charts make obvious the need for Bemf computation correction.

3.2.7 Startup Procedure

This sensorless BLDC drive DSP solution does not need any open loop starting strategy. The following steps are the procedure implemented to start the sensorless BLDC drive. Step 1: the algorithm gets the initial rotor position. Step 2: the shift time algorithm is fed with an *a priori* shift time value. Step 3: the 60º sector adjacent to the initial rotor position is fed with Direct Current and the Bemf zero crossing detection algorithm is immediately started. After one complete revolution the initial shift time (set *a priori*) is replaced by the computed one.

The paragraphs below describe how to give the control software the *a priori* value of the shift time,T, and two ways of getting the initial rotor position. The choice between the two initial position strategies is made according to the motor reluctance characteristics. For a motor with no reluctance variation around the air gap (such as the motor used for this drive) the Rotor Magnetic Stall must be implemented. If the motor shows a reluctance variation around the air gap the reluctance based method should be implemented to ensure the exact initial rotor position determination and to fix the direction of rotation.

3.2.7.1 First Shift Time Calculation

The sensorless control software needs the shift time as a parameter to perform the first revolution. This time can be calculated off line by knowing the motor Torque constant and the braking torque (including friction losses and load) at startup. The fundamental dynamic principal applied to rotational systems:

$$J\frac{d^2\theta}{dt^2} = \sum_i Ti$$

(where J is the system inertia, θ is the angular position and Ti represents torque) gives a double integral equation. To be solved, this equation needs to know system torque and inertia. The torque sum might be considered as constant - and one that we can calculate, as we control the current flowing and as we know the torque constant ([Nm/A]). The system inertia can be calculated from the load characteristics. The solution to this equation is the time necessary for the motor to perform one revolution. This first shift time T is then given by the relation:

$$T = \frac{1}{2}\sqrt{\frac{4J\pi}{\sum_i Ti}}$$

This calculated value should be divided by the Bemf scan loop period and the result should be stored into the software shift time variable.

3.2.7.2 Rotor Magnetic Stall

When there is no reluctance variation around the motor air gap (as in the motor we used in this drive) it might be impossible to estimate the initial rotor position. So at startup is it necessary to energize one arbitrary phase pair and wait for the rotor to be aligned with the stator flux that is created. Nevertheless the algorithm must address the following questions: how long should the algorithm wait for the shaft to stop oscillating prior starting the control loops? Is the applied torque enough to let the load move? In this application a counter is set so that the magnetic stall is performed for long enough to allow the to shaft to move and then stop oscillating. The current flowing in the supplied phase pair is regulated to a value fixed *a priori* at the beginning of the software. To improve this sensorless drive the two questions above might be answered by scanning the non-fed phase and the results can be interpreted as follows: an oscillation of the measured voltages means back and forth movements of the rotor, a constant measured signal means no rotor movement. Such an interpretation leads to an adaptive phase current level and to a time optimized function for obtaining initial rotor position.

3.2.7.3 Reluctance Based Rotor Initial Position

If there is significant reluctance variation around the air gap (salient poles on the rotor…) we can measure each phase pair current response to a voltage pulse applied to each winding for a short and constant period of time. By evaluating the relative magnitude of the current flow through the phase windings it is possible to get the initial rotor position and thus to determine the proper starting phase of the motor.

4. Software Organization

This chapter presents the overall software structure, the variables and the Interrupt Sub Routine flowcharts. This software is based on two modules: the initialization and the run module. The first one is performed only once at the beginning. The second module is the BLDC control dedicated software. It is based on a waiting loop interrupted by both the PWM Unit and the ADC Unit. The waiting loop can easily be replaced by a user's interface (to get the reference speed and/or to monitor the control variables). The overview of this software is given in the flow chart below:

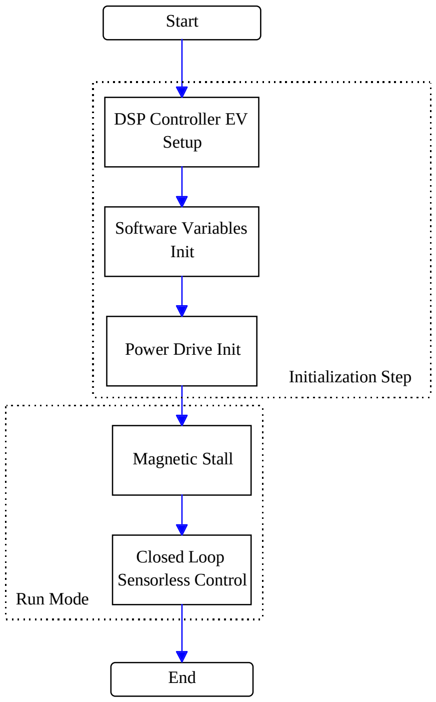


Figure 12: Global Structure of the Sensorless Control Code

The following chapters deal with every module of the global flowchart presented.

4.1 DSP Controller Setup

This part is dedicated to the handling of the different DSP Controller resources (Core settings and peripherals settings). First of all the PLL unit is set so that the CPUCLK runs at 20MHz based on the 10MHz quartz crystal provided on the EVM board. For the development stage it is necessary to disable the watchdog unit: first set the Vccp pin voltage to five volts by means of the EVM jumper,JP5, and then set the two watchdog dedicated registers to be inactive. The last thing to do to complete the DSP Controller core initialization is to set the core mask register correctly. In this sensorless application the interrupts coming from the PWM unit and from the ADC unit are acknowledged.

Now let us have a look at the initialization of the peripherals. To generate the necessary pulsed signals to the power electronics board the Full Compare Unit is used. It is set to generate symmetrical non-complementary PWM signals at the frequency of 80kHz with TIMER1 as time base and with the DEADBAND unit disabled. The PWM resolution is equal to the CPUCLOCK period: 50ns. The new duty cycle reloads are made on the timer counter zero crossing at the frequency of 20kHz. The PWM unit is also set to generate one interrupt when the timer counter reaches the timer 1 period value. Finally, the ADC unit is set to receive the Start Of Conversion from software and to generate an interrupt request to the core when the conversion is finished.

4.2 Software Variables Initialization

The second initialization step is the variable initialization. In this software 19 variables are needed, they are described in this chapter. They can be sorted into five categories: variables which are dedicated to the Bemf scanning function, to the commutation algorithm, to the rotor magnetic stall, to the current regulator and to the speed regulator.

4.2.1 Bemf Scanning function variables

The six variables used for this function are CUR_COUNT, V1, V2, V3, NEUTRAL, OFFSET and FLAG. All of these variables are initialized to zero.

CUR_COUNT is used in the PWM period interrupt as a PWM interrupt counter. It is used in order to know whether or not a conversion should be started (cf figure 6).

V1, V2 and V3 are used in the ADC interrupt to save the respective phase terminal converted voltages.

NEUTRAL is also used in the ADC interrupt to store the computed neutral voltage. As it takes a shorter time to perform a multiplication than a division, NEUTRAL simply contains the sum of the three phase voltages rather than one third of the sum of the three phase voltages. The Bemf computation is performed by subtracting Neutral from three times the desired phase voltage. Fifteen CPU cycles are saved in this way.

OFFSET is computed in the speed control loop but is used in the SEQUENCE function. OFFSET is the single Bemf correction computed in this software; it is subtracted from the Bemf before comparing the Bemf to zero.

FLAG is set to one when the Bemf changes sign. When this flag is equal to one, the sign change detection is no longer performed until the next phase pair has been commutated. This is to make sure that there will not be any ripple in the sign change detection, thus making it possible to achieve very precise commutation angles.

4.2.2 Commutation Algorithm Variables

The variables used to compute the phase commutation point are ASYM, CAPT, B2COUNT. They are all initialized to zero.

The current 60º sector is stored in the CAPT variable. As CAPT is used in the SEQUENCE function to branch to the code which gives the right ACTR register setting, the values that it can take are 0, 2, 4, 6, 8, 10, 12 (corresponding respectively to sector #1, 2, 3, 4, 5, 6).

ASYM is used as a counter in the SEQUENCE function. This counter is reset to zero every time a phase commutation occurs. It increments at each new current regulation loop (in other words at each new neutral point voltage computation or, that is, every 50µs) and up to the time it reaches the predefined value. Before reaching the predefined value the computed Bemf is not taken into account. This counter function filters the imbalance of the three phase system created by the commutation. The predefined value is application dependent (power board design, direct current driven) and should be adjusted to each new drive.

B2COUNT is used as a counter in the ADC interrupt. As soon as the Bemf crosses zero this counter is loaded with the computed shift time. This counter, once loaded, decrements every 50µs and when it reaches zero the phases are commutated.

4.2.3 Rotor Magnetic Stall Variables

The variables used in this function are SPEEDFLAG, BCOUNT and STALL.

BCOUNT is used in the ADC interrupt during the rotor magnetic stall as a counter. This counter increments every 50µs (every current regulation loop) until a preset value. This preset value defines how many times the software has to stall the rotor in order to let the shaft stop oscillating prior starting the closed loop control.

Once the BCOUNT counter reaches the predefined value, the STALL variable is set to one. This variable is used as a flag which enables the closed loop control when set to one.

SPEEDFLAG is a flag variable which (when set to one) disables any speed regulation during the first revolution. This implies that the motor runs the first revolution with a regulated constant torque. This flag is reset to zero when the first revolution has been completed thus allowing the speed control to run in closed loop.

4.2.4 Current Loop Variables

The variables used for this function are Idc_ref, Idc_errorK, COMP and FLAGCUR. These variables are initialized to zero apart from COMP, which is set equal to the PWM period.

Idc_ref is the variable used to store the direct current reference given by the speed regulation. Idc_errorK is used to store the current error before using it. COMP is the output of the current regulator (in other words this is the updated duty cycle).

FLAGCUR is the flag used to tell the background waiting loop that it is time to give the inverter the new ACTR settings and the new duty cycle. After completion of these tasks FLAGCUR is reset to zero in the SEQUENCE function.

4.2.5 Speed Loop Variables

The variables used here are SPEED_REF, SPEED_COUNT, BCOUNT, FLAGUP and the auxiliary register AR3.

SPEED_COUNT is used in the ADC interrupt to know if it is time to update the speed regulation loop. SPEED_REF is used to store the desired speed value.

FLAGUP is initialized to zero and is set to one once a mechanical revolution has been completed. If FLAGUP is equal to one, then the revolution time is divided by 12 (one twelfth of a revolution is a 30º shift angle) and the result is stored into BCOUNT. So, during closed loop control, BCOUNT is the variable which contains the shift time (this shift time will be loaded into B2COUNT after any BEMF zero crossing).

Every 50µs the auxiliary register AR3 is incremented at the beginning of the SEQUENCE function. So after one complete revolution AR3 register contains the previous revolution time. The largest value that AR3 might contain is #0FFFFh (65535). Multiplying by 50µs gives 3.27s (the time to complete one revolution). So in this software the lowest speed reachable is 19rpm.

4.3 Interrupts Flow Charts

The time diagram of the incoming interrupts is given by figure 6. The forthcoming chapters present the interrupt internal organizations. Some software features are realized in the interrupt subroutines and some are performed outside. Interrupt operations performed inside are: the starts of conversion, current regulation, neutral point voltage computation, speed update calculation and phase commutation. Interrupt operations performed outside are: Bemf computation, Bemf sign change detection, three phase system asymmetry correction and PWM output. Calling the outside functions is ensured by flags that are polled in the waiting loop and set inside the interrupts.

4.3.1 PWM Unit Interrupt Flowchart

The interrupt request is generated every PWM period (cf figure 6, note that the PWM period interrupt is NOT the PWM Timer period interrupt). The flowchart below depicts the ISR organization. The PWM period interrupt lasts a maximum of 30 CPU cycles (1.5µs) and thus needs 2.4MIPS among the 20 available. With some changes the scheme below can be used with another chopping frequency: let us assume a 20kHz chopping frequency and a 10kHz current loop frequency. In this case a conversion should be started at every PWM period. So there is no longer any need to acknowledge the PWM interrupt request and the Start Of Conversion should be given automatically by the DSP Controller Event Manager. This would spare some more CPU power.

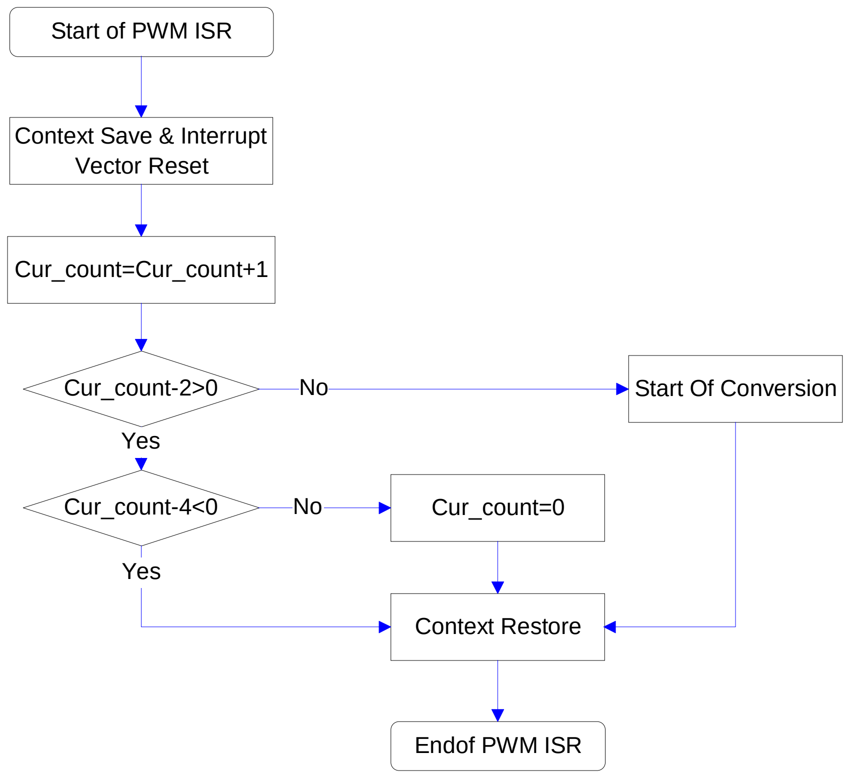


Figure 13: PWM Interrupt Flowchart

4.3.2 *ADC Unit Interrupt Flowchart*

The interrupt request is generated when the End Of Conversion signal reaches the DSP Controller core. The flowchart below depicts what operations are performed in this interrupt.

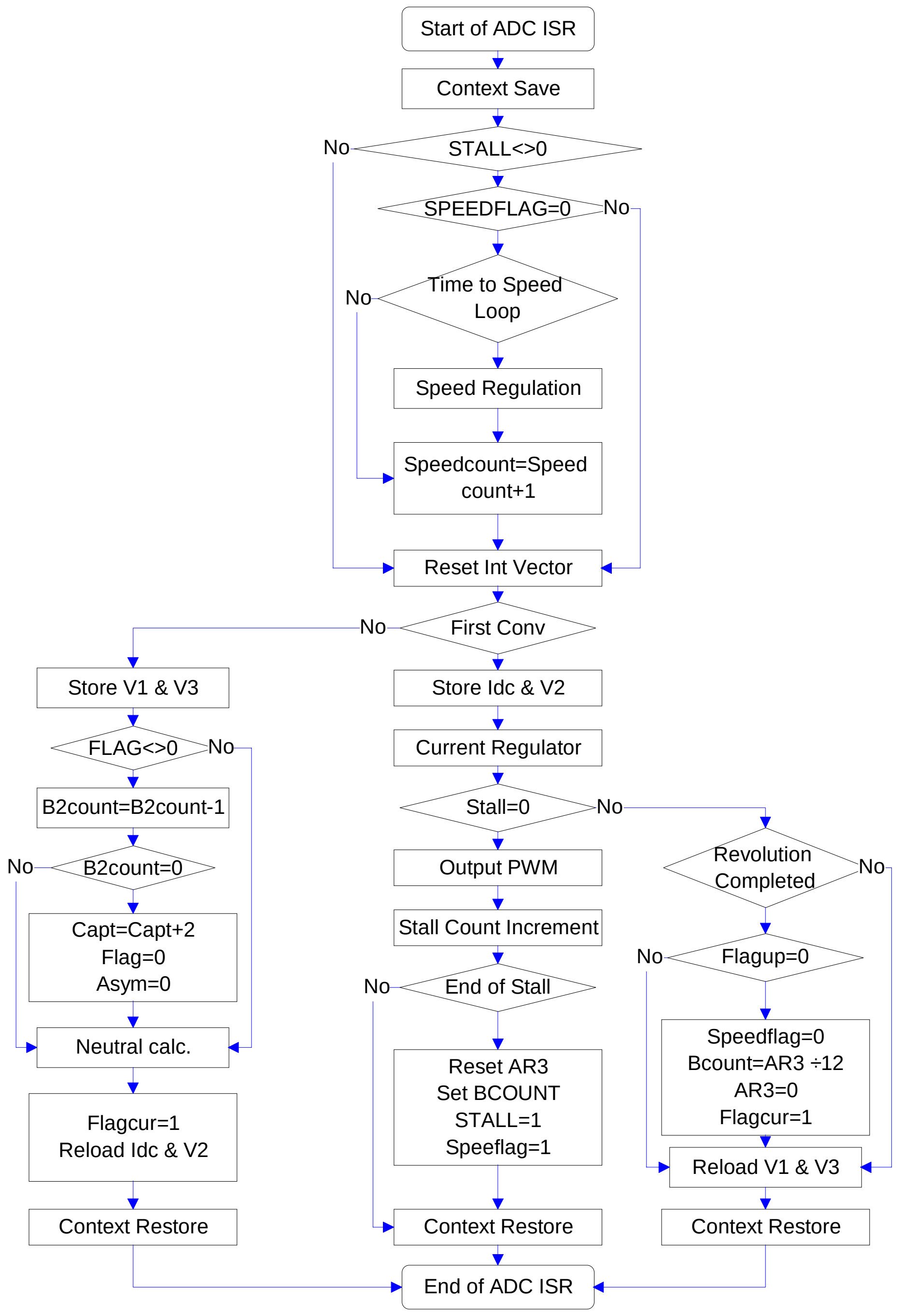


Figure 14: ADC Interrupt Flowchart

What is missing at this point is the Bemf sign change detection as well as the output of PWM in the running mode. This is covered in the following chapter.

4.4 SEQUENCE Function Flow Chart

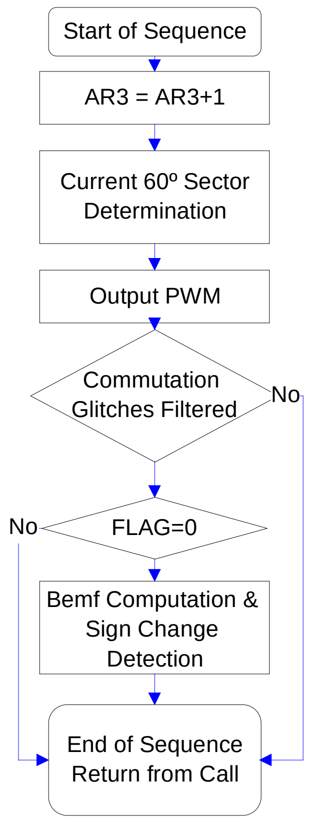

Start of Sequence
AR3 = AR3+1
Current 60º Sector Determination
Output PWM
Commutation Glitches Filtered
No
FLAG=0
No
Bemf Computation & Sign Change Detection
End of Sequence Return from Call

Figure 15: SEQUENCE Function Flowchart

The SEQUENCE is called by the waiting loop when Flagcur variable has been set to one (in the ADC interrupt). This allows the ADC interrupt not to be too long, particularly at a high chopping frequency. The ADC interrupt occurs 6.6µs after the PWM period interrupt and the PWM period is equal to 12.5µs. If the sequence function were called from the ADC ISR there would be insufficient time to acknowledge the subsequent PWM period interrupt at the right point.

5. Summary

This document deals with obtaining the rotor position of a sensorless Brush Less DC drive by means of software. This single chip solution is based entirely on the TMS320x24x DSP Controller capabilities: namely, high 2xx DSP core CPU power and extremely versatile peripherals. This solution replaces the position sensor cost at no cost other than software development. Furthermore, this solution increases the drive reliability as the position information will no longer be sensitive to temperature or vibration. This makes the sensorless solution really cost effective and reliable. Assuming that the initial rotor position is detectable and knowing that this solution runs closed loop from quasi zero speed, it is apparent that this sensorless drive might advantageously replace the sensored solution in a wide range of speed control applications.

References

1. TMS320C24x DSP Controllers - Reference Set: Vol.1, Texas Instruments Inc, 1997.

2. TMS320C24x DSP Controllers - Reference Set: Vol.2 Texas Instruments Inc, 1997.

3. Implementation of a Speed Controlled Brushless DC Drive using TMS320F240, Texas Instruments Inc., 1997 part #BPRA064.

4. DSP Solutions for BLDC Motors, Texas Instruments Inc.,1997 part #BPRA055.

5. DC 12-24V 30A Three Phase Power Hardware for either PMSM or AC Induction Machine, Texas Instruments Inc.,1997 part #BPRA071.

Appendix A Sensorless BLDC Drive Control Assembly Code

```
;*********************************************************
; File Name    : sensor.asm
;
; Target System       : c240 evm
;
; Description  : BLDC motor speed controlled
;                40W 18V July/97
;                Sensorless Speed control based on
;                Bemf measurement.
;                Magnetic Stall at start up.
;                Phases’ use imbalance Correction
;                Commutation dV/dt and dI/dT filtering
;
;Date          September 1997
;**********************************************************

               .include       "c240app.h"
;-------------------------------------------
;Current regulator coeff setting
;-------------------------------------------
Kp             .set    280     ;Q11 (1=2048) Kp=0.12

;-------------------------------------------
;Speed regulator coeff setting
;-------------------------------------------
Kps            .set    100
Kis            .set    100
;-------------------------------------------
; Variable definitions
;-------------------------------------------
               .bss    CAPT,1         ; capt indication
               .bss            COMP,1
               .bss    Idc_ref,1
               .bss    Idc_errorK,1
               .bss    FLAGCUR,1
               .bss    CUR_COUNT,1
               .bss    SPEED_REF,1
               .bss    SPEED_COUNT,1
               .bss    V1,1
               .bss    V2,1
               .bss    V3,1
               .bss    NEUTRAL,1
               .bss    FLAG,1
               .bss    FLAGUP,1
               .bss    BCOUNT,1
               .bss    B2COUNT,1
               .bss    STALL,1
               .bss    ASYM,1
               .bss    SPEEDFLAG,1
               .bss    OFFSET,1
               .bss    stack,6

;======================================
; Reset & interrupt vectors
;======================================
               .sect   "vectors"
RSVECT         B       _c_int0
               .space  16*2
INT2           B       PWMINT         ;Assign PWM interrupt vector
```

```
               .space  16*6
INT6           B       ADCINT         ;Assign ADC interrupt vector
               .space  16*38



               .text
               .global _c_int0
_c_int0
               SETC    CNF
               CLRC    OVM            ;Reset overflow mode
               SETC    SXM            ;Reset sign extension mode
               CLRC    XF             ;Reset debugging pin
               SETC    INTM           ;Set global interrupt mask

               MAR     *,AR2          ;Auxiliary Registers Init
               LAR     AR2,#0300h
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               SPLK    #0,*+
               LAR     AR2,#0300h
               LAR     AR1,#stack
               LAR     AR3,#0307h

       ;Disable watchdog (Vccp=5v)& watchdog counter reset p6-12
               LDP     #00E0h
               SPLK    #0006Fh, WD_CNTL
               SPLK    #05555h, WD_KEY
               SPLK    #0AAAAh, WD_KEY

       ; initialize WDT registers
               SPLK #06Fh, WD_CNTL     ; clear WDFLAG, Disable WDT, set WDT for 1

       ; Set up CLKOUT to be SYSCLK p6-6
               SPLK    #40C0h,SYSCR
               LACC    SYSSR
               AND             #069FFh
               SACL            SYSSR

       ;set up PLL clockin=10Mhz,CPUCLOCK=20Mhz,SYSCLK=10Mhz
               SPLK    #0002h,CKCR0   ; PLL disabled
               SPLK    #00b1h,CKCR1
               SPLK    #0081h,CKCR0
       ;Set up one Wait States for I/O space (DAC needs one wait state)
               MAR     *,AR1
               LACC    #0004h
               SACL    *
               OUT     *,WSGR

       ;I/O setting p11-11
               LDP     #00E1h
```

```
        SPLK    #0000fh, OCRA
        SPLK    #0070h, OCRB
        SPLK    #02800h, PBDATDIR        ;Clear Fault and no Shut Down
        LACC    PBDATDIR

; Clear EV control registers
        LDP     #0E8h
        SPLK    #0000h,T1CON   ;no timer enable
        SPLK    #0000h,T1PER   ;no timer features enable
        SPLK    #0000h,T1CNT
        SPLK    #0000h,T1CMP
        SPLK    #0000h,T2CON
        SPLK    #0000h,T2PER
        SPLK    #0000h,T2CNT
        SPLK    #0000h,T2CMP
        SPLK    #0000h,T3CON
        SPLK    #0000h,T3PER
        SPLK    #0000h,T3CNT
        SPLK    #0000h,T3CMP
        SPLK    #0000h,COMCON
        SPLK    #0000h,DBTCON
        SPLK    #0000h,ACTR
        SPLK    #0000h,SACTR
        SPLK    #0000h,CMPR1
        SPLK    #0000h,CMPR2
        SPLK    #0000h,CMPR3
        SPLK    #0000h,SCMPR1
        SPLK    #0000h,SCMPR2
        SPLK    #0000h,SCMPR3
        SPLK    #0000h,CAPCON  ;no capture enable
        SPLK    #00ffh,CAPFIFO
        LACC    FIFO1
        LACC    FIFO2
        LACC    FIFO3

;PWM Unit setting
        SPLK    #0125,T1PER
        SPLK    #0000h,T1CNT
        SPLK    #0FFFh,ACTR
        SPLK    #0508h,DBTCON
        SPLK    #00125,CMPR1
        SPLK    #00125,CMPR2
        SPLK    #00125,CMPR3
        SPLK    #0287h,COMCON
        SPLK    #8287h,COMCON
        SPLK    #2800h,T1CON
        SPLK    #2840h,T1CON
        SPLK    #0000h,GPTCON

;Capture Unit Setting
        SPLK    #0b0fch,CAPCON
        SPLK    #00ffh,CAPFIFO

;Core Mask Setting
        LDP     #0
        LACC    #022H   ;ADC and Group A Interrupt Core Mask
        SACL    IMR
        LACC    IFR     ;Clear Core Flag Register
        SACL    IFR

;EV Mask Setting, Vector & Flag reset p11-46
        LDP     #0E8h
```

```
               LACC    IFRA
               SACL            IFRA
               LACC    IFRB
               SACL            IFRB
               LACC    IFRC
               SACL            IFRC
               SPLK    #0200H,IMRA
               SPLK            #0,IMRB
               SPLK            #7,IMRC
               LACC    IVRA
               LACC            IVRB
               LACC            IVRC

       ;ADC Unit setting p3-8
               LDP     #0E0h
               SPLK    #0003h,ADCTRL2
               SPLK            #1b6ah,ADCTRL1 ;ADC 14&5

               CLRC    INTM

               LDP     #0
               SPLK    #020H,Idc_ref          ;Magnetic Stall Desired Current
               SPLK    #0,Idc_errorK
               SPLK    #0300H,SPEED_REF       ;Speed Reference
               SPLK    #00112,COMP            ;Minimum Duty Cycle
               SPLK    #0000,CUR_COUNT
               SPLK    #0000,FLAGCUR
               SPLK    #0000,SPEED_COUNT
               SPLK    #0000H,CAPT
               SPLK    #0000H,V1
               SPLK    #0000H,V2
               SPLK    #0000H,V3
               SPLK    #0000H,NEUTRAL
               SPLK    #0000H,FLAG
               SPLK    #0001H,FLAGUP
               SPLK    #0000H,BCOUNT
               SPLK    #0000H,B2COUNT
               SPLK    #0000H,STALL
               SPLK    #0000H,ASYM
               SPLK    #0000H,OFFSET

               MAR     *,AR2
               LAR     AR2,#0300h


FAULT_CLEAR
               LDP     #0E1h                  ;Check if the Pre Driver Signal
               LACC    PBDATDIR               ;Is Cleared
               AND     #010h
               BZ      FAULT_CLEAR
               SPLK    #02820h,PBDATDIR       ;If cleared then stop Clear Fault

               LDP     #0
               LACC    COMP                   ;Load Duty Cycle and Set
               LDP     #0E8h                  ;PWM Register for the Magnetic
               SPLK    #03FDh,ACTR            ;Stall Function
               SACL    CMPR1
               SPLK    #0FFFH,CMPR2
               SPLK    #0FFFH,CMPR3

MAGSTALL
               LDP     #0                     ;Check if the Magnetic Stall is
               LACC    STALL                  ;Terminated
               BZ      MAGSTALL

               ;Set next commutation sequence
               LACC    COMP                   ;If so then set the ACTR Register
               LDP     #0E8h                  ;to commutate the next phase pair.
               SPLK    #03DFh,ACTR
               SACL    CMPR2
               SPLK    #0FFFFH,CMPR3
               SPLK    #0FFFFH,CMPR1

               LDP     #0             ;Trace of the current 60º sector
               SPLK    #4,CAPT

LOOP
               LDP     #0
               LACC    FLAGCUR
               BZ      LOOP           ;Time to update the Duty Cycle?
               SPLK    #0,FLAGCUR
               CALL    SEQUENCE       ;If yes the Call Sequence function
               B       LOOP           ;Then wait for the next updated Duty Cycle

SEQUENCE
               MAR     *,AR3`         ;Revolution Time Counter
               LAR     AR3,#0307H
               LACC    *
               ADD     #1
               SACL    *

               LDP     #0             ;Which CMPR register should be updated
               LACC    CAPT           ;with the new duty cycle?
               ADD     #CAPT_DETER
               BACC
CAPT_DETER
               B       RISING1
               B       FALLING3
               B       RISING2
               B       FALLING1
               B       RISING3

FALLING2
               LACC    COMP           ;Input Current Path:   Phase C
               LDP     #0E8h          ;Output Current Path:  Phase B
               SPLK    #0D3FH,ACTR    ;Non Fed Phase:               Phase A
               SACL    CMPR3
               SPLK    #0FFFH,CMPR2
               SPLK    #0FFFH,CMPR1

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero already crossed?
               LDP     #0
               LACC    FLAG           ;Did BemfA sign already changed?
               BNZ     END            ;If yes then END
                                      ;Else:
```

```
               ;Bemf zero crossing detection
               LDP     #0
               LACC    V1,1
               ADD     V1             ;ACC=3*(BemfA + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfA
               SUB     OFFSET         ;ACC=3*BemfA corrected
               BLZ     END            ;Sign Change?
               SPLK    #1,FLAG        ;BemfA Sign has changed
               LACC    BCOUNT         ;Load Shift Time.
               SACL    B2COUNT
               B       END

RISING3
               LACC    COMP           ;Input Current Path           Phase C
               LDP     #0E8h          ;Output Current Path  Phase A
               SPLK    #0DF3H,ACTR    ;Non Fed Phase        phase B
               SACL    CMPR3
               SPLK    #0FFFH,CMPR2
               SPLK    #0FFFH,CMPR1

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero crossed?
               LDP     #0
               LACC    FLAG           ;Did bemfB sign already changed?
               BNZ     END            ;If yes then END
                                      ;Else
               ;Bemf zero detection
               LACC    V2,1
               ADD     V2             ;ACC=3*(BemfB + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfB
               BGEZ    END            ;Sign Changed?
               SPLK    #1,FLAG        ;Bemf Sign Has Changed
               LACC    BCOUNT         ;Load Shift Time
               SACL    B2COUNT
               B       END

FALLING3
               LACC    COMP           ;Input Current Path           Phase A
               LDP     #0E8h          ;Output Current Path  Phase C
               SPLK    #03FDH,ACTR    ;Non Fed Phase        Phase B
               SACL    CMPR1
               SPLK    #0FFFH,CMPR2
               SPLK    #0FFFH,CMPR3

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero crossed?
               LDP     #0
               LACC    FLAG           ; Did BemfB sign changed already?
               BNZ     END            ;If yes then END
                                      ;Else:
               ;Bemf zero detection
               LACC    V2,1
               ADD     V2             ;ACC=3*(BemfB + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfB
               SUB     OFFSET         ;ACC=3*BemfB corrected
               BLZ     END            ;Sign Changed?
               SPLK    #1,FLAG        ;Bemf sign has changed
               LACC    BCOUNT         ;Load Shift Time
               SACL    B2COUNT
               B       END

RISING2
               LACC    COMP           ;Input Current Path:          Phase B
               LDP     #0E8h          ;Output Current Path:  Phase C
               SPLK    #03DFH,ACTR    ;Non fed phase         Phase A
               SACL    CMPR2
               SPLK    #0FFFH,CMPR3
               SPLK    #0FFFH,CMPR1

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero crossed?
               LDP     #0
               LACC    FLAG           ;Did BemfA sign already changed?
               BNZ     END            ;If yes then END
                                      ;Else:
               ;Bemf zero detection
               LACC    V1,1
               ADD     V1             ;ACC=3*(BemfA + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfA
               BGEZ    END            ;Sign Changed?
               SPLK    #1,FLAG        ;BemfA Sign has Changed
               LACC    BCOUNT         ;Load Shift Time
               SACL    B2COUNT
               B       END

RISING1
               LACC    COMP           ;Input Current Path           Phase A
               LDP     #0E8h          ;Output Current Path   Phase B
               SPLK    #0F3DH,ACTR    ;Non fed phase         Phase C
               SACL    CMPR1
               SPLK    #0FFFH,CMPR2
               SPLK    #0FFFH,CMPR3

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero crossed?
               LDP     #0
               LACC    FLAG           ;Did BemfC sign already changed?
               BNZ     END            ;If yes then END
                                      ;Else:
               ;Bemf zero detection
               LDP     #0
               LACC    V3,1
               ADD     V3             ;ACC=3*(BemfC + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfC
               BGEZ    END            ;Sign Changed?
               SPLK    #1,FLAG        ;BemfC Sign has changed
               LACC    BCOUNT         ;Load Shift Time
               SACL    B2COUNT
               B       END

FALLING1
               LACC    COMP           ;Input Current Path          Phase B
               LDP     #0E8h          ;Output Current Path  Phase A
               SPLK    #0FD3H,ACTR    ;Non fed phase        Phase C
               SACL    CMPR2
               SPLK    #0FFFH,CMPR3
               SPLK    #0FFFH,CMPR1

               ;Commutation glitches filter
               LDP     #0
               LACC    ASYM
               ADD     #1
               SACL    ASYM
               SUB     #10
               BLEZ    END
               SPLK    #10,ASYM

               ;Zero crossed?
               LDP     #0
               SPLK    #0,FLAGUP
               LACC    FLAG           ; Did BemfC sign already changed?
               BNZ     END            ;If Yes then END
                                      ;Else
               ;Bemf zero detection
               LDP     #0
               LACC    V3,1
               ADD     V3             ;ACC=3*(BemfC + Neutral)
               SUB     NEUTRAL ;ACC=3*BemfC
               SUB     OFFSET         ;ACC=3*Bemf3 corrected
               BLZ     END            ;Sign Changed?
               SPLK    #1,FLAG        ;BemfC Sign has changed
               LACC    BCOUNT         ;Load Shift Time
               SACL    B2COUNT
END
               RET


SPEED_REG
               MAR     *,AR2
               LAR     AR2,#0303h             ;Speed error Pointer Init
               LDP     #0
               SPLK    #32,SPEED_COUNT        ;Speed Error Shifter setting

               ;Speed and speed error calc
               CLRC    SXM
               ZAC
               OR      #0FFFFH
               RPT     #15
               SUBC    BCOUNT
               AND     #0FFFFH                ;Acc=1 / (rev. time/ 12) = Speed
               SETC    SXM
               SUB     SPEED_REF
               NEG                            ;Acc= Spd ref-Spd Fb= Spd error

               ;Speed error limitation
               BGEZ    POS
               ABS
               SPLK    #-32,SPEED_COUNT
POS
               SACL    *
               SUB     #03FFH
               BLEZ    OKPOS
               SPLK    #03FFH,*
OKPOS
               LT      *                      ;-1024 < Speed error < 1024
               MPY     SPEED_COUNT
               PAC
               SACL    *                              ;Speed error <<5

               ;Speed regulation
               LT      *
               MPY     #Kps
               PAC
               ADD     Idc_ref,16
               SACH    Idc_ref                ;Idc_ref(k)=Idc_ref(k-1) + K*Speed_error(k)

               ;Idc_ref compensation
               LACC    Idc_ref
               BGEZ    RES
               SPLK    #0,Idc_ref

               ;Reset Speed loop timer
RES
               LACC    SPEED_REF,11           ;Bemf correction computation
               SACH    OFFSET
               SPLK    #0,SPEED_COUNT
               RET

PWMINT
               ;save status registers
               MAR     *,AR1
               MAR     *+                             ; skip one position
               SST     #1, *+                         ; save ST1
               SST             #0, *+                         ;save ST0
               SACH    *+                             ;save acc high
               SACL    *                      ;save acc low

               LDP     #0E8H                  ;Clear Interrupt Vector
               LACC    IVRA

               LDP     #0
               LACC    CUR_COUNT
               ADD     #1
               SACL    CUR_COUNT
               SUB     #2
               BGZ     CONV                   ;Time to start a Conversion?
               LDP     #0E0H
               LACC    ADCTRL1
               OR      #02000H
               SACL    ADCTRL1
               B       CONTEXT

CONV
               SUB     #2                    ;Reset CUR_COUNT variable?
               BLZ     CONTEXT
               SPLK    #0,CUR_COUNT
CONTEXT
               ;restore status registers
               MAR     *, AR1                        ; make stack pointer active
               LACL    *-                    ;Restore Acc low
               ADDH    *-                    ;Restore Acc high
               LST     #0, *-                        ; load ST0
               LST             #1, *-                        ; load ST1

               CLRC    INTM
               RET

ADCINT
               MAR     *,AR1
               MAR     *+                            ; skip one position
               SST     #1, *+                        ; save ST1
               SST     #0, *+                        ; save ST0
               SACH    *+                            ;save acc high
               SACL    *                             ;save acc low

               ;Magnetic stall flag
               LDP     #0
               LACC    STALL
               BZ      Vdc_or_Idc     ;If Magnetic Stall is cur. Performed then
                                      ;Branch to Vdc_or_Idc
               LACC    SPEEDFLAG      ;If the first Revolution has not been
               BNZ     Vdc_or_Idc     ;Completed then branch to Vdc_or_Idc

               ;Time to speed loop?
               LACC    SPEED_COUNT
               SUB     #2000
               BNZ     NO_SPEED_REG
               CALL    SPEED_REG      ;If Time then Speed Loop
NO_SPEED_REG
               LACC    SPEED_COUNT
               ADD     #1
               SACL    SPEED_COUNT


               ;Vdc OR Idc?
Vdc_or_Idc
               LDP     #0E0h
               LACC    SYSIVR         ;Clear interrupt vector

               LDP     #0E0H
               LACC    ADCTRL1
               AND     #0002H
               BZ      Vdc            ;If second conv then branch to Vdc

               ;current error calculation
               CLRC    SXM
               LACC    ADCFIFO1,10    ;high Acc = Idc
               ADD     ADCFIFO2       ;low Acc = V2<<6
               LDP     #0
               SACH    Idc_errorK     ;Store Idc
               SACL    V2             ;Store V2<<6
               LACC    V2,10
               SACH    V2             ;Store V2
               SETC    SXM

               LACC    Idc_errorK,5
               SUB     Idc_ref,5
               SACL    Idc_errorK     ;Store Current error<<5

               ;Current regulation
               LT      Idc_errorK
               MPY     #Kp
               PAC                            ;Acc = Kp*Current_error
               ADD     COMP,16
               SACH    COMP                   ;Store Kp*Current_error + COMP(k-1)
               LACC    COMP                   ;Load updated Duty cycle

               ;current reg output limitation
               BGZ     SUP_LIM
               SPLK    #0,COMP
               B       COMP_OK
SUP_LIM
               SUB     #0112
               BLZ     COMP_OK
               SPLK    #0112,COMP
COMP_OK                               ;0 < Updated Duty cycle < Max duty
               LACC    STALL
               BNZ     SPEEDUP        ;If running mode Branch to SPEEDUP
               CLRC    SXM                    ;Else (Magnetic Stall Performing):
               SPLK    #2,CUR_COUNT           ;Cancel second conversion start
               LACC    COMP
               LDP     #0E8H
               SACL    CMPR1                  ;Output Updated Magnetic Stall Duty Cycle
               LDP     #0
               LACC    BCOUNT                 ;Magnetic Stall Counter Increment
               ADD     #1
               SACL    BCOUNT
               SUB     #0FFFFH
               SETC    SXM
               BNZ     RESTO                  ;Is magnetic stall to the end?
               MAR     *,AR3                          ;If yes then
               LAR     AR3,#0307H
               SPLK    #0,*
               SPLK    #050H,BCOUNT           ;Load first Time Shift
               SPLK    #1,STALL                       ;Enable running mode
               SPLK    #1,SPEEDFLAG
               B       RESTO

               ;Update speed?
SPEEDUP
               LACC    CAPT
               SUB     #4
               BNZ     RELOAD                 ;Time to update speed feedback?
               LACC    FLAGUP                 ;Speed feedback already updated?
               BNZ     RELOAD                 ;If yes branch to RELOAD
               MAR     *,AR3
               LAR     AR3,#0307H
               SPLK    #0,SPEEDFLAG           ;Release the first revolution indicator flag
               CLRC    SXM
               LACC    *                      ;Load the revolution time
               SPLK    #012,BCOUNT
               RPT     #15
               SUBC    BCOUNT                ;Divide it by 12 (i.e. 30º)
               AND     #0FFFFH
               SACL    BCOUNT                ;Store the new Shift Time
               SETC    SXM
               SPLK    #0,*
               SPLK    #1,FLAGUP             ;Speed feedback updated

               ;Reload Vdc channel
RELOAD
               LDP     #0E0H
               SPLK            #1B5Ch,ADCTRL1 ;ADC 6&13

               ;restore context
RESTO
               MAR             *, AR1                         ; make stack pointer active
               LACL    *-
               ADDH    *-
               LST             #0, *-                         ; load ST0
               LST             #1, *-                         ; load ST1

               CLRC    INTM
               RET

Vdc
               ;Read Vshunt values
               LDP     #0E0H
               CLRC    SXM
               LACC    ADCFIFO1,10           ;Acc high = V3
               ADD     ADCFIFO2              ;Acc low = V1<<6
               LDP     #0
               SACH    V3                    ;Store V3
               SACL    V1                    ;Store V1<<6
               LACC    V1,10
               SACH    V1             ;Store V1

               ;Zero crossed?
               LACC    FLAG           ;Did the Bemf already crossed zero?
               BZ      NEU            ;If Not then Branch to NEU

               ;Commutation instant
               LACC    B2COUNT ;If yes decrement shift time counter
               SUB     #1
               SACL    B2COUNT ;Store new shift time counter value
               SETC    SXM
               BNZ     NEU            ;Is it time to commutate a new phase pair?
               LACC    CAPT           ;If yes then upgrade the commutated phases
               ADD     #2             ;remainder
               SACL    CAPT           ;Store the commutated phases remainder
               SUB     #0CH
               BNZ     OKCAPT
               SPLK    #0,CAPT
OKCAPT
               SPLK    #0,FLAG        ;Reset flags for the new Bemf scanning
               SPLK    #0,ASYM

               ;Neutral calculation
NEU
               LACC    V1
               ADD     V2
               ADD     V3             ;Acc=3*neutral point voltage
```

```
SACL    NEUTRAL ;Store 3*neutral point voltage

;output PWM
SPLK    #1,FLAGCUR     ;Set flag to let the PWM unit to be updated

;Reload Idc channel
LDP     #0E0H
SPLK           #1b6ah,ADCTRL1 ;Loaded ADC lines: Idc and V2

;restore context
MAR     *, AR1         ; make stack pointer active
LACL    *-
ADDH    *-
LST     #0, *-         ; load ST0
LST     #1, *-         ; load ST1

CLRC    INTM
RET
```

Appendix B Linker File

```
                                        /*------------------------------------------------------------------------------------------*/
                                        /*  LINKER COMMAND FILE - MEMORY SPECIFICATION for C240 */
                                        /*  Last update 24 Sep 96                                                  */
                                        /*------------------------------------------------------------------------------------------*/

MEMORY
{
   PAGE 0 :        VECS    : origin =    0h , length =   040h         /* PROGRAM */
                   PROG    : origin =   40h , length =  0800h         /* Ext mem */

   PAGE 1 :        MMRS    : origin =    0h , length =   05Fh         /* MMRS    */
                   B2      : origin = 0060h , length =   020h         /* DARAM   */
                   B0      : origin = 0100h , length =  0200h         /* DARAM   */
                   B1      : origin = 0300h , length =  0200h         /* DARAM   */
}


                                           /*---------------------------------------------------------------------------------*/
                                            /* SECTIONS ALLOCATION                                                  */
                                           /*---------------------------------------------------------------------------------*/
SECTIONS
{
   .vectors : { } > VECS      PAGE 0         /* INTERRUPT VECTOR TABLE        */
   .text    : { } > PROG      PAGE 0         /* CODE                          */
   .mmrs    : { } > MMRS      PAGE 1         /* Memory Mapped Registers       */
   .blk0    : { } > B0        PAGE 1         /* Block B0 - page 4         */
   .bss     : { } > B2        PAGE 1         /* Block B2 - page 0         */
   .blk1    : { } > B1        PAGE 1         /* Block B1 - page ?         */

}
```