Order by AN1930/D
(Motorola Order Number)
Rev. 0, 10/02

3-Phase AC Induction Motor Vector Control Using DSP56F80x

Design of Motor Control Application Based on Motorola's Software Development Kit

Jaroslav Lepka, Petr Stekl

Contents

1. Introduction

This application note describes the design of a 3-phase AC induction vector control drive with position encoder coupled to the motor shaft. It is based on Motorola's DSP56F80x dedicated motor control device. The software design takes advantage of the SDK (Software Development Kit) developed by Motorola.

AC induction motors, which contain a cage, are very popular in variable speed drives. They are simple, rugged, inexpensive and available at all power ratings. Progress in the field of power electronics and microelectronics enables the application of induction motors for high-performance drives, where traditionally only DC motors were applied. Thanks to sophisticated control methods, AC induction drives offer the same control capabilities as high performance four-quadrant DC drives.

The drive application concept presented is that of vector control of the AC induction motor running in a closed-speed loop with the speed/position sensor coupled to the shaft. The application serves as an example of AC induction vector control drive design using a Motorola DSP with SDK support. It also illustrates the usage of dedicated motor control libraries that are included in the SDK.

This application note includes a description of Motorola DSP features, basic AC induction motor theory, system design concept, hardware implementation and software design including the PC master software visualization tool.

2. Motorola DSP Advantages and Features

The Motorola DSP56F80x family is well-suited for digital motor control, combining the DSP's calculation capability with an MCU's controller features on a single chip. These DSPs offer many dedicated peripherals, including a Pulse Width Modulation (PWM) unit, an Analog-to-Digital Converter (ADC), timers, communication peripherals (SCI, SPI, CAN), on-board Flash and RAM. Generally, all the family members are well-suited for AC induction motor control.

A typical member of the family, the DSP56F805, provides the following peripheral blocks:

- Two Pulse Width Modulator units (PWMA & PWMB), each with six PWM outputs, three Current Sense inputs, and four Fault inputs, fault tolerant design with deadtime insertion; supports both Center- and Edge-aligned modes
- 12-bit Analog-to-Digital Convertors (ADCs), supporting two simultaneous conversions with dual 4-pin multiplexed inputs; ADC can be synchronized by PWM modules
- Two Quadrature Decoders (Quad Dec0 & Quad Dec1), each with four inputs, or two additional Quad Timers A & B
- Two dedicated General Purpose Quad Timers totalling six pins: timer C with two pins and timer D with four pins
- CAN 2.0 A/B Module with 2-pin ports used to transmit and receive
- Two Serial Communication Interfaces (SCI0 & SCI1), each with two pins, or four additional GPIO lines
- Serial Peripheral Interface (SPI), with configurable 4-pin port (or four additional GPIO lines)
- Computer Operating Properly (COP)/watchdog timer
- Two dedicated external interrupt pins
- 14 dedicated General Purpose I/O (GPIO) pins, 18 multiplexed GPIO pins
- External reset pin for hardware reset
- JTAG/On-Chip Emulation (OnCE) for unobtrusive, processor speed-independent debugging
- Software-programmable, Phase Lock Loop-based frequency synthesizer for the DSP core clock

Table 2-1. Memory Configuration

	DSP56F801	DSP56F803	DSP56F805	DSP56F807
Program Flash	8188 x 16-bit	32252 x 16-bit	32252 x 16-bit	61436 x 16-bit
Data Flash	2K x 16-bit	4K x 16-bit	4K x 16-bit	8K x 16-bit
Program RAM	1K x 16-bit	512 x 16-bit	512 x 16-bit	2K x 16-bit
Data RAM	1K x 16-bit	2K x 16-bit	2K x 16-bit	4K x 16-bit
Boot Flash	2K x 16-bit	2K x 16-bit	2K x 16-bit	2K x 16-bit

The key feature of the motor control DSP is the inclusion of PWM modules. The device is designed to control most motor types, including induction motors. An interesting feature for controlling the AC induction motor at low speeds is the patented PWM waveform distortion correction circuit. Each PWM is double-buffered and includes interrupt controls. The PWM module provides a reference output to synchronize the Analog-to-Digital Converters.

The PWM block has the following features:

- Three complementary PWM signal pairs, or six independent PWM signals
- Complementary channel operation
- Deadtime insertion
- Separate top and bottom pulse width correction via current status inputs or software
- Separate top and bottom polarity control
- Edge-aligned or center-aligned PWM signals
- 15-bit resolution
- Half-cycle reload capability
- Integral reload rates from 1 to 16
- Individual software-controlled PWM output
- Programmable fault protection
- Polarity control
- 20-mA current sink capability on PWM pins
- Write-protectable registers

The Analog-to-Digital Converter (ADC) consists of a digital control module and two analog sample and hold (S/H) circuits. The ADC features:

- 12-bit resolution
- Maximum ADC clock frequency of 5MHz with a 200ns period
- Single conversion time of 8.5 ADC clock cycles (8.5 x 200ns = 1.7μs)
- Additional conversion time of six ADC clock cycles (6 x 200ns = 1.2μs)
- Eight conversions in 26.5 ADC clock cycles (26.5 x 200ns = 5.3μs) using simultaneous mode
- ADC can be synchronized to the PWM via the SYNC signal
- Simultaneous or sequential sampling
- Internal multiplexer to select two of eight inputs
- Ability to sequentially scan and store up to eight measurements
- Ability to simultaneously sample and hold two inputs
- Optional interrupts at end of scan if an out-of-range limit is exceeded or at zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs

The application utilizes the ADC block in simultaneous mode and sequential scan. It is synchronized with PWM pulses. Such a configuration allows conversion of the desired analog values of all phase currents, voltage and temperature at once in the desired time.

3. Target Motor Theory

3.1 AC Induction Motor

The AC induction motor is a rotating electric machine designed to operate from a 3-phase source of alternating voltage. For variable speed drives, the source is normally an inverter that uses power switches to produce approximately sinusoidal voltages and currents of controllable magnitude and frequency.

A cross-section of a two-pole induction motor is shown in **Figure 3-1**. Slots in the inner periphery of the stator accommodate 3-phase winding a,b,c. The turns in each winding are distributed so that a current in a stator winding produces an approximately sinusoidally-distributed flux density around the periphery of the air gap. When three currents that are sinusoidally varying in time, but displaced in phase by 120° from each other, flow through the three symmetrically-placed windings, a radially-directed air gap flux density is produced that is also sinusoidally distributed around the gap and rotates at an angular velocity equal to the angular frequency ω_s of the stator currents.

The most common type of induction motor has a squirrel cage rotor in which aluminum conductors or bars are cast into slots in the outer periphery of the rotor. These conductors or bars are shorted together at both ends of the rotor by cast aluminum end rings, which also can be shaped to act as fans. In larger induction motors, copper or copper-alloy bars are used to fabricate the rotor cage winding.

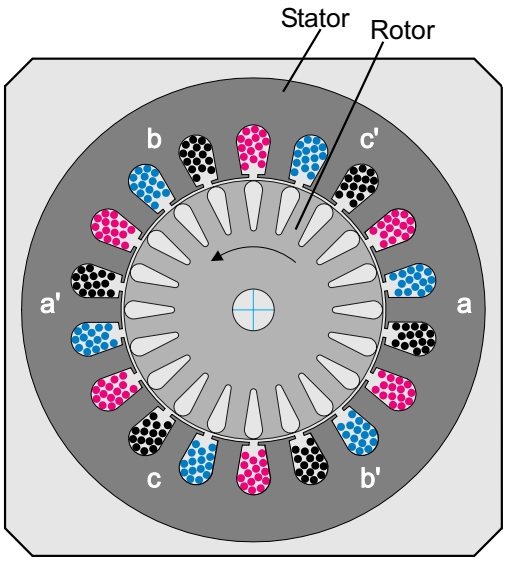

Figure 3-1. 3-Phase AC Induction Motor

As the sinusoidally-distributed flux density wave produced by the stator magnetizing currents sweeps past the rotor conductors, it generates a voltage in them. The result is a sinusoidally-distributed set of currents in the short-circuited rotor bars. Because of the low resistance of these shorted bars, only a small relative angular velocity ω_r between the angular velocity ω_s of the flux wave and the mechanical angular velocity ω of the two-pole rotor is required to produce the necessary rotor current. The relative angular velocity ω_r is called the slip velocity. The interaction of the sinusoidally-distributed air gap flux density and induced rotor currents produces a torque on the rotor. The typical induction motor speed-torque characteristic is shown in **Figure 3-2**.

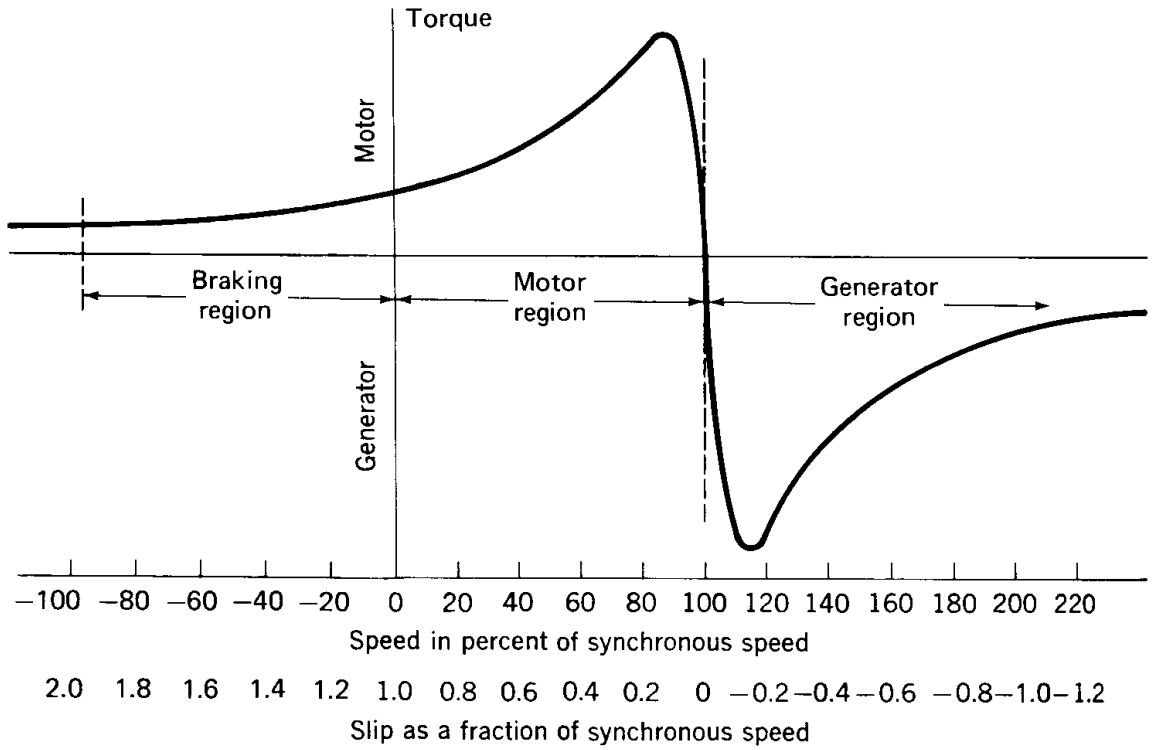

Figure 3-2. AC Induction Motor Speed-torque Characteristic

Squirrel-cage AC induction motors are popular for their simple construction, low cost per horsepower and low maintenance (they contain no brushes, as do DC motors). They are available in a wide range of power ratings. With field-oriented vector control methods, AC induction motors can fully replace standard DC motors, even in high-performance applications.

3.2 Mathematical Description of AC Induction Motors

There are a number of AC induction motor models. The model used for vector control design can be obtained by utilization of the space vector theory. The 3-phase motor quantities (such as voltages, currents, magnetic flux, etc.) are expressed in the term of complex space vectors. Such a model is valid for any instantaneous variation of voltage and current and adequately describes the performance of the machine under both steady-state and transient operation. Complex space vectors can be described using only two orthogonal axes. We can look at the motor as a 2-phase machine. The utilization of the 2-phase motor model reduces the number of equations and simplifies the control design.

3.2.1 Space Vector Definition

Let's assume i_{sa} , i_{sb} , and i_{sc} are the instantaneous balanced 3-phase stator currents:

$$i_{sa} + i_{sb} + i_{sc} = 0 \qquad \text{(EQ 3-1)}$$

Then we can define the stator current space vector as follows:

$$\bar{i_s} = k(i_{sa} + a i_{sb} + a^2 i_{sc}) \qquad \text{(EQ 3-2)}$$

where a and a^2 are the spatial operators, $a = e^{j2\pi/3}$, $a^2 = e^{j4\pi/3}$ and k is the transformation constant and is chosen $k=2/3$. **Figure 3-3** shows the stator current space vector projection:

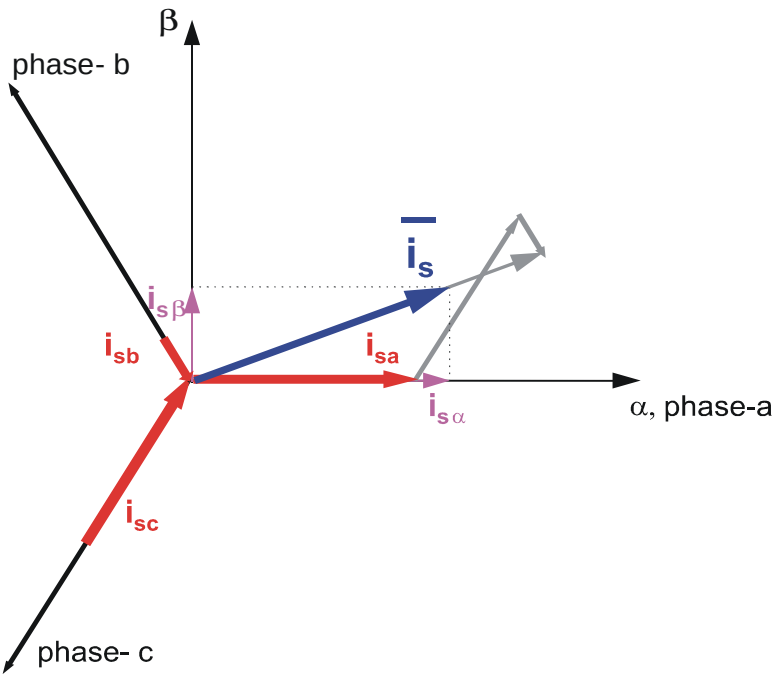

Figure 3-3. Stator Current Space Vector and Its Projection

The space vector defined by (EQ 3-2) can be expressed utilizing the two-axis theory. The real part of the space vector is equal to the instantaneous value of the direct-axis stator current component, $i_{s\alpha}$, and whose imaginary part is equal to the quadrature-axis stator current component, $i_{s\beta}$. Thus, the stator current space vector in the stationary reference frame attached to the stator can be expressed as:

$$\bar{i}_s = i_{s\alpha} + j i_{s\beta} \quad \text{(EQ 3-3)}$$

In symmetrical 3-phase machines, the direct and quadrature axis stator currents $i_{s\alpha}$, $i_{s\beta}$ are fictitious quadrature-phase (2-phase) current components, which are related to the actual 3-phase stator currents as follows:

$$i_{s\alpha} = k\left(i_{sa} - \frac{1}{2}i_{sb} - \frac{1}{2}i_{sc}\right) \quad \text{(EQ 3-4)}$$

$$i_{s\beta} = k\frac{\sqrt{3}}{2}(i_{sb} - i_{sc}) \quad \text{(EQ 3-5)}$$

where $k=2/3$ is a transformation constant.

The space vectors of other motor quantities (voltages, currents, magnetic fluxes, etc.) can be defined in the same way as the stator current space vector.

3.2.2 AC Induction Motor Model

The AC induction motor model is given by the space vector form of the voltage equations. The system model defined in the stationary α,β-coordinate system attached to the stator is expressed by the following equations. The motor model is supposed to be ideally symmetrical with a linear magnetic circuit characteristic.

a. The stator voltage differential equations:

$$u_{s\alpha} = R_s i_{s\alpha} + \frac{d}{dt}\Psi_{s\alpha} \quad \text{(EQ 3-6)}$$

$$u_{s\beta} = R_s i_{s\beta} + \frac{d}{dt}\Psi_{s\beta} \quad \text{(EQ 3-7)}$$

b. The rotor voltage differential equations:

$$u_{r\alpha} = 0 = R_r i_{r\alpha} + \frac{d}{dt}\Psi_{r\alpha} + \omega\Psi_{r\beta} \quad \text{(EQ 3-8)}$$

$$u_{r\beta} = 0 = R_r i_{r\beta} + \frac{d}{dt}\Psi_{r\beta} - \omega\Psi_{r\alpha} \quad \text{(EQ 3-9)}$$

c. The stator and rotor flux linkages expressed in terms of the stator and rotor current space vectors:

$$\Psi_{s\alpha} = L_s i_{s\alpha} + L_m i_{r\alpha} \quad \text{(EQ 3-10)}$$

$$\Psi_{s\beta} = L_s i_{s\beta} + L_m i_{r\beta} \quad \text{(EQ 3-11)}$$

$$\Psi_{r\alpha} = L_r i_{r\alpha} + L_m i_{s\alpha} \quad \text{(EQ 3-12)}$$

$$\Psi_{r\beta} = L_r i_{r\beta} + L_m i_{s\beta} \quad \text{(EQ 3-13)}$$

d. Electromagnetic torque expressed by utilizing space vector quantities:

$$t_e = \frac{3}{2} p_p (\Psi_{s\alpha} i_{s\beta} - \Psi_{s\beta} i_{s\alpha}) \quad \text{(EQ 3-14)}$$

where:

α,β	stator orthogonal coordinate system	
$u_{s\alpha,\beta}$	stator voltages	[V]
$i_{s\alpha,\beta}$	stator currents	[A]
$u_{r\alpha,\beta}$	rotor voltages	[V]
$i_{r\alpha,\beta}$	rotor currents	[A]
$\Psi_{s\alpha,\beta}$	stator magnetic fluxes	[Vs]
$\Psi_{r\alpha,\beta}$	rotor magnetic fluxes	[Vs]
R_s	stator phase resistance	[Ohm]
R_r	rotor phase resistance	[Ohm]
L_s	stator phase inductance	[H]

L_r	rotor phase inductance	[H]
L_m	mutual (stator to rotor) inductance	[H]
ω / ω_s	electrical rotor speed / synchronous speed	[rad/s]
p_p	number of pole pairs	[-]
t_e	electromagnetic torque	[Nm]

Besides the stationary reference frame attached to the stator, motor model voltage space vector equations can be formulated in a general reference frame, which rotates at a general speed ω_g. If a general reference frame, with direct and quadrature axes *x*,*y* rotating at a general instantaneous speed $\omega_g = d\theta_g/dt$ is used, as shown in **Figure 3-4**, where θ_g is the angle between the direct axis of the stationary reference frame (α) attached to the stator and the real axis (*x*) of the general reference frame, then the following equation defines the stator current space vector in general reference frame:

$$\overline{i_{sg}} = \bar{i}_s e^{-j\theta_g} = i_{sx} + j i_{sy} \qquad \text{(EQ 3-15)}$$

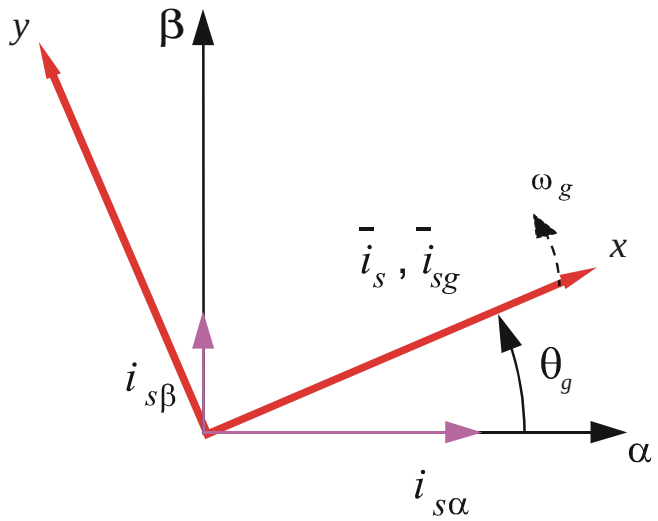

Figure 3-4. Application of the General Reference Frame

The stator voltage and flux-linkage space vectors can be similarly obtained in the general reference frame.

Similar considerations hold for the space vectors of the rotor voltages, currents and flux linkages. The real axis ($r\alpha$) of the reference frame attached to the rotor is displaced from the direct axis of the stator reference frame by the rotor angle θ_r . It can be seen that the angle between the real axis (*x*) of the general reference frame and the real axis of the reference frame rotating with the rotor ($r\alpha$) is θ_g-θ_r . In the general reference frame, the space vector of the rotor currents can be expressed as:

$$\overline{i_{rg}} = \bar{i}_r e^{-j(\theta_g - \theta_r)} = i_{rx} + j i_{ry} \qquad \text{(EQ 3-16)}$$

where $\bar{i}_r$ is the space vector of the rotor current in the rotor reference frame.

Similarly, the space vectors of the rotor voltages and rotor flux linkages in the general reference frame can be expressed.

By utilizing introduced transformations of the motor quantities from one reference frame to the general reference frame, the motor model voltage equations in the general reference frame can be expressed. The AC induction motor model is often used in vector control algorithms. The aim of vector control is to implement control schemes which produce high dynamic performance and are similar to those used to control DC machines. To achieve this, the reference frames may be aligned with the stator flux-linkage space vector, the rotor flux-linkage space vector or the magnetizing space vector. The most popular reference frame is the reference frame attached to the rotor flux linkage space vector with direct axis (*d*) and quadrature axis (*q*). After transformation into d-q coordinates the motor model is the following:

$$u_{sd} = R_s i_{sd} + \frac{d}{dt}\Psi_{sd} - \omega_s \Psi_{sq} \quad \text{(EQ 3-17)}$$

$$u_{sq} = R_s i_{sq} + \frac{d}{dt}\Psi_{sq} - \omega_s \Psi_{sd} \quad \text{(EQ 3-18)}$$

$$u_{rd} = 0 = R_r i_{rd} + \frac{d}{dt}\Psi_{rd} - (\omega_s - \omega)\Psi_{rq} \quad \text{(EQ 3-19)}$$

$$u_{rq} = 0 = R_r i_{rq} + \frac{d}{dt}\Psi_{rq} + (\omega_s - \omega)\Psi_{rd} \quad \text{(EQ 3-20)}$$

$$\Psi_{sd} = L_s i_{sd} + L_m i_{rd} \quad \text{(EQ 3-21)}$$

$$\Psi_{sq} = L_s i_{sq} + L_m i_{rq} \quad \text{(EQ 3-22)}$$

$$\Psi_{rd} = L_r i_{rd} + L_m i_{sd} \quad \text{(EQ 3-23)}$$

$$\Psi_{rq} = L_r i_{rq} + L_m i_{sq} \quad \text{(EQ 3-24)}$$

$$t_e = \frac{3}{2} p_p (\Psi_{sd} i_{sq} - \Psi_{sq} i_{sd}) \quad \text{(EQ 3-25)}$$

3.3 Digital Control of an AC Induction Motor

In adjustable speed applications, AC motors are powered by inverters. The inverter converts DC power to AC power at the required frequency and amplitude.

Figure 3-5 illustrates a typical 3-phase inverter.

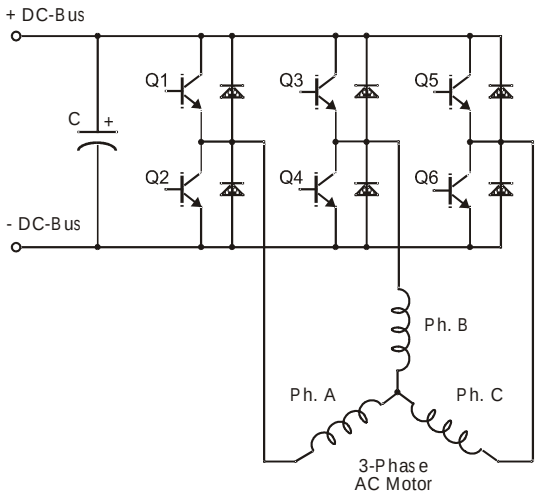

Figure 3-5. 3- Phase Inverter

The inverter consists of three half-bridge units where the upper and lower switch are controlled complementarily, meaning when the upper one is turned on, the lower one must be turned off, and vice versa. As the power device's turn-off time is longer than its turn-on time, some dead-time must be inserted between the turn-off of one transistor of the half-bridge and the turn-on of its complementary device. The output voltage is mostly created by a pulse width modulation (PWM) technique, where an isosceles triangle carrier wave is compared with a fundamental-frequency sine modulating wave and the natural points of intersection determine the switching points of the power devices of a half-bridge inverter. This technique is shown in **Figure 3-6.** The 3-phase voltage waves are shifted 120^{o} to one another and thus a 3-phase motor can be supplied.

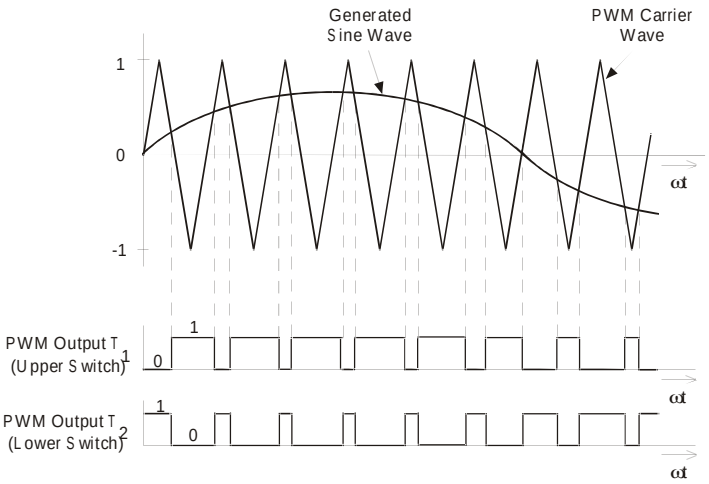

Figure 3-6. Pulse Width Modulation

The most popular power devices for motor control applications are Power MOSFETs and IGBTs.

A Power MOSFET is a voltage-controlled transistor. It is designed for high-frequency operation and has a low-voltage drop, so it has low power losses. However, saturation temperature sensitivity limits the MOSFET's use in high-power applications.

An Insulated-Gate Bipolar Transistor (IGBT) is controlled by a MOSFET on its base. The IGBT requires low drive current, has fast switching time, and is suitable for high switching frequencies. The disadvantage is the higher voltage drop of the bipolar transistor, causing higher conduction losses.

4. Vector Control of AC Induction Machines

Vector control is the most popular control technique of AC induction motors. In special reference frames, the expression for the electromagnetic torque of the smooth-air-gap machine is similar to the expression for the torque of the separately excited DC machine. In the case of induction machines, the control is usually performed in the reference frame (d-q) attached to the rotor flux space vector. That's why the implementation of vector control requires information on the modulus and the space angle (position) of the rotor flux space vector. The stator currents of the induction machine are separated into flux- and torque-producing components by utilizing transformation to the d-q coordinate system, whose direct axis (*d*) is aligned with the rotor flux space vector. That means that the *q*-axis component of the rotor flux space vector is always zero:

$$\Psi_{rq} = 0 \text{ and also } \frac{d}{dt}\Psi_{rq} = 0 \qquad \text{(EQ 4-1)}$$

The rotor flux space vector calculation and transformation to the d-q coordinate system require the high computational power of a microcontroller. The digital signal processor is suitable for this task. The following sections describe the space vector transformations and the rotor flux space vector calculation.

4.1 Block Diagram of the Vector Control

Figure 4-1 shows the basic structure of the vector control of the AC induction motor. To perform vector control, it is necessary to follow these steps:

- Measure the motor quantities (phase voltages and currents)
- Transform them to the 2-phase system (α,β) using a Clarke transformation
- Calculate the rotor flux space vector magnitude and position angle
- Transform stator currents to the d-q coordinate system using a Park transformation
- The stator current torque (i_{sq}) and flux (i_{sd}) producing components are separately controlled
- The output stator voltage space vector is calculated using the decoupling block
- The stator voltage space vector is transformed by an inverse Park transformation back from the d-q coordinate system to the 2-phase system fixed with the stator
- Using the space vector modulation, the output 3-phase voltage is generated

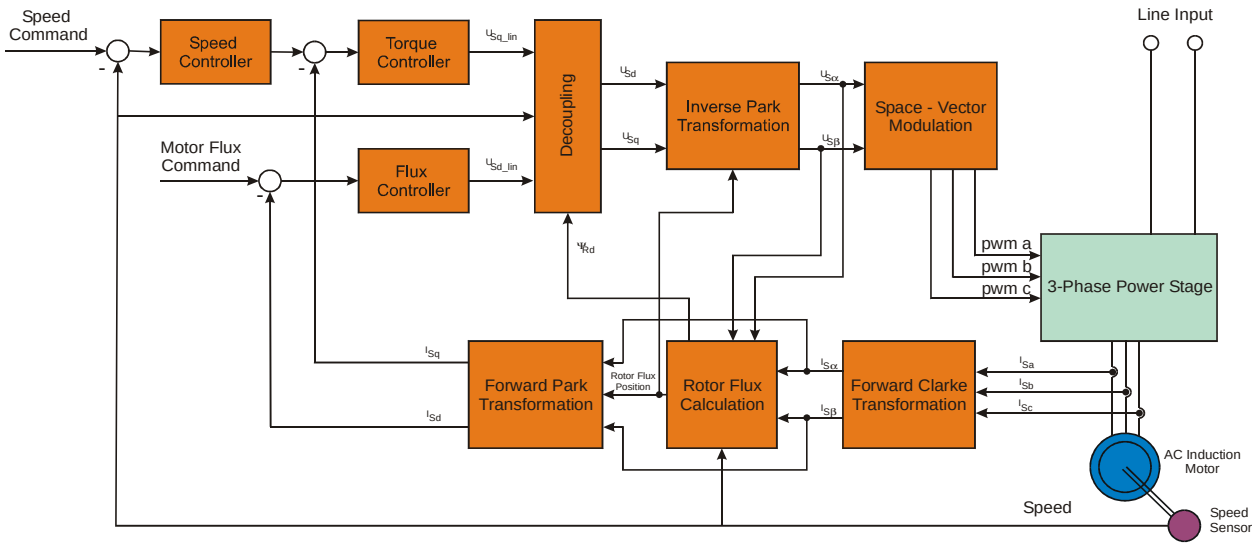

Figure 4-1. Block Diagram of the AC Induction Motor Vector Control

4.2 Forward and Inverse Clarke Transformation (a,b,c to α,β and backwards)

The forward Clarke transformation converts a 3-phase system a,b,c to a 2-phase coordinate system α,β. **Figure 4-2** shows graphical construction of the space vector and projection of the space vector to the quadrature-phase components α,β.

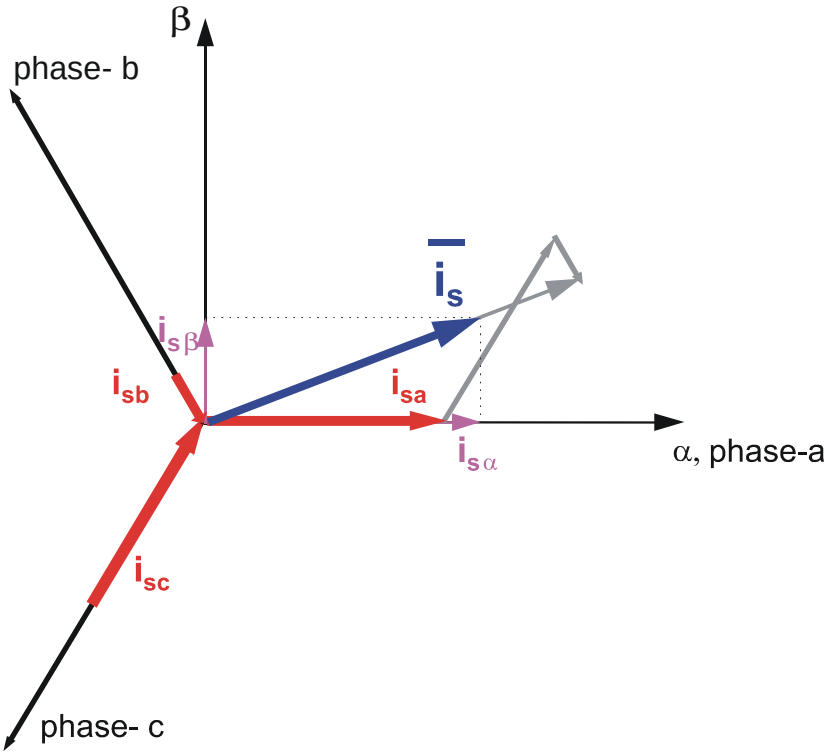

Figure 4-2. Clarke Transformation

Assuming that the *a* axis and the α axis are in the same direction, the quadrature-phase stator currents $i_{s\alpha}$ and $i_{s\beta}$ are related to the actual 3-phase stator currents as follows:

$$i_{s\alpha} = k\left[i_{sa} - \frac{1}{2}i_{sb} - \frac{1}{2}i_{sc}\right]$$
$$i_{s\beta} = k\frac{\sqrt{3}}{2}(i_{sb} - i_{sc}) \quad \text{(EQ 4-2)}$$

where:

i_{sa}	actual current of the motor Phase a	[A]
i_{sb}	actual current of the motor Phase b	[A]
i_{sc}	actual current of the motor Phase c	[A]

For the non-power-invariant transformation the constant *k* equals *k*=2/3. In this case, the quantities i_{sa} and $i_{s\alpha}$ are equal. If we assume $i_{sa} + i_{sb} + i_{sc} = 0$, the quadrature-phase components can be expressed utilizing only two phases of the 3-phase system:

$$i_{s\alpha} = i_{sa}$$
$$i_{s\beta} = \frac{1}{\sqrt{3}}i_{sa} + \frac{2}{\sqrt{3}}i_{sb} \quad \text{(EQ 4-3)}$$

The inverse Clarke transformation goes back from a 2-phase (α,β) to a 3-phase i_{sa}, i_{sb}, i_{sc} system. For constant *k*=2/3, it is given by the following equations:

$$i_{sa} = i_{s\alpha}$$
$$i_{sb} = -\frac{1}{2}i_{s\alpha} + \frac{\sqrt{3}}{2}i_{s\beta} \quad \text{(EQ 4-4)}$$
$$i_{sc} = -\frac{1}{2}i_{s\alpha} - \frac{\sqrt{3}}{2}i_{s\beta}$$

4.3 Forward and Inverse Park Transformation (α,β to d-q and backwards)

The components $i_{s\alpha}$ and $i_{s\beta}$, calculated with a Clarke transformation, are attached to the stator reference frame α, β. In vector control, it is necessary to have all quantities expressed in the same reference frame. The stator reference frame is not suitable for the control process. The space vector $\bar{i}_s$ is rotating at a rate equal to the angular frequency of the phase currents. The components $i_{s\alpha}$ and $i_{s\beta}$ depend on time and speed. We can transform these components from the stator reference frame to the d-q reference frame rotating at the same speed as the angular frequency of the phase currents. Then the i_{sd} and i_{sq} components do not depend on time and speed. If we consider the *d*-axis aligned with the rotor flux, the transformation is illustrated in **Figure 4-3**, where θ_{Field} is the rotor flux position.

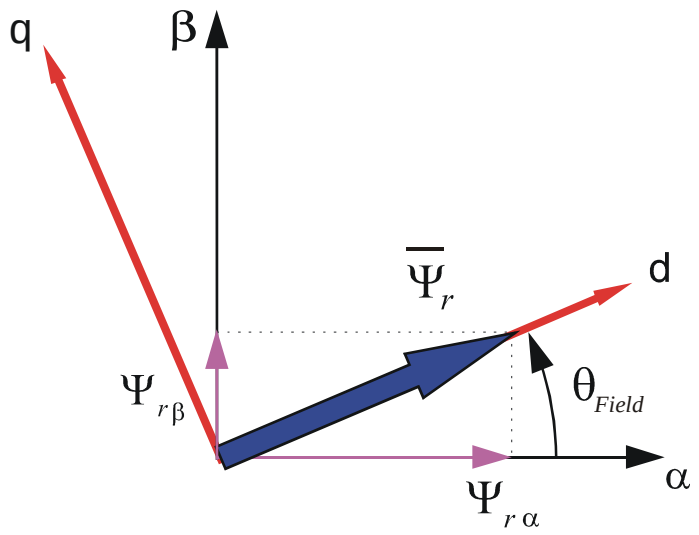

Figure 4-3. Park Transformation

The components i_{sd} and i_{sq} of the current space vector in d-q reference frame are determined by the following equations:

$$\begin{aligned} i_{sd} &= i_{s\alpha}\cos\theta_{Field} + i_{s\beta}\sin\theta_{Field} \\ i_{sq} &= -i_{s\alpha}\sin\theta_{Field} + i_{s\beta}\cos\theta_{Field} \end{aligned} \quad \text{(EQ 4-5)}$$

The component i_{sd} is called the direct axis component (flux producing component) and i_{sq} is called the quadrature axis component (torque producing component). They are time invariant and the flux and torque control with them is easy. To avoid using trigonometric functions on the DSP we can directly calculate $\sin\theta_{Field}$ and $\cos\theta_{Field}$ using division. They are defined by the following equations:

$$\Psi_{rd} = \sqrt{\Psi_{r\alpha} + \Psi_{r\beta}} \quad \text{(EQ 4-6)}$$

$$\begin{aligned} \sin\theta_{Field} &= \frac{\Psi_{r\beta}}{\Psi_{rd}} \\ \cos\theta_{Field} &= \frac{\Psi_{r\alpha}}{\Psi_{rd}} \end{aligned} \quad \text{(EQ 4-7)}$$

The inverse Park transformation from the d-q to α,β coordinate system is given by the following equations:

$$\begin{aligned} i_{s\alpha} &= i_{sd}\cos\theta_{Field} - i_{sq}\sin\theta_{Field} \\ i_{s\beta} &= i_{sd}\sin\theta_{Field} + i_{sq}\cos\theta_{Field} \end{aligned} \quad \text{(EQ 4-8)}$$

4.4 Rotor Flux Model

Knowledge of the rotor flux space vector magnitude and position is key information for the AC induction motor vector control. With the rotor magnetic flux space vector, the rotational coordinate system (d-q) can be established. There are several methods for obtaining the rotor magnetic flux space vector. The implemented flux model utilizes monitored rotor speed and stator voltages and currents. It is calculated in the stationary reference frame (α,β) attached to the stator. The error in the calculated value of the rotor flux, influenced by the changes in temperature, is negligible for this rotor flux model.

The rotor flux space vector is obtained by solving the differential equations (EQ 4-2) and (EQ 4-3), which are resolved into the α and β components. The equations are derived from the equations of the AC induction motor model **(see Section 3.2.2 AC Induction Motor Model)**.

$$[(1-\sigma)T_s + T_r]\frac{d\Psi_{r\alpha}}{dt} = \frac{L_m}{R_s}u_{s\alpha} - \Psi_{r\alpha} - \omega T_r \Psi_{r\beta} - \sigma L_m T_s \frac{di_{s\alpha}}{dt} \quad \text{(EQ 4-9)}$$

$$[(1-\sigma)T_s + T_r]\frac{d\Psi_{r\beta}}{dt} = \frac{L_m}{R_s}u_{s\beta} + \omega T_r \Psi_{r\alpha} - \Psi_{r\beta} - \sigma L_m T_s \frac{di_{s\beta}}{dt} \quad \text{(EQ 4-10)}$$

where:

L_s	self-inductance of the stator	[H]
L_r	self-inductance of the rotor	[H]
L_m	magnetizing inductance	[H]
R_r	resistance of a rotor phase winding	[Ohm]
R_s	resistance of a stator phase winding	[Ohm]
ω	angular rotor speed	[rad.s^{-1}]
p_p	number of motor pole-pairs	
$T_r = \frac{L_r}{R_r}$	rotor time constant	[s]
$T_s = \frac{L_s}{R_s}$	stator time constant	[s]
$\sigma = 1 - \frac{L_m^2}{L_s L_r}$	resultant leakage constant	[-]

$u_{s\alpha}$, $u_{s\beta}$, $i_{s\alpha}$, $i_{s\beta}$, $\Psi_{r\alpha}$, $\Psi_{r\beta}$ are the α, β components of the stator voltage, currents and rotor flux space vectors

4.5 Decoupling Circuit

For purposes of the rotor flux-oriented vector control, the direct-axis stator current i_{sd} (rotor flux-producing component) and the quadrature-axis stator current i_{sq} (torque-producing component) must be controlled independently. However, the equations of the stator voltage components are coupled. The direct axis component u_{sd} also depends on i_{sq} and the quadrature axis component u_{sq} also depends on i_{sd}. The stator voltage components u_{sd} and u_{sq} cannot be considered as decoupled control variables for the rotor flux and electromagnetic torque. The stator currents i_{sd} and i_{sq} can only be

independently controlled (decoupled control) if the stator voltage equations are decoupled and the stator current components i_{sd} and i_{sq} are indirectly controlled by controlling the terminal voltages of the induction motor.

The equations of the stator voltage components in the d-q coordinate system (EQ 3-22) and (EQ 3-23) can be reformulated and separated into two components: linear components $u_{sd}^{lin}, u_{sq}^{lin}$ and decoupling components $u_{sd}^{decouple}, u_{sq}^{decouple}$. The equations are decoupled as follows:

$$u_{sd} = u_{sd}^{lin} + u_{sd}^{decouple} = \left[K_R i_{sd} + K_L \frac{d}{dt} i_{sd}\right] - \left[\omega_s K_L i_{sq} + \frac{\Psi_{rd} L_m}{L_r T_r}\right] \quad \text{(EQ 4-11)}$$

$$u_{sq} = u_{sq}^{lin} + u_{sq}^{decouple} = \left[K_R i_{sq} + K_L \frac{d}{dt} i_{sq}\right] + \left[\omega_s K_L i_{sd} + \frac{L_m}{L_r} \omega \Psi_{rd}\right] \quad \text{(EQ 4-12)}$$

where:

$$K_R = R_s + \frac{L_m^2}{L_r^2} R_r \quad \text{(EQ 4-13)}$$

$$K_L = L_s - \frac{L_m^2}{L_r} \quad \text{(EQ 4-14)}$$

The voltage components $u_{sd}^{lin}, u_{sq}^{lin}$ are the outputs of the current controllers which control i_{sd} and i_{sq} components. They are added to the decoupling voltage components $u_{sd}^{decouple}, u_{sq}^{decouple}$. In this way, we can get direct and quadrature components of the terminal output voltage. This means the voltage on the outputs of the current controllers is:

$$u_{sd}^{lin} = K_R i_{sd} + K_L \frac{d}{dt} i_{sd} \quad \text{(EQ 4-15)}$$

$$u_{sq}^{lin} = K_R i_{sq} + K_L \frac{d}{dt} i_{sq} \quad \text{(EQ 4-16)}$$

And the decoupling components are:

$$u_{sd}^{decouple} = -\left(\omega_s K_L i_{sq} + \frac{L_m}{L_r T_r} \Psi_{rd}\right) \quad \text{(EQ 4-17)}$$

$$u_{sq}^{decouple} = \left(\omega_s K_L i_{sd} + \frac{L_m}{L_r} \omega \Psi_{rd}\right) \quad \text{(EQ 4-18)}$$

As can be seen, the decoupling algorithm transforms the nonlinear motor model to linear equations which can be controlled by general PI or PID controllers instead of complicated controllers.

4.6 Space Vector Modulation

Space Vector Modulation (SVM) can directly transform the stator voltage vectors from α,β-coordinate system to pulse width modulation (PWM) signals (duty cycle values).

The standard technique of the output voltage generation uses an inverse Clarke transformation to obtain 3-phase values. Using the phase voltage values, the duty cycles needed to control the power stage switches are then calculated. Although this technique gives good results, the space vector modulation is more straightforward (valid only for transformation from the α,β-coordinate system).

The basic principle of the standard space vector modulation technique can be explained with the help of the power stage schematic diagram depicted in **Figure 4-4**.

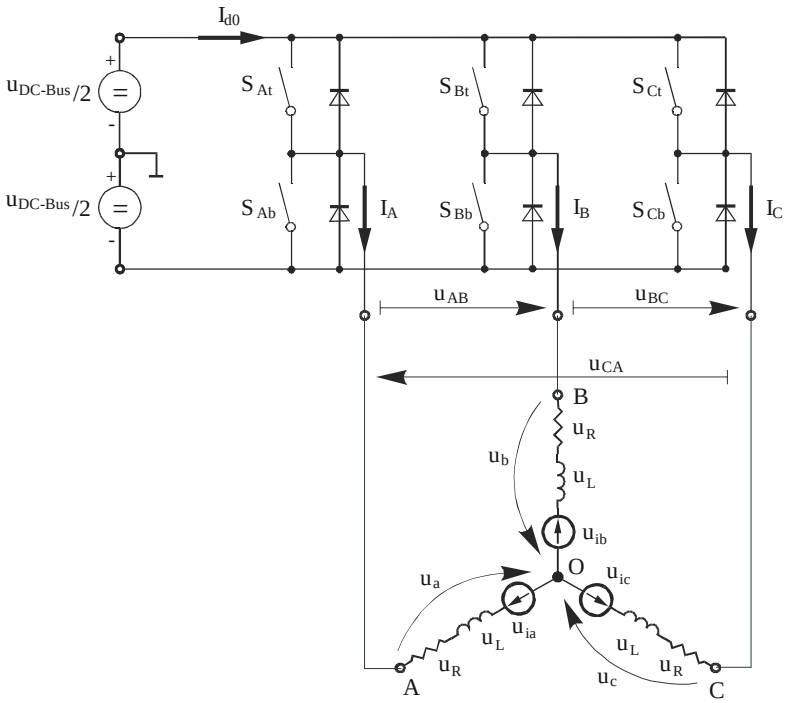

Figure 4-4. Power Stage Schematic Diagram

Regarding the 3-phase power stage configuration, as shown in **Figure 4-4**, eight possible switching states (vectors) are feasible. They are given by combinations of the corresponding power switches.The graphical representation of all combinations is the hexagon shown in **Figure 4-5**. There are six non-zero vectors, U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, and two zero vectors, O_{000} and O_{111}, defined in α,β coordinates.

The combination of ON/OFF states of the power stage switches for each voltage vector is coded in **Figure 4-5** by the three-digit number in parenthesis. Each digit represents one phase. For each phase, a value of one means that the upper switch is ON and the bottom switch is OFF. A value of zero means that the upper switch is OFF and the bottom switch is ON. These states, together with the resulting instantaneous output line-to-line voltages, phase voltages and voltage vectors, are listed in **Table 4-1**.

Table 4-1. Switching Patterns and Resulting Instantaneous Line-to-Line and Phase Voltages

a	b	c	U_a	U_b	U_c	U_{AB}	U_{BC}	U_{CA}	Vector
0	0	0	0	0	0	0	0	0	O_{000}
1	0	0	$2U_{DCBus}/3$	$-U_{DCBus}/3$	$-U_{DCBus}/3$	U_{DCBus}	0	$-U_{DCBus}$	U_0
1	1	0	$U_{DCBus}/3$	$U_{DCBus}/3$	$-2U_{DCBus}/3$	0	U_{DCBus}	$-U_{DCBus}$	U_{60}
0	1	0	$-U_{DCBus}/3$	$2U_{DCBus}/3$	$-U_{DCBus}/3$	$-U_{DCBus}$	U_{DCBus}	0	U_{120}
0	1	1	$-2U_{DCBus}/3$	$U_{DCBus}/3$	$U_{DCBus}/3$	$-U_{DCBus}$	0	U_{DCBus}	U_{240}
0	0	1	$-U_{DCBus}/3$	$-U_{DCBus}/3$	$2U_{DCBus}/3$	0	$-U_{DCBus}$	U_{DCBus}	U_{300}
1	0	1	$U_{DCBus}/3$	$-2U_{DCBus}/3$	$U_{DCBus}/3$	U_{DCBus}	$-U_{DCBus}$	0	U_{360}
1	1	1	0	0	0	0	0	0	O_{111}

.

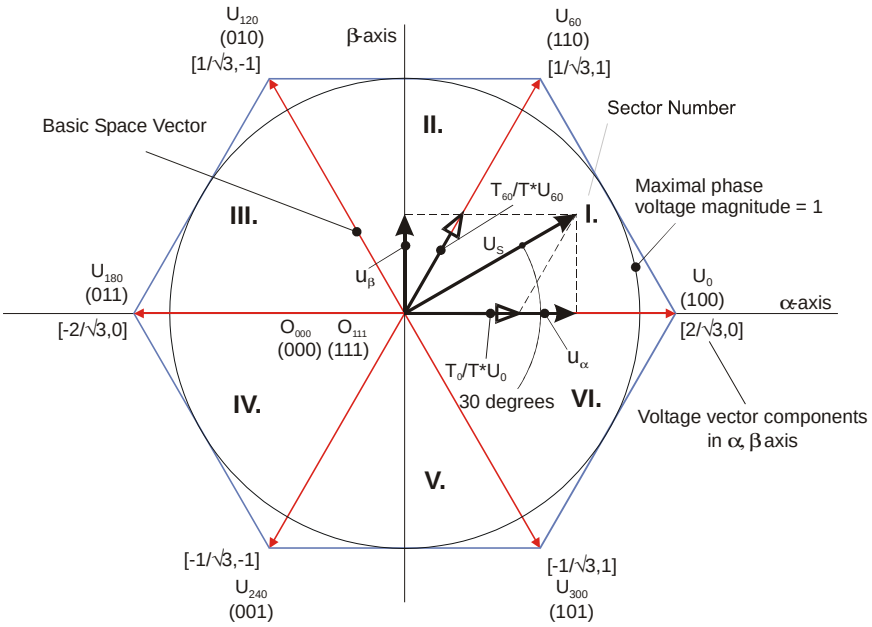

Figure 4-5. Basic Space Vectors and Voltage Vector Projection

SVM is a technique used as a direct bridge between vector control (voltage space vector) and PWM.

The SVM technique consists of several steps:

1. Sector identification
2. Space voltage vector decomposition into directions of sector base vectors U_x, $U_{x\pm60}$
3. PWM duty cycle calculation

The principle of SVM is the application of the voltage vectors U_{XXX} and O_{XXX} for certain instances in such a way that the “mean vector” of the PWM period T_{PWM} is equal to the desired voltage vector.

This method gives the greatest variability of arrangement of the zero and non-zero vectors during the PWM period. One can arrange these vectors to lower switching losses; another might want to approach a different result, such as center-aligned PWM, edge-aligned PWM, minimal switching, etc.

For the chosen SVM, we define the following rule:

- The desired space voltage vector is created only by applying the sector base vectors: the non-zero vectors on the sector side, (U_x, $U_{x\pm60}$) and the zero vectors (O_{000} or O_{111}).

The following expressions define the principle of the SVM:

$$T_{PWM} \cdot U_{S[\alpha,\beta]} = T_1 \cdot U_x + T_2 \cdot U_{x\pm60} + T_0 \cdot (O_{000} \vee O_{111}) \quad \text{(EQ 4-19)}$$

$$T_{PWM} = T_1 + T_2 + T_0 \quad \text{(EQ 4-20)}$$

In order to solve the time periods T_0, T_1 and T_2, it is necessary to decompose the space voltage vector $U_{S[\alpha,\beta]}$ into directions of the sector base vectors U_x, $U_{x\pm60}$. The equation (EQ 4-19) splits into equations (EQ 4-21) and (EQ 4-22).

$$T_{PWM} \cdot U_{Sx} = T_1 \cdot U_x \quad \text{(EQ 4-21)}$$

$$T_{PWM} \cdot U_{S(x\pm60)} = T_2 \cdot U_{x\pm60} \quad \text{(EQ 4-22)}$$

By solving this set of equations, we can calculate the necessary duration of the application of the sector base vectors U_x, $U_{x\pm60}$ during the PWM period T_{PWM} to produce the right stator voltages.

$$T_1 = \frac{|U_{Sx}|}{|U_x|} T_{PWM} \quad \text{for vector } U_x \quad \text{(EQ 4-23)}$$

$$T_2 = \frac{|U_{Sx}|}{|U_{x\pm60}|} T_{PWM} \quad \text{for vector } U_{x\pm60} \quad \text{(EQ 4-24)}$$

$$T_0 = T_{PWM} - (T_1 + T_2) \quad \text{either for } O_{000} \text{ or } O_{111} \quad \text{(EQ 4-25)}$$

5. Design Concept of ACIM Vector Control Drives

5.1 System Outline

The system is designed to drive a 3-phase AC induction motor (ACIM). The application has the following specifications:

- Vector control technique used for ACIM control
- Speed control loop of the ACIM
- Targeted for DSP56F803EVM, DSP56F805EVM, DSP56F807EVM
- Running on 3-phase AC induction motor control development platform at variable line voltage 115/230V AC (range -15%.....+10%)

- Control technique incorporates
  - — Speed control loop with inner *q* axis stator current loop
  - — Rotor flux control loop with inner *d* axis stator current loop
  - — Field-weakening technique
  - — Stator phase current measurement method
  - — AC induction flux model calculation in α, β - stationary reference frame
  - — Forward Clarke and inverse Park transformations
  - — d-q establishment - transformation from the stationary reference frame to the rotating reference frame
  - — DCBus ripple elimination
  - — Space Vector Modulation (SVM)
- Motor mode
- Generator mode
- DCBus brake
- Minimum speed of 50 rpm
- Maximum speed of 2500 rpm at input power line 230V AC
- Maximum speed 1100 rpm at input power line 115V AC
- Manual interface (RUN/STOP switch, UP/DOWN push buttons control, LED indication)
- Power stage board identification
- Overvoltage, undervoltage, overcurrent and overheating fault protection
- PC remote control interface (Start/Stop Motor push buttons, speed set-up)
- PC master software remote monitor
  - — PC master software monitor interface (required speed, actual motor speed, PC master software mode, START MOTOR/STOP MOTOR controls, drive fault status, DCBus voltage level, identified power stage boards, drive status, mains detection)
  - — PC master software speed scope (observes actual and desired speed)

5.2 Application Description

The vector control algorithm is calculated on Motorola DSP56F80x. According to the user-required inputs, measured and calculated signals, the algorithm generates 3-phase PWM signals for an AC induction motor inverter.

The block diagram of the ACIM control algorithm is shown in **Figure 5-1**, which describes the structure of the implemented vector control algorithm (basic blocks and control signals).

The system incorporates the following hardware components:

- 3-phase AC induction motor with load coupled on the motor shaft
- 3-phase AC/BLDC high-voltage power stage
- DSP56F803EVM / DSP56F805EVM / DSP56F807EVM boards
- ECOPTINL, In-line optoisolation box, which is connected between the host computer and the DSP56F80xEVM

The drive can be controlled in two different operating modes:

- In the **manual operating mode**, the required speed is set by UP/DOWN push buttons and the drive is started and stopped by the RUN/STOP switch on the EVM board
- In the **PC remote control operating mode**, the required speed is set by the PC master software bar graph and the drive is started and stopped by the START MOTOR and STOP MOTOR controls

Measured quantities:

- DCBus voltage
- Phase currents (Phase A, Phase B, Phase C)
- Power module temperature
- Rotor speed

The faults used for drive protection:

- “Overvoltage”
- “Undervoltage”
- “Overcurrent”
- “Overheating”
- “Mains out of range”
- “Wrong hardware”
- “Overload”

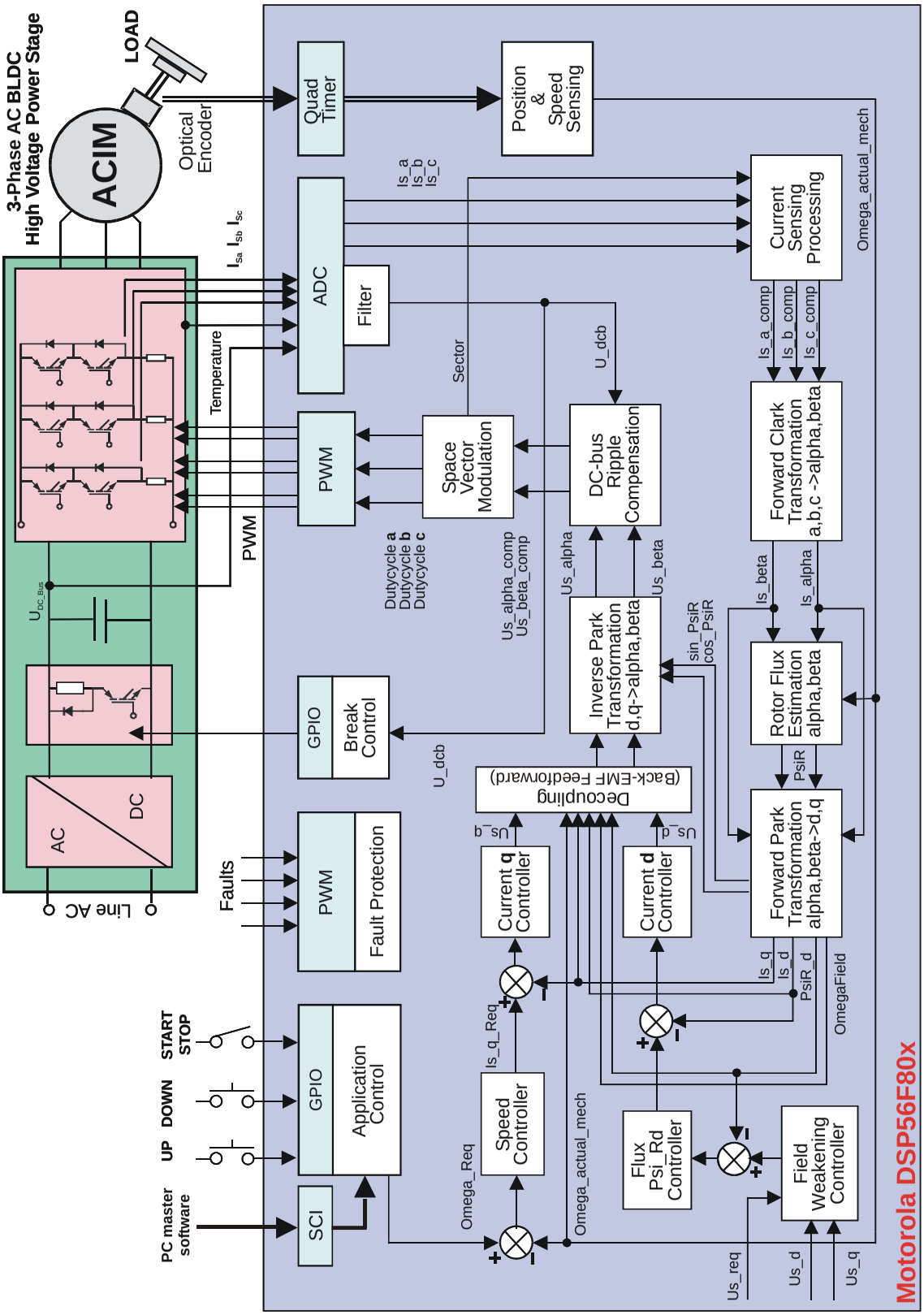

Figure 5-1. AC Induction Motor Vector Control Drive Structure

5.2.1 Control Process

After reset, the drive is in the INIT state and in the manual operation mode. When the RUN/STOP switch is detected in the stop position and there are no faults pending, the INIT state is changed to the STOP state. Otherwise, the drive waits in the INIT state. If a fault occurs, it goes to the FAULT state. In the INIT and STOP states, the operating mode can be changed from the PC master software. In the manual operating mode, the application is controlled by the RUN/STOP switch and UP/DOWN push buttons; in the PC remote-control mode, the application is controlled by the PC master software.

When the start command is accepted (from the RUN/STOP switch or the PC master software command), the STOP state is changed to the RUN state. The required speed is then calculated from the UP/DOWN push buttons or PC master software commands, if in PC remote control mode. The required speed is the input into the acceleration/deceleration ramp and the output is used as a reference command for the speed controller; **(see Figure 5-1.)**. The difference between the actual speed and the required speed generates a speed error. Based on the error, the speed controller generates an *Is_q_Req* current which corresponds to the torque component. The second component of the stator current *Is_d_Req*, which corresponds to the rotor flux, is given by the flux controller. The field-weakening algorithm generates the required rotor flux, which is compared to the calculated rotor flux from the *AC induction flux model calculation* algorithm. The difference between the required rotor flux and calculated rotor flux generates a flux error. Based on the flux error, the flux controller generates the required *Is_d_Req* stator current. Simultaneously, the stator currents *Is_a*, *Is_b* and *Is_c* (3-phase system) are measured and transformed to the stationary reference frame α, β (2-phase system) and to the d-q rotating reference frame consecutively. The decoupling algorithm generates *Us_q* and *Us_d* voltages (d-q rotating reference frame). The *Us_q* and *Us_d* voltages are transformed back to the stationary reference frame α, β. The space vector modulation then generates the 3-phase voltage system, which is applied to the motor.

5.2.2 Drive Protection

The DCBus voltage, DCBus current and power stage temperature are measured during the control process. They are used for the overvoltage, undervoltage, overcurrent and overheating protection of the drive. The undervoltage and the overheating protection is performed by software. The overcurrent and the overvoltage fault signals utilize fault inputs of the DSP controlled by hardware. The power stage is identified via board identification. If correct boards are not identified, the "Wrong hardware" fault disables drive operation. Line voltage is measured during application initialization. According to the detected voltage level, the 115VAC or 230VAC mains is recognized. If the mains is out of the -15%.... +10% range, the “Mains out of range” fault is set, and drive operation is disabled.

If any of the mentioned faults occur, the motor control PWM outputs are disabled in order to protect the drive and the application enters the FAULT state. The FAULT state can be left only when the fault conditions disappear and the RUN/STOP switch is moved to the STOP position (in PC remote control mode by PC master software).

5.2.3 Indication of the Application States

If the application is running and motor spinning is disabled (i.e., the system is ready), the green user LED blinks at a 2Hz frequency (slower). When motor spinning is enabled, the green user LED is turned on and the actual state of the PWM outputs is indicated by PWM output LEDs. If any fault occurs (overcurrent, overvoltage, undervoltage, mains out of range, overheating or wrong hardware) the green user LED blinks at an 8Hz frequency (faster). The PC master software control page shows the identified faults. The faults can be handled by switching the RUN/STOP switch to STOP in manual operating mode or bypushing the START MOTOR/STOP MOTOR buttons to the STOP

MOTOR state in PC remote control mode to acknowledge the fault state. Meanwhile, the “Mains out of range” and “Wrong hardware” faults can be exited only with an application reset. It is strongly recommended that the user inspect the entire application to locate the source of the fault before restart.

6. Hardware Implementation

The application can run on Motorola motor control DSPs using the DSP EVM Boards with:

- DSP56F803
- DSP56F805
- DSP56F807

The hardware set-up of the system for a particular DSP varies only by the EVM Board used. Application software is identical for all DSPs.

The application can run on the following motor control platform:

- 3-phase AC induction motor

System configuration is shown in **Figure 6-1**.

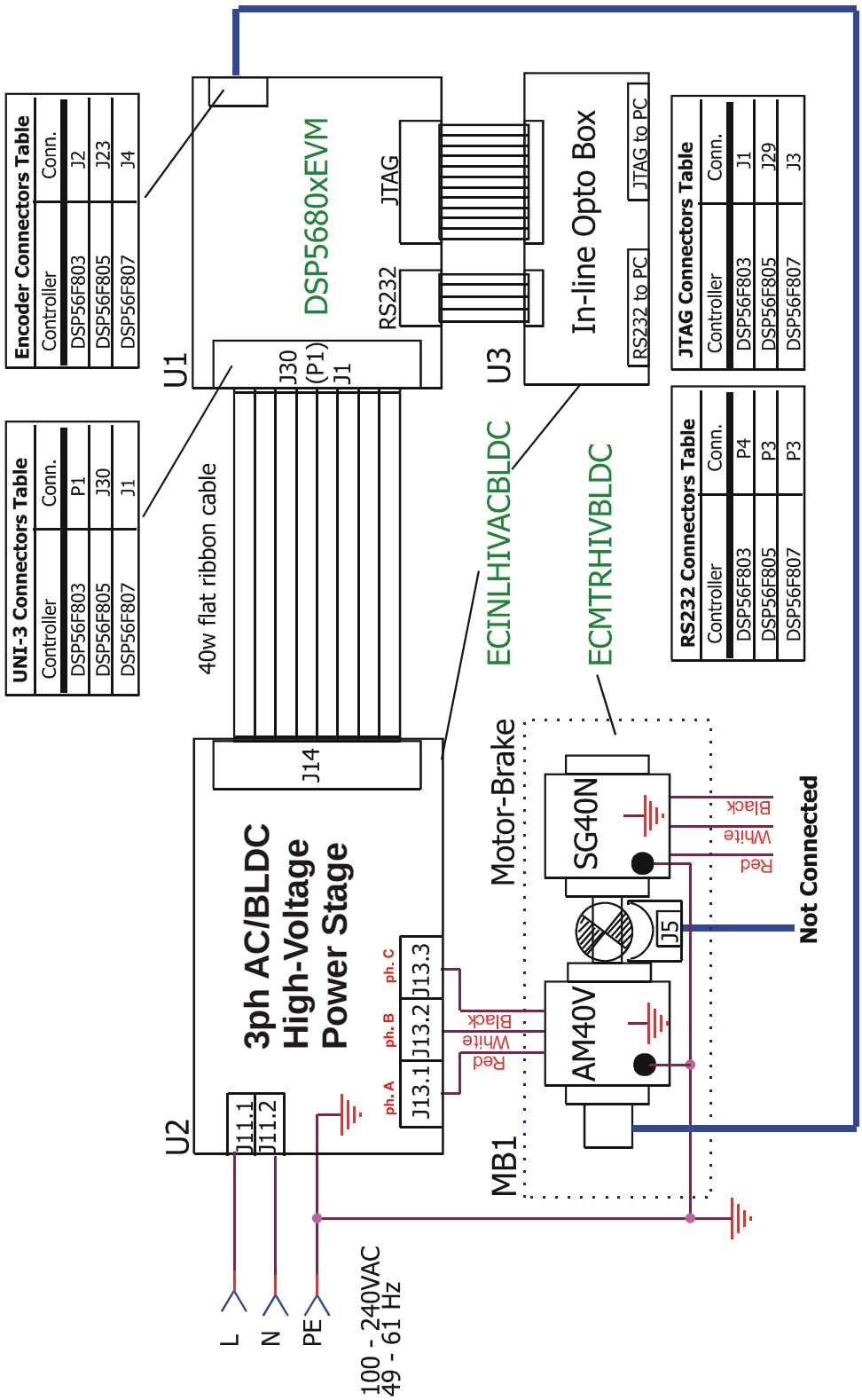

Figure 6-1. 3-Phase AC Induction Motor High-Voltage Platform Configuration

All the system parts are supplied and documented according to the following references:

- U1 - Controller Board for DSP56F805:
  - — supplied as: DSP56F805EVM
  - — described in: **DSP56F805EVMUM/D DSP Evaluation Module Hardware User's Manual**

  or U1 - Controller Board for DSP56F803:
  - — supplied as: DSP56F803EVM
  - — described in: **DSP56F803EVMUM/D DSP Evaluation Module Hardware User's Manual**

  or U1 - Controller Board for DSP56F807:
  - — supplied as: DSP56F807EVM
  - — described in: **DSP56F803EVMUM/D DSP Evaluation Module Hardware User's Manual**
- U2 - 3-phase AC/BLDC High Voltage Power Stage
  - — supplied in kit with In-Line Optoisolation Box as: ECINLHIVACBLDC
  - — described in: **MEMC3BLDCPSUM/D - 3-phase AC/BLDC High Voltage Power Stage**
- U3 - In-Line Optoisolation Box
  - — supplied in kit with 3-phase AC/BLDC High Voltage Power Stage as: ECINLHIVACBLDC, or separately as ECOPTINL
  - — described in: **MEMCILOBUM/D - In-Line Optoisolation Box**

Warning: The user must use the In-line Optoisolation Box during development to avoid damage to the development equipment.

- **MB1** Motor-Brake AM40V + SG40N
  - — supplied as: ECMTRHIVAC

Detailed descriptions of individual boards can be found in comprehensive User's Manuals belonging to each board or on the Motorola web **http://www.motorola.com**. The User's Manual incorporates the schematic of the board, description of individual function blocks and a bill of materials. An individual board can be ordered from Motorola as a standard product.

The AC induction motor-brake set incorporates a 3-phase AC induction motor and attached BLDC motor brake. The AC induction motor has four poles. The incremental position encoder is coupled to the motor shaft, and position Hall sensors are mounted between motor and brake. They allow sensing of the position if required by the control algorithm. Detailed motor-brake specifications are listed in **Table 6-1**.

Table 6-1. Motor - Brake Specifications

Set Manufactured	EM Brno, Czech Republic	
Motor Specification:	Motor Type:	AM40V 3-Phase AC Induction Motor
	Pole-Number:	4
	Nominal Speed:	1300 rpm
	Nominal Voltage:	3 x 200 V
	Nominal Current:	0.88 A
Brake Specification:	Brake Type:	SG40N 3-Phase BLDC Motor
	Nominal Voltage:	3 x 27 V
	Nominal Current:	2.6 A
	Pole-Number:	6
	Nominal Speed:	1500 rpm
Position Encoder	Type:	Baumer Electric BHK 16.05A 1024-12-5
	Pulses per Revolution:	1024

7. Software Implementation

This section describes the software design of the AC induction vector control drive application. First, the numerical scaling in fixed-point fractional arithmetic of the DSP is discussed. Then, the control software is described in terms of:

- Software Flowchart
- Control Algorithm Data Flow
- State Diagram

Finally, particular issues such as speed and current sensing are explained. The aim of the chapter presented is to facilitate understanding of the designed software.

7.1 Analog Value Scaling

The AC induction motor vector control application uses a fractional representation for all real quantities, except time. The N-bit signed fractional format is represented using 1.[N-1] format (1 sign bit, N-1 fractional bits). Signed fractional numbers (SF) lie in the following range:

$$-1.0 \leq SF \leq +1.0 - 2^{-[N-1]} \quad \text{(EQ 7-1)}$$

For words and long-word signed fractions, the most negative number that can be represented is -1.0, whose internal representation is $8000 and $80000000, respectively. The most positive word is $7FFF or $1.0 - 2^{-15}$, and the most positive long-word is $7FFFFFFF or $1.0 - 2^{-31}$

The following equation shows the relationship between a real and a fractional representation:

$$\text{Fractional Value} = \frac{\text{Real Value}}{\text{Real Quantity Range}} \quad \text{(EQ 7-2)}$$

7.1.1 Voltage Scaling

Voltage quantities are scaled to the maximum measurable voltage, which is dependent on the hardware. The relationship between real and fractional representations of voltage quantities is:

$$u_{Frac} = \frac{u_{Real}}{u_{Max}} \quad \text{(EQ 7-3)}$$

where:

u_{Frac}	fractional representation of voltage quantities	[-]
u_{Real}	real voltage quantities in physical units	[V]
u_{Max}	maximum defined voltage used for scaling in physical units	[V].

In the application, the u_{Max}value is the maximum measurable DC-bus voltage:

u_{Max} = 407 V

Other application voltage variables are scaled in the same way (`u_dc_bus`, `u_dc_bus_filt`, `u_SAlphaBeta`, `u_SDQ_ref`, `u_SDQ`, `u_Sabc`, `u_Samplitude`, etc.).

7.1.2 Current Scaling

The current quantities are scaled to the maximum measurable current, which is dependent on the hardware. The relationship between real and fractional representation of current quantities is:

$$i_{Frac} = \frac{i_{Real}}{i_{Max}} \quad \text{(EQ 7-4)}$$

where:

i_{Frac}	fractional representation of current quantities	[-]
i_{Real}	real current quantities in physical units	[A]
i_{Max}	maximum defined current used for scaling in physical units	[A].

In the application, the i_{Max} value is the maximum measurable current:

i_{Max} = 5.86 A

Other application current variables are scaled in the same way (`i_Sabc_comp`, `i_SAlphaBeta`, `i_Sphase_max`, `i_SD_desired`, `i_SQ_desired`, etc.).

7.1.3 Flux Scaling

Magnetic flux quantities are scaled to the maximum motor flux, which is dependent on the motor used. The maximum flux can be expressed as:

$$\Psi_{Max} \approx C_{sf} \cdot \frac{60 \cdot \sqrt{2}}{2 \cdot \pi \cdot \sqrt{3}} \cdot \frac{u_{nom}}{p_p \cdot n_s} \quad \text{(EQ 7-5)}$$

where:

Ψ_{Max}	maximum calculated flux value used for scaling in physical units	[Vs]
u_{nom}	nominal line-to-line voltage of motors	[V]
n_s	motor-synchronous speed dependent on pair of poles	[rpm]
p_p	number of pole pairs	[-]
C_{sf}	safety margin constant	[-]

The relationship between real and fractional representation of flux quantities is:

$$\Psi_{Frac} = \frac{\Psi_{Real}}{\Psi_{Max}} \quad \text{(EQ 7-6)}$$

where:

Ψ_{Frac}	fractional representation of flux quantities	[-]
Ψ_{Real}	real flux quantities in physical units	[Vs]

In the application, the parameters for Ψ_{Max} calculation are:

u_{nom} = 200 V

n_s = 1500 rpm

p_p = 2

C_{sf} = 1.92

The maximum motor flux value is then:

Ψ_{Max} = 1 Vs

Other application flux variables are scaled in the same way (`psi_RAlphaBeta`, `psi_RD_desired`, etc.).

7.1.4 Speed Scaling

Speed quantities are scaled to the defined maximum mechanical speed, which is dependent on the drive. The relationship between real and fractional representation of speed quantities is:

$$\omega_{Frac} = \frac{\omega_{Real}}{\omega_{Max}} \quad \text{(EQ 7-7)}$$

where:

ω_{Frac}	fractional representation of speed quantities	[-]
ω_{Real}	real speed quantities in physical units	[rpm]
ω_{Max}	maximum defined speed used for scaling in physical units	[rpm].

In the application, the ω_{Max} value is defined as:

ω_{Max}= 4000 rpm

Other speed variables are scaled in the same way (`omega_reqPCM_mech`, `omega_desired_mech`, `omega_required_mech`, `omega_reqMAX_mech`, `omega_reqMIN_mech`, `omega_actual_mech`).

7.2 Software Flowchart

The general software flowchart incorporates the main routine entered from reset and interrupt states. The overview of the software flowchart is shown in **Figure 7-1**.

After reset, the main routine provides initialization of the drive parameters, the application and the DSP; it then enters an endless background loop. The background loop contains the routines: Fault Detection, Start/Stop Switch and Required Speed Scan, Brake Control and Application State Machine.

The following interrupt service routines (ISRs) are utilized:

- **PWMA Fault ISR** services faults invoked by external hardware fault
- **ADC End of Scan ISR** services ADC and provides the execution of the fast control loop; the ADC is synchronized with the PWM pulses. The PWM value registers are updated here. It is invoked with a 125µs period.
- **Timer C, Channel 0 On Compare ISR** provides the execution of the slow control loop, LED indication processing, push button processing and switch filtering; it is invoked with a 1000µs period.
- **SCI ISR** services PC master software communication
- **IRQA ISR** services the UP Push Button
- **IRQB ISR** services the DOWN Push Button

7.2.1 Initialization

Initialization occurs after reset. The first phase of initialization of the DSP peripherals is done through the `appconfig.h` file defines. The next phase is done in the application code. The drive parameters are set, then the application and DSP initializations are executed.

Initialization performed by the `appconfig.h`:

- SDK drivers are included/excluded
- DSP56F80X chip revision is defined (A,B or D). It is not necessary to define it for other revisions, since it is required only for elimination of the offset and gain errors of the first DSP versions.
- PWMA module is initialized
  - — Center-aligned complementary PWM mode, positive polarity
  - — Set PWM modulus: defines the PWM frequency as 16kHz
  - — Deadtime is set to 1µs
  - — Output pads are disabled
  - — PWM faults are disabled

- ADC module is initialized
  - ADC is triggered simultaneously
  - Conversion started by SYNC pulse
  - Associate interrupt service routine with ADC end of scan event
  - Defines ADC samples
- Timers defined to be used by the application
  - Timer A channels 0, 1, 2, 3
  - Timer C channels 0, 2
- IRQA and IRQB interrupts are initialized
  - Interrupt service routine is associated with the IRQA and IRQB
  - Interrupt priority is set
- Interrupt priorities are set in descending order from highest to lowest
  - PWMA fault interrupt
  - ADC end of scan interrupt
  - Timer C channel 0 On Compare interrupt
  - SCI interrupts
  - IRQA, IRQB interrupts
- PC master software recorder is initialized; see **Section 8.5**

For more information on the `appconfig.h` file, see **Section 8.2**.

The following drive parameters are set in the `DriveParamSet` routine:

- The output voltage structure is initialized to zero volts
- Parameters of the AC induction flux model are set
  - Integration state variables are reset
  - Motor-dependent constants are set
- Parameters of the d-q establishment algorithm are set
  - Rotor flux zero limit value is initialized
  - Motor-dependent constants are set
- Parameters of the decoupling algorithm are set
  - Motor-dependent constants are set
- Parameters of the torque- and flux-producing current components controllers and speed, flux and field-weakening controllers are set
  - Proportional and integral gain and their scaling constants are set
  - Controller output limits are set
  - Controller integral portion is reset to zero
- Currents limitation algorithm parameter is set
  - Maximum motor-current value is set

- States of the application state machine are set as follows:
  - Application state is set to INIT
  - Substate of application RUN state is set to DE-EXCITATION
  - Substate of application INIT state is set to BRANCH
- Application operating mode is set to MANUAL
- `PrimaryCtrl` bit in `appInitControl` control word is set
- Start/Stop switch, switch filter and overload filter are initialized

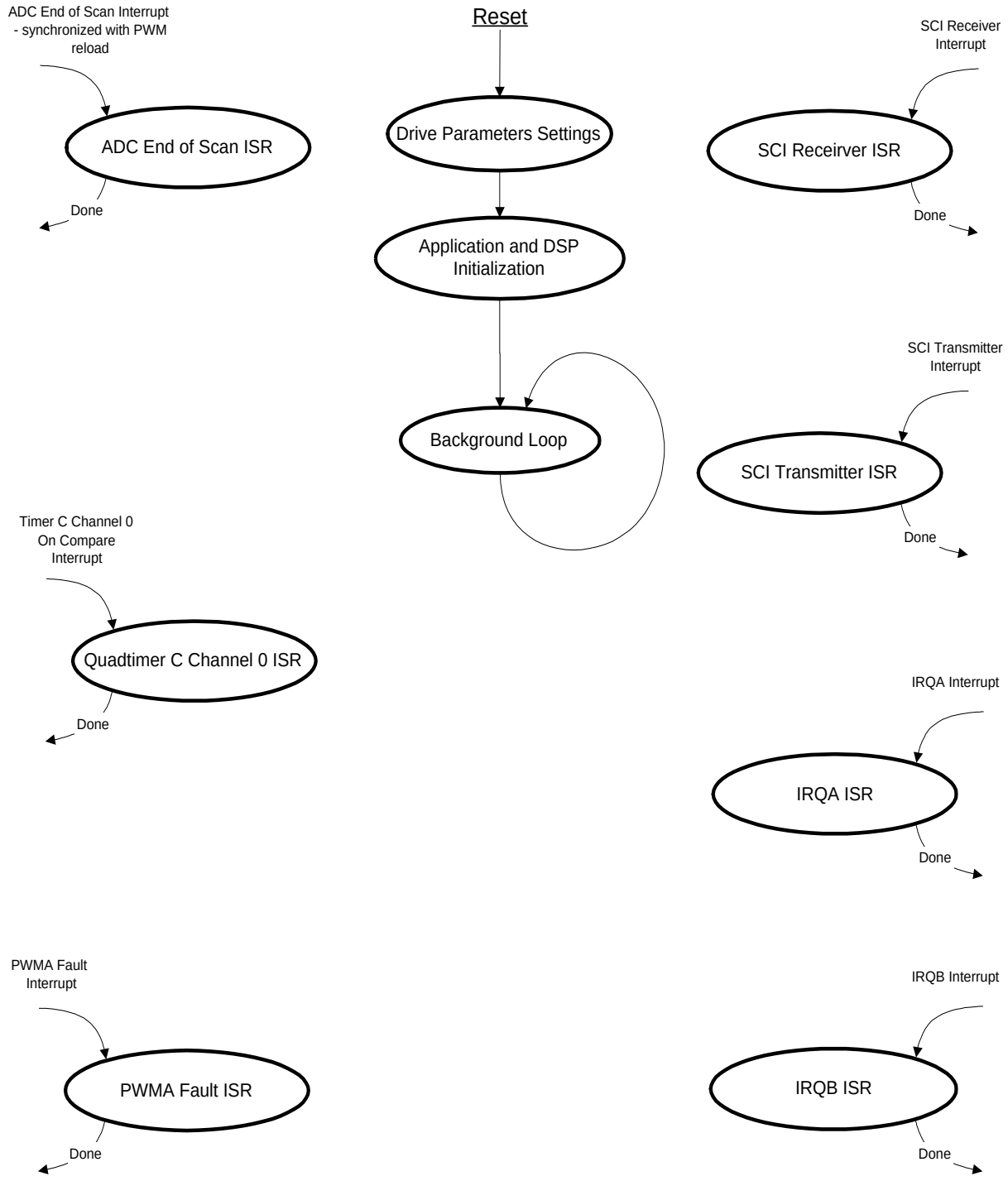

Figure 7-1. Software Flowchart Overview

After initialization of the drive parameters is completed, the application and DSP initialization routine is executed:

- ADC channels are assigned to the sensed quantities
  - — ADC channel 2 to sample 0 - Phase current A
  - — ADC channel 3 to sample 1 - Phase current B
  - — ADC channel 4 to sample 2 - Phase current C
  - — ADC channel 0 to sample 4 - DCBus voltage
  - — ADC channel 5 to sample 4,5,7 - power module temperature
- Quad Timer C channel 0 driver initialization (slow control loop time base)
  - — Count Up
  - — Prescaler 2
  - — Interrupt On Compare (compare value set to 1000μs period)
  - — Associate interrupt service routine with On Compare event
- Quad Timer C channel 2 driver initialization (ADC and PWM synchronization)
  - — Count Up
  - — Prescaler 1
  - — Started by PWM reload signal
- Switch control is initialized
- PWMA fault interrupt service routine is initialized
- Brake control is initialized
- Power stage board identification
  - — Identifies hardware set connected to the EVM board
- Speed and position measurement is initialized
  - — Quad Timer A channels 0, 1, 2, 3 initialized for speed and position measurement. The position measurement (Quad Timer A channel 1) is not applied in the application.
  - — Speed measurement-specific variables are initialized
- Status LEDs control is initialized
- Quad Timer C channel 0 is enabled
- Push button control is initialized
- Interrupts are enabled

7.2.2 Background Loop

After initialization, the background loop is entered. It runs in an endless loop and is asynchronously interrupted by the system interrupt service routines. The processes executed in the background are:

- Fault Detection
  - Fault DCBus overvoltage and overcurrent pins are scanned for a fault signal occurrence
  - Measured DCBus voltage in `u_dc_bus_filt` is checked for undervoltage
  - Measured power module temperature in `temperature_filt` is checked for overheating
  - Mains detection fault flag is checked
  - Hardware identification fault flag is checked
  - Drive overload fault is detected
  - When a fault occurs, the appropriate bits in `appFaultStatus` and `appFaultPending` words are set. The `FaultCtrl` bit in `appControl` is set to change application state to FAULT.
- Start/Stop Switch and Required Speed Scan
  - Based on the application operating mode, the process selects whether the required speed and start/stop command are set manually with the switches and buttons or by the PC master software interface. The required speed is limited to maximum and minimum values.
- Brake Control Background
  - Sets the generator mode flag if the drive is running in the generator mode. If the drive is in motor mode, the brake switch is turned off.
- Application State Machine
  - Ensures the execution of the active application state and the transition between the states, according to bits in the application control word.

7.2.3 ADC End of Scan ISR

The ADC End of Scan ISR is the most critical and the routine most demanding of the processor's time. Most of the AC induction motor vector control processes must be linked to this ISR.

The Analog-to-Digital Converter is initiated synchronously with a PWM reload pulse. It simultaneously scans phase currents, phase voltage and temperature. When the conversion is finalized, the ADC End of Scan ISR is called. The PWM reload pulse frequency is set to every second PWM opportunity. For the PWM frequency of 16kHz, this means the PWM reload pulse frequency is 8kHz, which corresponds to the 125µs ADC End of Scan ISR period.

The routine calls control functions according to application state. If the application state is RUN, the `FastControlLoopEnabled` function is called; otherwise, the `FastControlLoopDisabled` function is called. The ADC End of Scan diagram is shown in **Figure 7-2**.

The `FastControlLoopEnabled` function provides the following services and calculations:

- Sets a compare value for QuadTtimer C channel 2, defining the ADC start, needed for phase current measurement
- Calls the analog-sensing and correction function
- Calls the forward Clarke transformation
- Calls the rotor flux model calculation

- Calls the d-q system establishment function
- Calls i_{sd} and i_{sq} current-component controllers
- Calls the decoupling algorithm
- Calls the inverse Park transformation
- Calls the DCBus ripple elimination function
- Calls the space vector modulation function
- Calls the analog-sensing correction reconfiguration function
- Passes calculated duty cycle ratios to the PWM driver
- Calls the brake control function

The `FastControlLoopDisabled` function is called in the application states when the vector control algorithm is not executed. The function services only the analog-sensing correction process, space vector modulation algorithm and PWM generation. The drive control variables are set to their initial values.

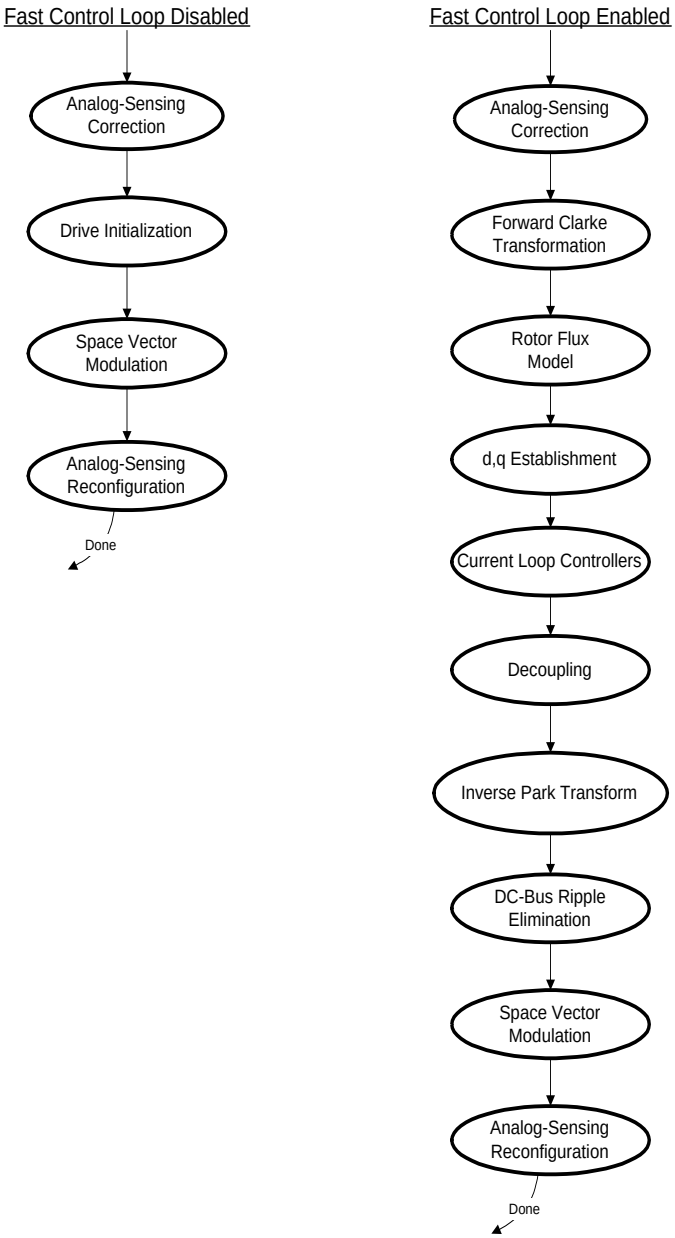

Figure 7-2. ADC End of Scan ISR

7.2.4 Quad Timer C, Channel 0, On Compare ISR

The routine calculates part of the vector control algorithm and handles LED indication, button processing and switch filtering. It is called with a 1000µs period. The tasks provided by individual functions are:

- Slow control loop is executed. It provides the part of vector control algorithm calculations, which can be executed in a slower control loop. The function `SlowControlLoopEnabled` is called.
  - — Reads the actual motor speed and handles the speed measurement process
  - — Executes the speed acceleration/deceleration ramp algorithm
  - — Calculates the output stator voltage amplitude
  - — Field-weakening controller is called
  - — Rotor flux and speed controllers are called
  - — Current limit algorithm is called
- LED indication process handles the LED indication of the application state (INIT, RUN, STOP, FAULT)
- Button processing handles the UP/DOWN button debounce counter
- Switch-filter processing handles the Start/Stop switch filtering
- PC master software recorder routine is called

7.2.5 PWMA Fault ISR

The **PWMA Fault ISR** is the highest priority interrupt implemented in the software. In the case of DCBus, overcurrent or overvoltage fault detection, the external hardware circuit generates a fault signal that is detected on the fault input pin of the DSP's PWMA module. The signal disables PWM outputs in order to protect the power stage and generates a fault interrupt where the fault condition is handled. The routine sets the records of the corresponding fault source to the fault status word and sets the fault bit in the application control word.

7.2.6 SCI ISR

The interrupt handler provides SCI communication and PC master software service routines. These routines are fully independent of the motor control tasks.

7.2.7 IRQA and IRQB ISR

Push button interrupt handlers take care of the push button service. The UpButton ISR sets the UP button flag and the DownButton ISR sets the DOWN button flag. The desired speed is incremented/decremented according to the debounced UP/DOWN button flag.

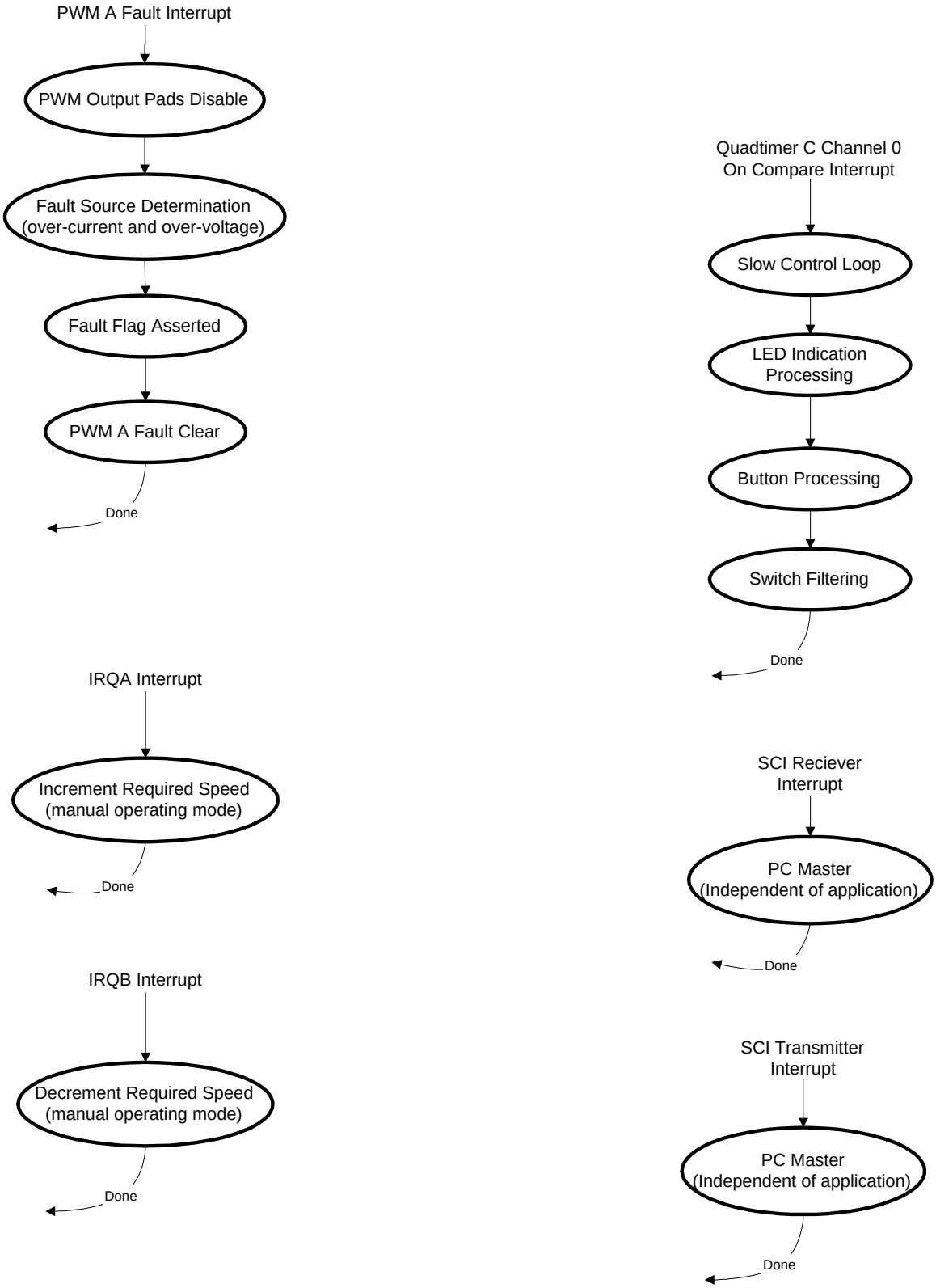

Figure 7-3. Application Interrupt Service Routines

7.3 Control Algorithm Data Flow

The 3-phase AC induction motor vector control algorithm data flow is described in **Figure 7-4**, **Figure 7-5** and **Figure 7-6**.

The individual processes are described in detail in the following sections.

7.3.1 Analog-Sensing Corrections

The analog-sensing process handles sensing, filtering and correction of analog variables (phase currents, temperature, DCBus voltage).

7.3.2 Speed Measurement

The speed measurement process provides the mechanical angular speed, `omega_actual_mech`.

7.3.3 Forward Clarke Transformation

The forward Clarke transformation transforms the 3-phase system a,b,c to a 2-phase orthogonal reference frame α,β. For theoretical background, see **Section 4.2**. The algorithm is included in the SDK motion control function library. For more details, refer to the SDK library manual.

7.3.4 Rotor Flux Model

The rotor flux model process calculates the rotor magnetic flux of the AC induction motor in the (α, β) 2-phase stationary reference frame. The flux model utilizes monitored rotor speed and stator voltages and currents. For theoretical background, see **Section 4.4**. The algorithm is included in the SDK motion control function library. For more details, refer to the SDK library manual.

7.3.5 d-q System Establishment

This process transforms quantities from an (α, β) 2-phase reference frame attached to the stator into a d-q-) 2-phase reference frame rotating with the magnetic flux angular speed. The rotor magnetic flux space vector is put into the *d* axis of the coordinate system. The function calculates the magnitude of the rotor magnetic flux and the sine and cosine of its position angle `theta_field` in the (α, β) coordinate system. For theoretical background, see **Section 4.3**. The algorithm is included in the SDK motion control function library. For more details, refer to the SDK library manual.

7.3.6 Decoupling

The decoupling process calculates the decoupling rotational voltage components of the AC induction machine in the d-q coordinate system and adds them to the outputs of the currents controllers which control the i_{sd} and i_{sq} components. It yields to the d and q output stator voltage components. The output voltage vector is limited to the desired limits. For theoretical background, see **Section 4.5**. The algorithm is included in the SDK motion control function library. For details, refer to the SDK library manual.

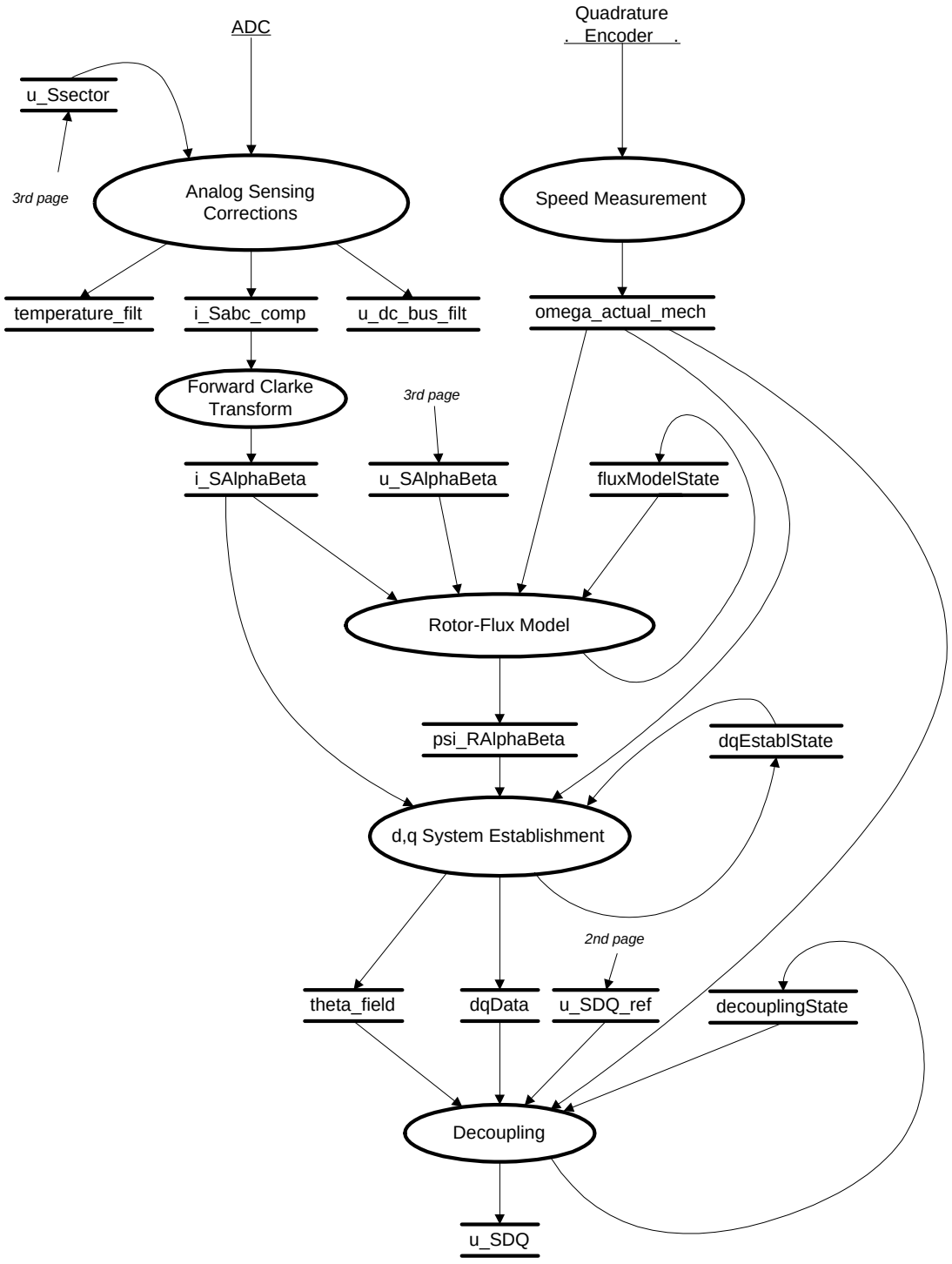

Figure 7-4. Vector Control Application Data Flow

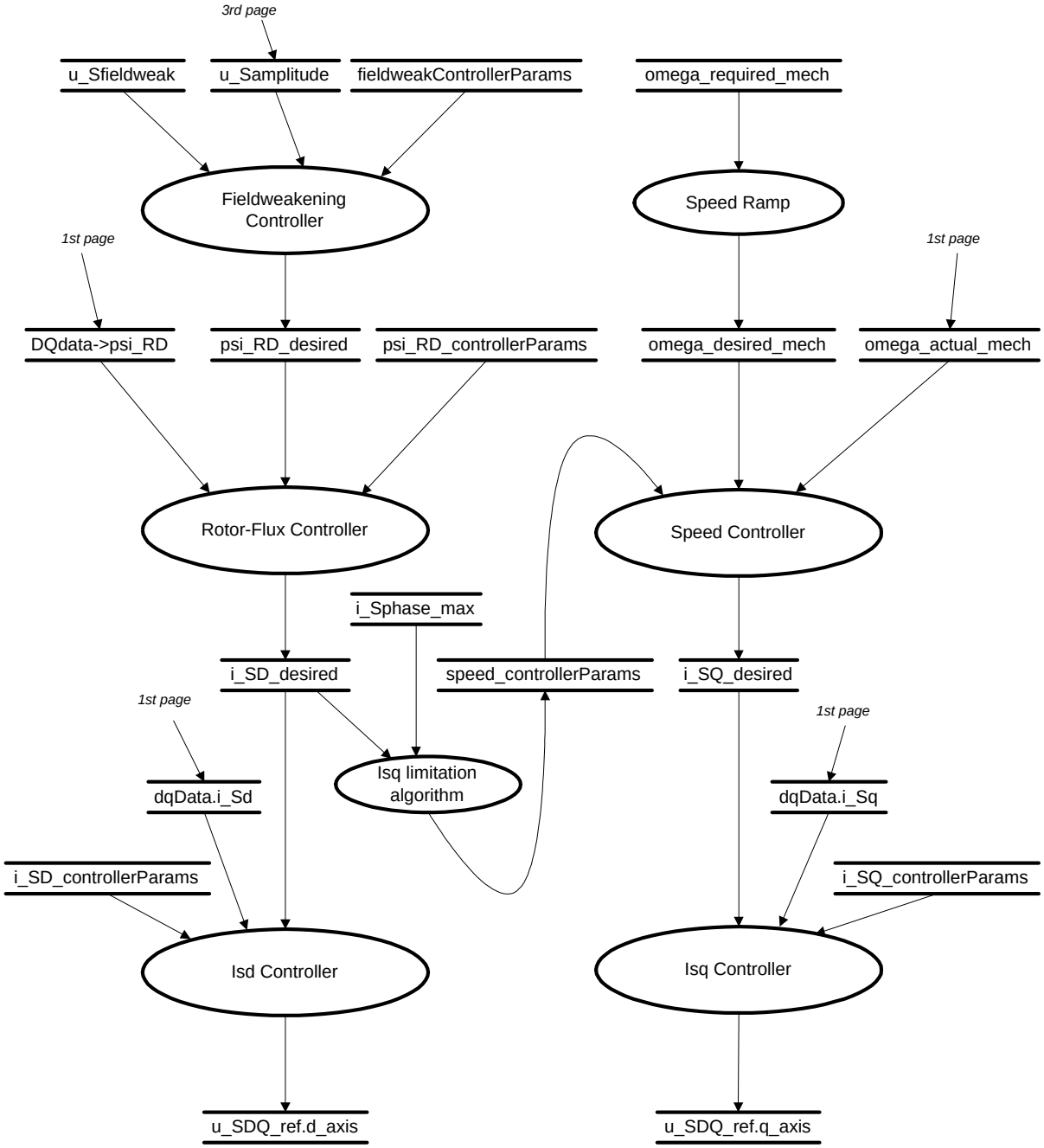

Figure 7-5. Controllers Data Flow Chart

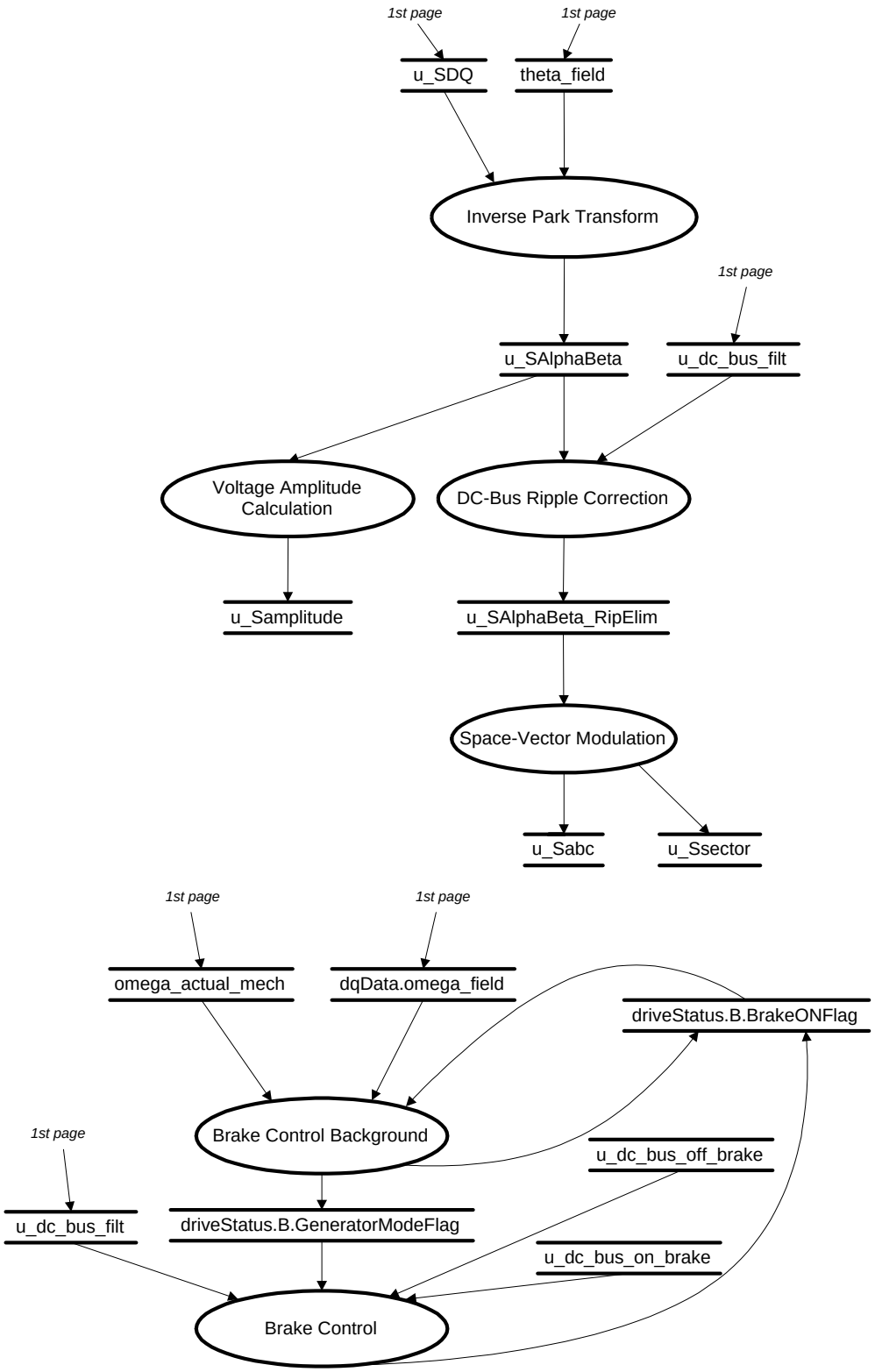

Figure 7-6. Space Vector Modulation and Brake Control Data Flow

7.3.7 Speed Ramp

This process calculates the desired speed (`omega_desired_mech`), based on the required speed according to the acceleration/deceleration ramp. The required speed (`omega_required_mech`) is determined either by the push buttons, if in manual mode, or by PC master software, if in PC remote control mode.

7.3.8 Speed Controller

This process calculates the desired i_{sq} stator current component (`i_SQ_desired`) according to the speed error, which is the difference between the actual and desired speeds. The PI controller is implemented.

7.3.9 I_{sq} Controller

This process calculates the linear portion of the stator voltage space vector *q* component (`u_SDQ_ref.q_axis`) based on the i_{sq} stator current component error, which is the difference between the actual and desired i_{sq} stator current components. The PI controller is implemented.

7.3.10 Field-Weakening Controller

The field-weakening process provides control of the desired rotor flux (`psi_RD_desired`) in order to achieve a higher motor speed than nominal. It compares the actual output motor stator-voltage amplitude with nominal field-weakening voltage; the desired rotor flux is set based on the calculated error.

7.3.11 Flux Controller

This process calculates the desired i_{sd} stator current component (`i_SD_desired`) according to rotor flux error, which is the difference between the actual and desired rotor flux. The PI controller is implemented.

7.3.12 I_{sd} Controller

This process calculates the linear portion of the stator voltage space vector *d* component (`u_SDQ_ref.d_axis`), based on the i_{sd} stator current component error, which is the difference between the actual and desired i_{sd} stator current components.

7.3.13 Inverse Park Transformation

The Inverse Park Transformation process converts stator voltage space vector components from the rotating orthogonal coordinate system (d-q) attached to the rotor magnetic flux to the stationary orthogonal coordinate system (α,β) attached to the stator. For theoretical background, see **Section 4.3**. The algorithm is included in the SDK motion control function library. For more details, refer to the SDK library manual.

7.3.14 DCBus Ripple Elimination

This process provides for the elimination of the voltage ripple on the DCBus. It compensates an amplitude of the direct-α and the quadrature-β components of the stator reference voltage vector U_S for imperfections in the DCBus voltage. The algorithm is included in the SDK motion control function library. For more details, refer to the SDK library manual.

7.3.15 Space Vector Modulation

This process directly transforms the stator voltage space vector from the α,β coordinate system to pulse width modulation (PWM) signals (duty cycle values). The duty cycle ratios are then passed to the PWM module in the `u_Sabc` structure. For theoretical background, see **Section 4.6**. The algorithm is included in the SDK motion control function library. For details, refer to the SDK library manual.

7.3.16 Voltage Amplitude Calculation

This process provides a calculation of the actual stator voltage space vector magnitude from the d-q components of the stator voltage. The actual stator voltage amplitude is used in field-weakening. It is the value controlled by the field-weakening controller.

7.3.17 Brake Control Background

This process is executed in the background. It sets the generator mode flag if the drive is running in generator mode. If the drive is in motor mode, the generator mode flag is cleared. In motor mode, if the brake-on flag is set, the brake switch is turned off and the brake-on flag is cleared.

7.3.18 Brake Control

This process is executed in the ADC End of Scan ISR. If the generator mode flag is set, switching of the brake switch is enabled. The brake switch is turned on if the DCBus voltage is higher than `u_dc_bus_on_brake` and turned off if it is lower than `u_dc_bus_off_brake`. The brake-on flag is set if the switch is on and cleared if it is off.

Notes: Constants of controllers were designed using standard control theory in a continuous time domain. The step responses of the controlled system measured by the PC master software were used to investigate system parameters. The least-square method, programmed in Matlab, identified the respective system parameters in the Laplace domain. The parameters were designed using standard Matlab functions, such as the Bode plot of frequency response, Nyquist plot, step response, etc. The results in the continuous time domain were then transformed to the discrete time domain for DSP usage. In the application, the controller parameters were tuned slightly.

7.4 Application State Diagram

The processes described above are implemented in the state machine, as illustrated in **Figure 7-7**. The state machine provides transitions between the states INIT, STOP, RUN, FAULT.

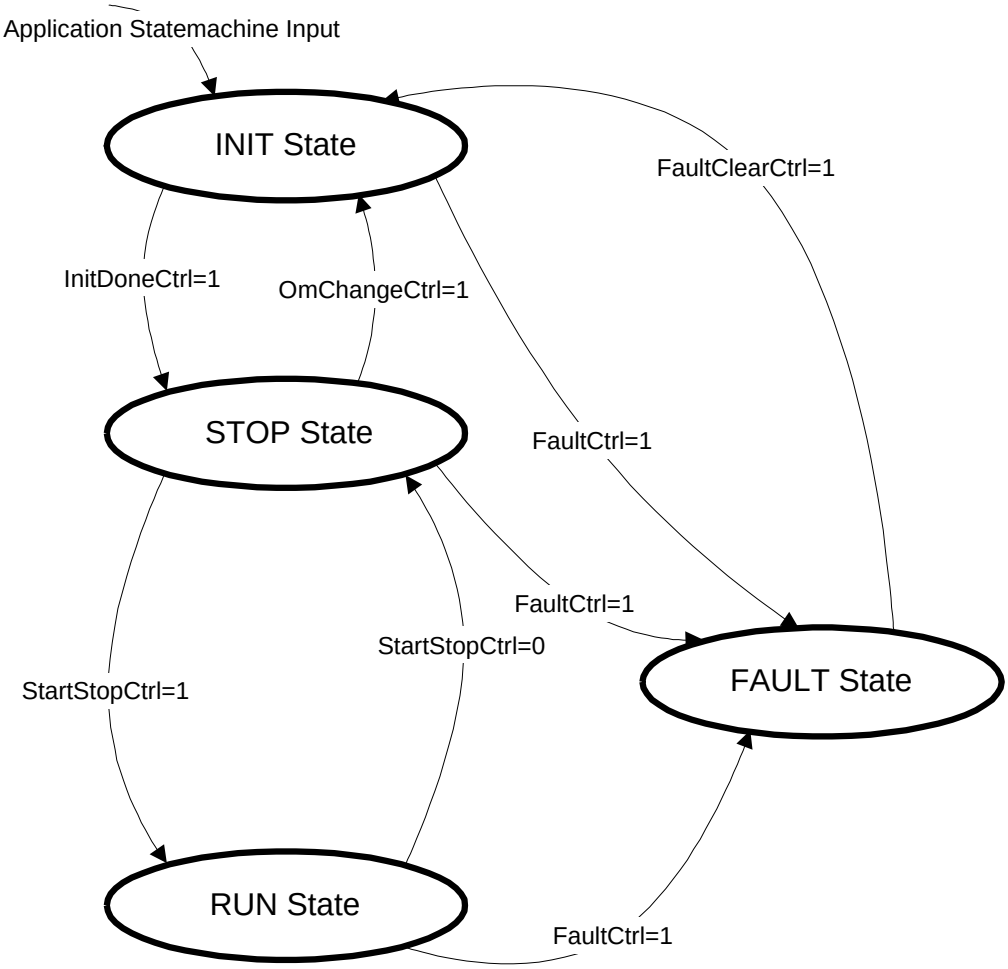

Figure 7-7. Application State Diagram

7.4.1 Application State - INIT

After RESET, the application enters the INIT state, which provides DSP and application initialization. In this state, the drive is disabled and the motor cannot be started. The INIT state is divided into three substates which handle the different phases of the INIT state. The substates of the INIT state are illustrated in **Figure 7-8**. The tasks provided by the INIT substates are:

- The BRANCH substate decides whether or not the primary initialization is executed. It is entered any time there is a transition from any other state to the INIT state. It is entered just once after the INIT state is set. It calls the transition function to either the PRIMARY or OPERATING MODE substates.
- The PRIMARY substate provides the primary initialization of the DSP and the application. It is entered from the BRANCH substate after the application is reset or after a transition from a FAULT to an INIT application state. In the transition from the BRANCH to the PRIMARY

substate, analog-sensing correction initialization is started. After the initialization is finished, mains detection is executed and the state is changed to the OPERATING MODE substate.

- The OPERATING MODE substate handles the operating mode change logic. It is entered from the BRANCH or PRIMARY substates and sets the actual operating mode (MANUAL or PC_MASTER).This state can be exited only if the RUN/STOP switch is in the stop position and the application transits to the STOP state. If the switch is in the start position, the application remains in the INIT state; it serves as protection against start after reset if the RUN/STOP switch is in the start position.

If any fault is detected, the application transits to the FAULT state (protection against fault).

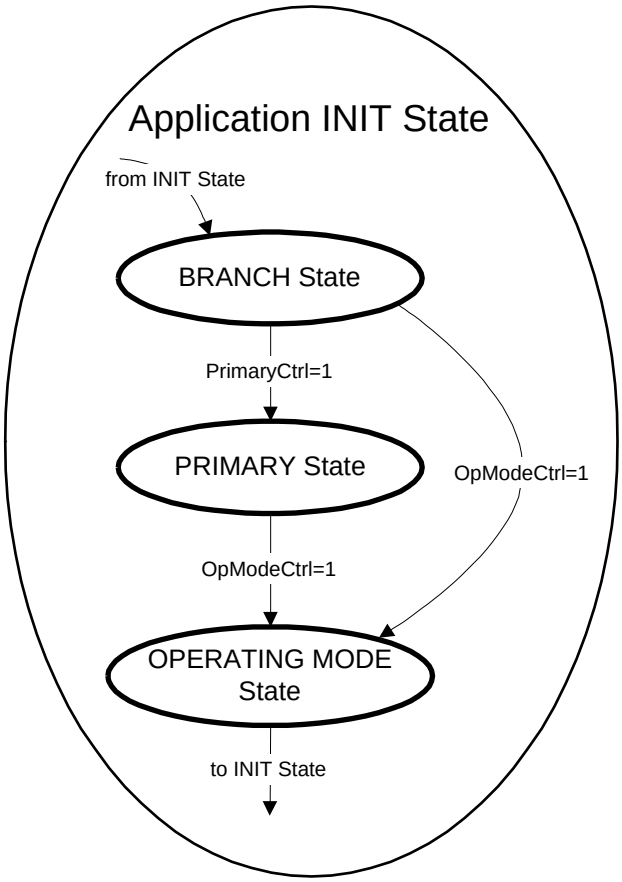

Figure 7-8. INIT State Substates State Machine

7.4.2 Application State - STOP

The STOP state can be entered either from the INIT state or the RUN state. The STOP state provides a motor standstill. In the STOP state, the drive is disabled, PWM output pads are disabled and the `FastControlLoopDisabled` function is called by the ADC End of Scan ISR. The application waits for the start command.

When the application is in the STOP state, the operating mode can be changed, either from manual mode to PC master software mode, or vice versa. When the operating mode request is asserted, the application always transits to the INIT state, where the mode is changed.

If a fault is detected in the STOP state, the application enters the FAULT state (fault protection). If no fault is present and the start command is accepted, the application transits to the RUN state and the motor is started.

7.4.3 Application State - RUN

The RUN state can be entered from the STOP state. The RUN state performs a motor spinning. In the RUN state, the drive is enabled and the motor runs. The PWM output pads are enabled and the `FastControlLoopEnabled` function is called by the ADC End of Scan ISR. The RUN state is divided into three substates which handle the different phases of the RUN state. The RUN substates' state machine is illustrated in **Figure 7-9**. The tasks provided by the RUN substates are:

- The EXCITATION substate provides the excitation of the motor during start-up. It is entered after the transition from the STOP stat; motor excitation is then enabled. After the motor is excited to the nominal rotor flux value, the substate is changed to SPINNING. If the stop command is accepted before the motor is fully excited, the substate is changed to DE-EXCITATION.
- The SPINNING substate provides motor spin and acceleration/deceleration. It is entered from the EXCITATION substate. The required speed command is accepted, and the motor spins at the required speed. If a stop command is accepted, the substate changes to DE-EXCITATION.
- The DE-EXCITATION substate provides de-excitation as the motor is going to the STOP state. It is entered from the EXCITATION or SPINNING substates. The speed command is set to zero turns. When zero turns are reached, motor de-excitation is executed. If the motor is deexcited, the application transits to the STOP state.

If any fault in the RUN state is detected, the application enters the FAULT state (fault protection).

7.4.4 Application State - FAULT

The FAULT state can be entered from any state. In the FAULT state, the drive is disabled and the application waits for the faults to be cleared.

When it detects that the fault has disappeared and the fault clear command is accepted, the RUN/STOP switch is moved to the stop position and the application transits to the INIT state. The "Wrong hardware" and "Mains out of range" faults can only be cleared by reset.

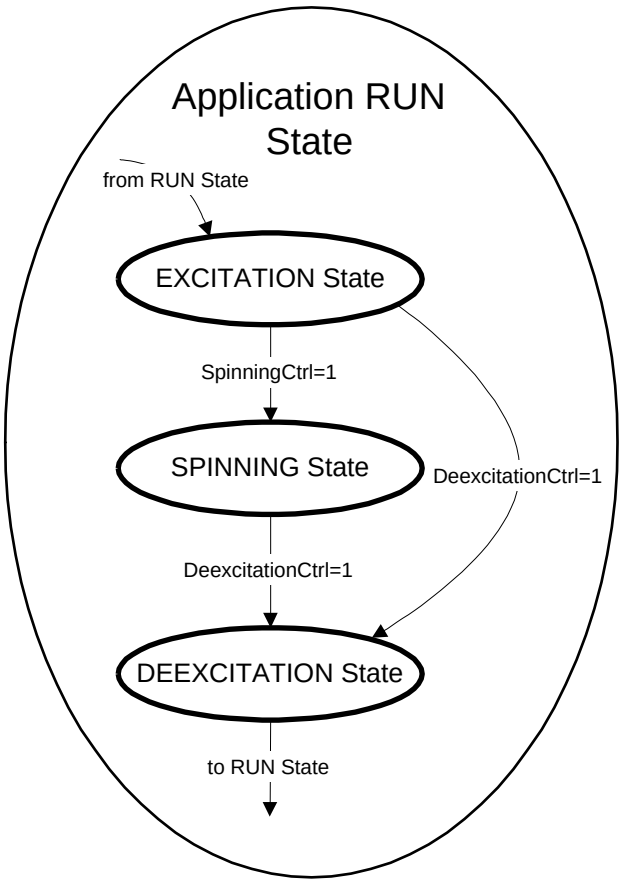

Figure 7-9. RUN State Substates State Machine

7.5 Speed Sensing

All members of the Motorola DSP56F80x family except the DSP56F801, contain a quadrature decoder module. This peripheral is commonly used for position and speed sensing. The quadrature decoder position counter counts up/down at each edge of the Phase A and Phase B signals according to their order; see **Figure 7-10**.

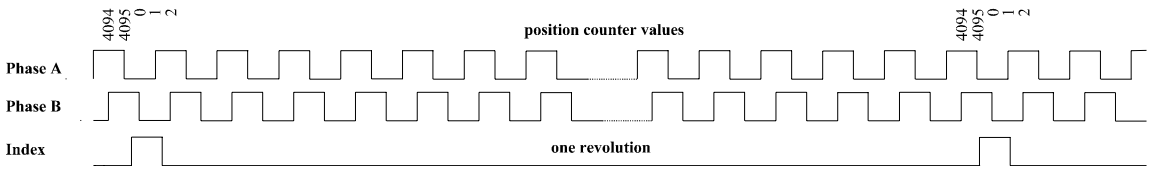

Figure 7-10. Quadrature Encoder Signals

In addition, the quadrature decoder input signals (Phase A, Phase B and Index) are connected to Quad Timer module A. The quad timer module contains four identical counter/timer groups. Due to the wide variability of quad timer modules, it is possible to use this module to decode quadrature encoder signals and to sense position and speed. The application presented uses the quad timer approach for speed measurement in order to be able to easily accommodate the software for the DSP56F801, which

does not have a quadrature decoder module. The configuration of the quad timer module is shown in **Figure 7-11**. The presented configuration is ready for position sensing handled by timer A1. In the AC induction motor vector control application presented, however, position sensing is not applied.

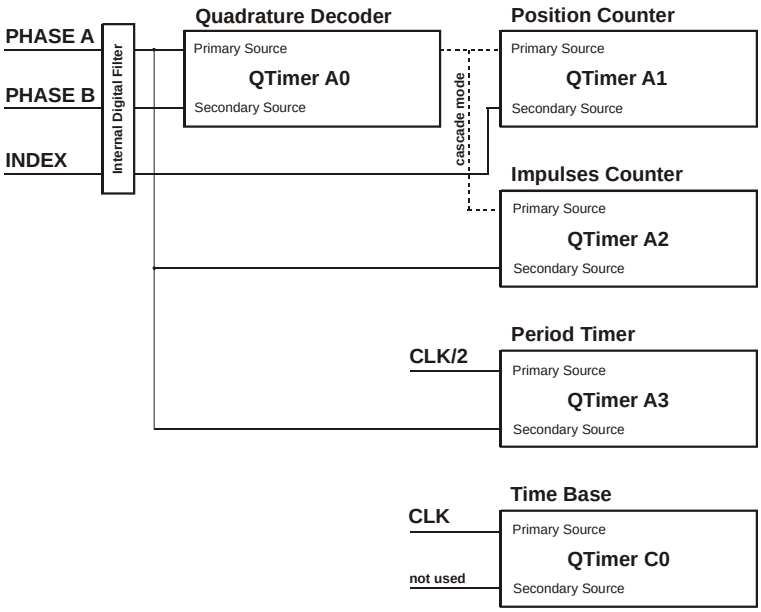

Figure 7-11. Quad Timer Module A Configuration

7.5.1 Speed Sensing

There are two common ways to measure speed. The first method measures the time between two following edges of the quadrature encoder; the second method measures the position difference per constant period. The first method is used at low speed. At higher speeds, when the measured period is very short, the speed calculation algorithm switches to the second method.

The proposed algorithm combines both of the above mentioned methods. The algorithm simultaneously measures the number of quadrature encoder pulses per constant period and their accurate time period. The speed can then be expressed as:

$$speed = \frac{k_1 \cdot N}{T} = \frac{k_1 \cdot N}{T_{clkT2} N_{clkT2}} = \frac{k \cdot N}{N_{clkT2}} \qquad \text{(EQ 7-8)}$$

where

$speed$	calculated speed	[-]
k	scaling constant	[-]
k_1	scaling constant	[s]
N	number of counted pulses per constant period	[-]
T	accurate period of N pulses	[s]
T_{clkT2}	period of input clock to timer A2	[s]
N_{clkT2}	number of pulses counted by timer A2	[-]

The speed-sensing algorithm uses three timers (A0, A2, A3) in quad timer module A and another timer as a time base (C0). The timer A0 is used in *quadrature count mode*, where the primary and secondary external inputs are decoded as quadrature encoded signals. The timer A0 counts to zero and then reinitializes. The other two timers (A2 and A3) are required for counting the quadrature signals and their period **(see Figure 7-11.)**. Timer A2 counts the quadrature encoder pulses from timer A0 and timer A3 counts a system clock divided by 2. The values in both timers can be captured by each edge of the Phase A signal. The time base is provided by timer C0, which is set to call a slow control loop every 1ms where the speed measurement is calculated. The speed processing algorithm works as follows:

1. The new captured values of both timers are read. The difference in the number of pulses and their accurate period are calculated from actual and previous values.
2. The new values are saved for the next period and the capture register is enabled. From this time, the first edge of the Phase A signal captures the values of both timers (A2, A3) and the capture register is disabled.
3. The speed is calculated using (EQ 7-1)
4. This process is repeated with each call of the speed processing algorithm; **(see Figure 7-12.)**

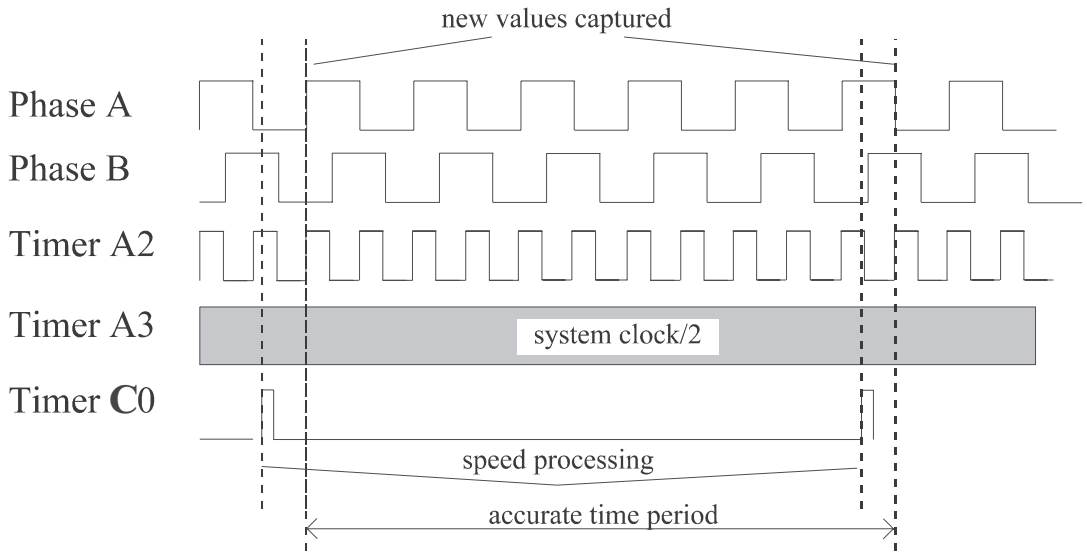

Figure 7-12. Speed Processing

7.5.1.1 Minimum and Maximum Speed Calculation

The minimum speed is given by the following equation:

$$v_{min} = \frac{60}{4N_E T_{calc}} \qquad \text{(EQ 7-1)}$$

where:

v_{min}	minimum obtainable speed	[rpm]
N_E	number of encoder pulses per revolution	[-]
T_{calc}	period of speed measurement (calculation period)	[s]

In this application, the quadrature encoder has 1024 pulses per revolution and the calculation period chosen is 1ms, based on the motor mechanical constant. Thus, equation (EQ 7-1) calculates the minimum speed as 14.6 rpm.

The maximum speed can be expressed as:

$$v_{max} = \frac{60}{4N_E T_{clkT2}} \quad \text{(EQ 7-2)}$$

where:

v_{max}	maximum obtainable speed	[rpm]
N_E	number of encoder pulses per revolution	[-]
T_{clkT2}	period of input clock to timer A2	[s]

After substitution in equation (EQ 7-2) for N and T_{clkT2} (timer A2 input clock = system clock 36 MHz/2), we calculate the maximum speed as 263,672 rpm. As shown, the algorithm presented can measure speed within a wide speed range. Because such a high speed is not practical, the maximum speed can be reduced to the required range by a constant k in (EQ 7-8). The constant k can be calculated as:

$$k = \frac{60}{4N_E T_{clkT2} v_{max}} \quad \text{(EQ 7-3)}$$

where:

k	scaling constant in the equation	[-]
v_{max}	maximum required speed	[rpm]
N_E	number of encoder pulses per revolution	[-]
T_{clkT2}	period of input clock to timer A2	[s]

In the application presented, the maximum measurable speed is limited to 4000 rpm.

Notes: To ensure correct speed calculation, the input clock of timer A2 must be chosen so that the calculation period of speed processing (in this case, 1ms) is represented in timer A2 as a value lower than 0x7FFF ($1000.10^{-6}/T_{clkT2}$<=0x7FFF).

7.6 Analog Sensing

7.6.1 Current Sensing

The DSP56F80x family provides the ability to synchronize the ADC and PWM modules via a SYNC signal. This exceptional hardware feature, which has been patented by Motorola, is used for current sensing. The PWM outputs a synchronization pulse, which is connected as an input to the synchronization module TC2 (quad timer C, channel 2). A high-true pulse occurs for each reload of the PWM regardless of the state of the LDOK bit. The intended purpose of TC2 is to provide a user-selectable delay between the PWM SYNC signal and the updating of the ADC values. A conversion process can be initiated by the SYNC input, which is an output of TC2. The time diagram of the automatic synchronization between PWM and ADC is shown in **Figure 7-13.**

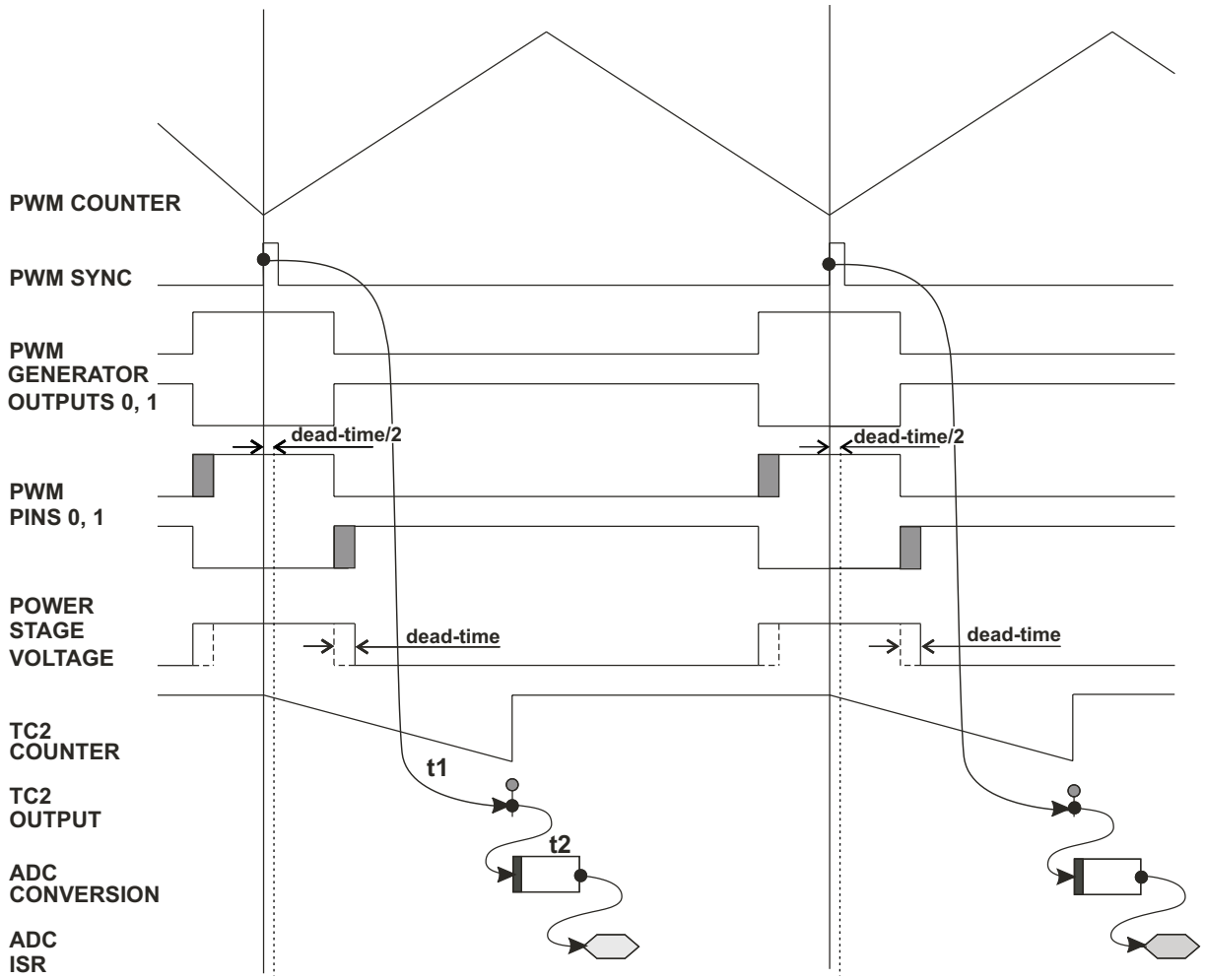

Figure 7-13. Time Diagram of PWM and ADC Synchronization

Phase currents are measured by shunt resistors at each phase. A voltage drop on the shunt resistor is amplified by an operational amplifier and shifted up by 1.65V. The resultant voltage is converted by the ADC; see **Figure 7-14.** and **Figure 7-15.**

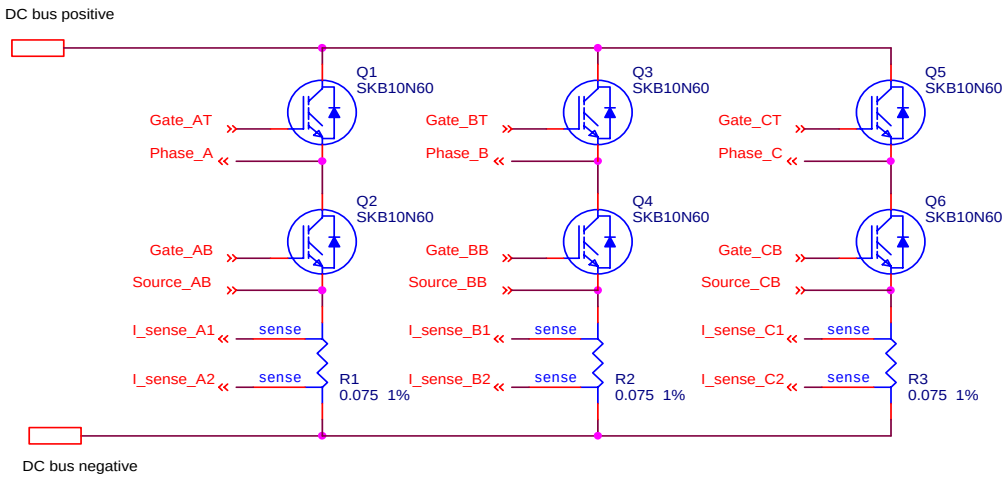

Figure 7-14. 3-Phase Bridge with Current Shunt Resistors

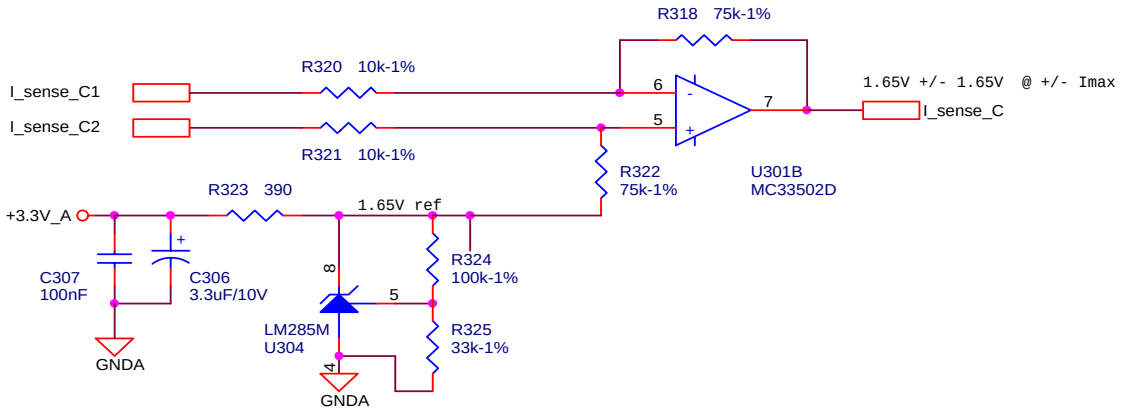

Figure 7-15. Current Amplifier for Phase C

The current cannot be measured at the current sense resistors at an arbitrary moment. This is because current only flows through the shunt resistor (for example, R1 corresponding to Phase A) if transistor Q2 is switched on. Only at that instant can the Phase A current be measured. Correspondingly, the current in Phase B can only be measured if transistor Q4 is switched on, and the current in phase C can only be measured if transistor Q6 is switched on. In order to get an actual instant of current sensing, voltage shape analysis must be performed.

Voltage shapes of two different PWM periods are shown in **Figure 7-16**. These voltage shapes correspond to center-aligned PWM sinewave modulation. As shown, the best instant of current sampling is in the middle of the PWM period, where all bottom transistors are switched on. However, not all three currents can be measured at an arbitrary voltage shape. *PWM period II* in **Figure 7-16** shows an instant when the bottom transistor of Phase A is on for a very short time. If the on time is shorter than a certain critical time, the current cannot be correctly measured. The specific critical time is given by the hardware configuration (transistor commutation times, response delays of the

processing electronics, etc.). In the 3-phase ACIM application, two PWM periods are always longer than the critical pulse width. Therefore, only two currents are measured and the third current is calculated from equation:

$$0 = i_A + i_B + i_C \quad \text{(EQ 7-4)}$$

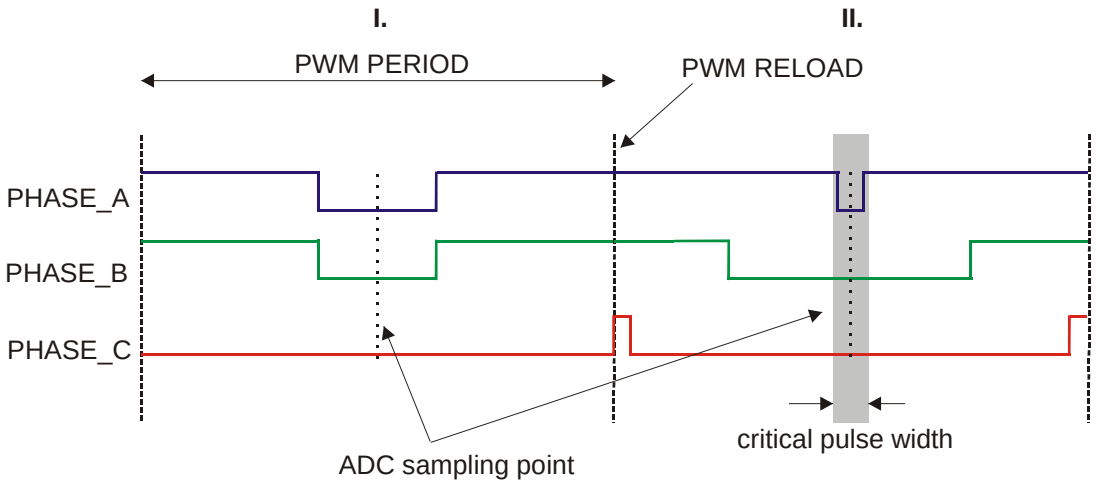

Figure 7-16. The Voltage Shapes of Two Different PWM Periods

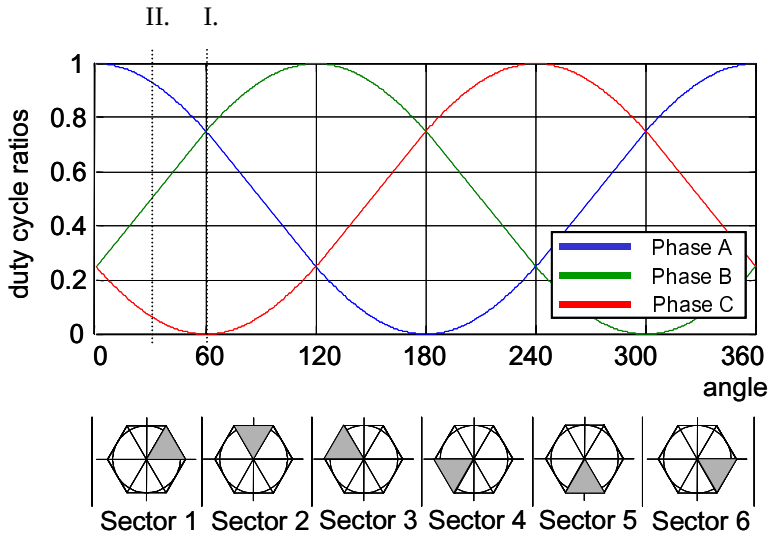

Figure 7-17. 3-Phase Sinewave Voltages and Corresponding Sector Values

The current that cannot be measured is the calculated. The simplest technique is to calculate the current of the most positive phase voltage, where the bottom PWM is switched on for the shortest time. For example, Phase A generates the most positive voltage within section 0 - 60°, Phase B within the section 60° - 120°, etc.; see **Figure 7-17**.

In the case presented, the output voltages are divided into six sectors; **(see Figure 7-17.)**. The current is then calculated according to the actual sector value:

for sectors 1, 6:

$$i_A = -i_B - i_C \quad \text{(EQ 7-5)}$$

for sectors 2, 3:

$$i_B = -i_A - i_C \quad \text{(EQ 7-6)}$$

for sectors 4, 5:

$$i_C = -i_B - i_A \quad \text{(EQ 7-7)}$$

Notes: The sector value is used only for current calculation and has no other meaning at the sinewave modulation. But if any type of the space vector modulation is used, the sector value can be obtained as a part of space vector calculation and used for phase current measurement.

7.6.2 Voltage Sensing

The resistor divider network in **Figure 7-18** is used to sense the DCBus voltage. The voltage signal is divided down to the 3.3V level and is ready for further processing. DCBus voltage does not change rapidly. It is almost a constant, with ripple caused by the structure of the power supply. If a bridge rectifier for conversion of the AC line voltage is used, the ripple frequency is twice the AC line frequency. Ripple amplitude should not exceed 10% of the nominal DCBus value if the power stage is designed correctly.

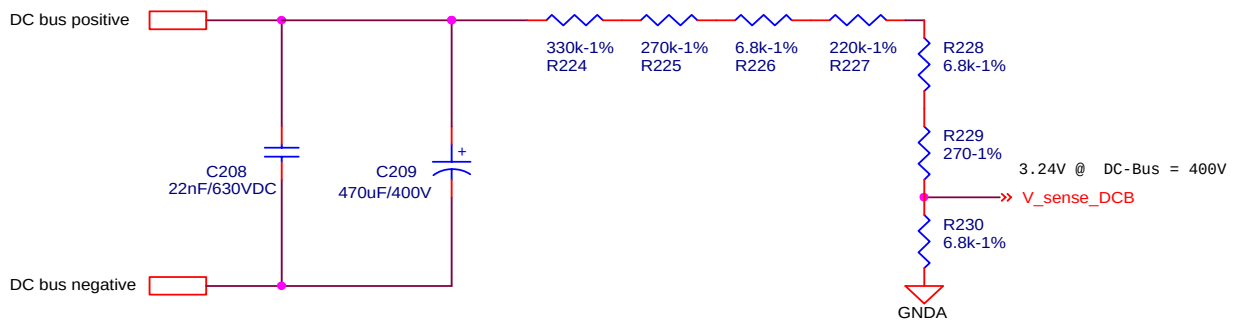

Figure 7-18. DC-Bus Voltage Sensing

The measured DCBus voltage must be filtered in order to eliminate noise. One of the easiest and fastest techniques is a first order filter, which calculates the average filtered value recursively from the last two samples and a coefficient C:

$$u_{DCBusFilt}(n+1) = (Cu_{DCBusFilt}(n+1) - Cu_{DCBusFilt}(n)) - u_{DCBusFilt}(n) \quad \text{(EQ 7-8)}$$

To speed up initialization of voltage sensing, the filter has an exponential dependence with a constant of 1/N samples and a moving average filter that calculates the average value from the last N samples is used:

$$u_{DCBusFilt} = \sum_{n=1}^{-N} u_{DCBus}(n) \quad \text{(EQ 7-9)}$$

7.6.3 Power Module Temperature Sensing

The measured power module temperature is used for thermal protection The hardware realization is shown in **Figure 7-19.** The circuit consists of four diodes connected in series, a bias resistor, and a noise suppression capacitor. The four diodes have a combined temperature coefficient of 8.8 mV/ºC. The resulting signal, *Temp_sense*, is fed back to an A/D input where software can be used to set safe operating limits. In the application presented, the temperature (in Celsius) is calculated according to the conversion equation:

$$\text{temp} = \frac{\text{Temp\_sense} - b}{a} \quad \text{(EQ 7-10)}$$

where:

temp	power module temperature in Celsius
Temp_sense	voltage drop on the diodes, which is measured by the ADC
a	diode-dependent conversion constant (a = -0.0073738)
b	diode-dependent conversion constant (b = 2.4596)

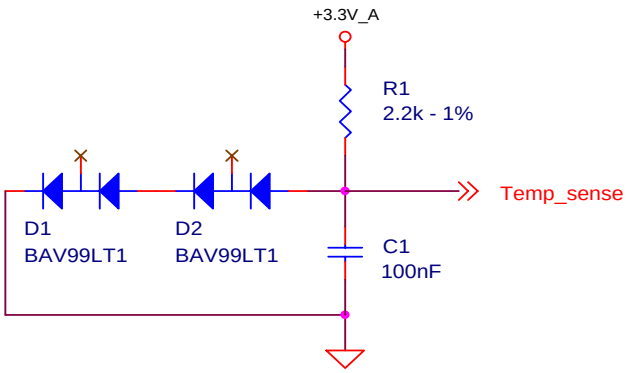

Figure 7-19. Temperature Sensing

7.7 START/STOP Switch and Button Control

The START/STOP switch is connected to a GPIO pin in the case of DSP56F805/7EVMs. The state of the START/STOP switch can be read directly from the GPIO Data Register. In the DSP56F803EVM, there are no free GPIO pins; therefore, the switch is connected to ADC input AN7 and switch status is then obtained with the ADC. The switch logical status is obtained by comparing a measured value with the threshold value.

Since the DSP56F803EVM does not have free GPIO pins for user buttons, they are connected to IRQA and IRQB pins. The DSP56F805/7EVMs use the same connection of the push buttons to the IRQA and IRQB pins, enabling the same code to run on DSP56F803/5/7EVM boards. The EVM boards do not solve the button contact bouncing, which may occur during the pushing and releasing of a button. Therefore, this issue must be solved by software.

The IRQA and IRQB are maskable interrupts connected directly to the DSP's core. The IRQA and IRQB lines are internally synchronized with the processor's internal clock and can be programmed as level-sensitive or edge-sensitive. The IRQA and IRQB interrupts do not have interrupt flags; therefore, the flags are replaced by software flags. The following algorithm is used to check the state of the IRQ line and is described for one interrupt.

The level-sensitive trigger mode of the IRQ interrupt is set. When the button is pressed, the logical level 0 is applied on the IRQ line and an interrupt occurs; see **Figure 7-20**. The ISR disables the IRQ interrupt to avoid multiple calls of the ISR due to contact bouncing, sets the debounce counter to a predefined value, and sets the variable *buttonStatus* to 1. The variable *buttonStatus* represents the interrupt flag. Using the DSP56F80x's timer or a software timer, the *ButtonProcessing* function is called periodically; see **Figure 7-20**. The function *ButtonProcessing* decrements the debounce counter and if the counter is zeroed, the IRQ interrupt is re-enabled. Button pressing is checked by the *ButtonEdge* function; see **Figure 7-21**. When the variable *buttonStatus* is set, the *ButtonEdge* function returns "1" and clears *buttonStatus*. When the variable *buttonStatus* is not set, the *ButtonEdge* function returns "0".

The value of the debounce counter should be set according to a *ButtonProcessing* calling period close to 180ms. This value is sufficient to prevent multiple IRQ ISR calls due to contact bouncing.

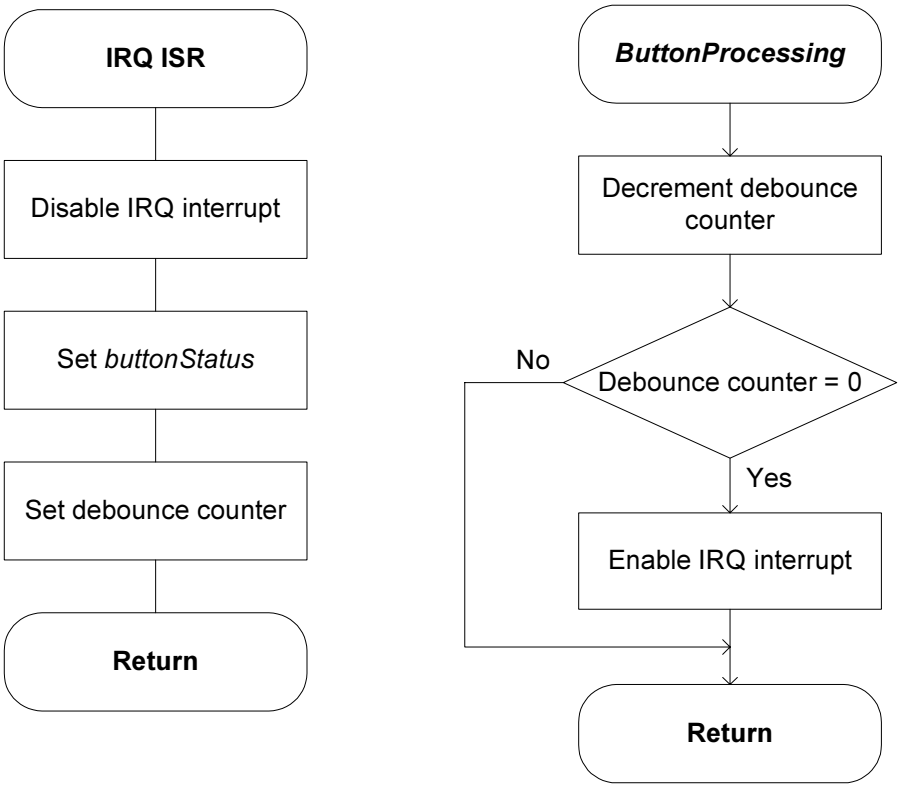

Figure 7-20. Button Control - IRQ ISR and *ButtonProcessing*

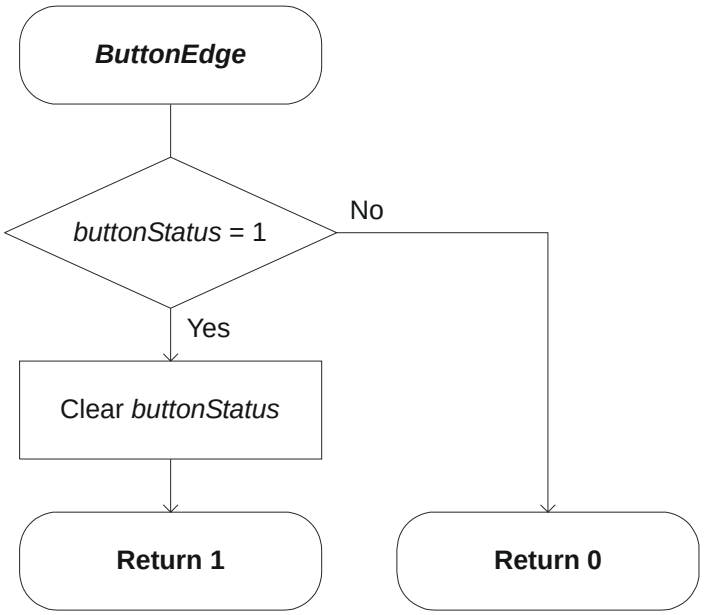

Figure 7-21. Button Control - *ButtonEdge*

8. SDK Implementation

The Motorola Embedded SDK is a collection of APIs, libraries, services, rules and guidelines. This software infrastructure is designed to let a DSP5680x software developer create high-level, efficient and portable code. The following section describes how the 3-phase AC induction motor vector control application was written under the SDK.

8.1 Drivers and Library Functions

The 3-phase AC induction motor vector control application uses the following drivers:

- ADC driver
- Quad timer driver
- Quadrature decoder driver
- PWM driver
- LED driver
- SCI driver
- PC master software driver
- Switch driver (only for DSP56F805EVM and DSP56F807EVM)
- Brake driver

All drivers are included in the *bsp.lib* library.

The 3-phase AC induction motor vector control application uses the following library functions:

- *boardId* (board identification, *bsp.lib* library)
- *fluxmodel* (rotor flux calculation, *mcfunc.lib* library)

- *cptrfmClarke* (forward Clarke transformation, *mcfunc.lib* library)
- *dqestabl* (d-q system establishment, *mcfunc.lib* library)
- *decoupling* (stator voltage decoupling, *mcfunc.lib* library)
- *cptrfmParkInv* (inverse Park transformation, *mcfunc.lib* library)
- *svmElimDcBusRip* (DCBus ripple elimination, *mcfunc.lib* library)
- *svmPwmIct* (space vector modulation, *mcfunc.lib* library)

8.2 *Appconfig.h* File

The purpose of the *appconfig.h* file is to provide a mechanism for overwriting default configuration settings, which are defined in the *config.h* file.

There are two *appconfig.h* files The first is dedicated to external RAM (..*ConfigExtRam* directory) and the second is dedicated to flash memory (..*ConfigFlash* directory). In the case of the 3-phase AC induction motor vector control application, the files are identical, with the following exceptions:

- The *appconfig.h* for the ExtRAM target contains a definition of the memory support; Flash support is not defined
- The *appconfig.h* for Flash target contains a definition of the flash support; memory support is not defined
- The *appconfig.h* for the ExtRAM target contains a PC master software recorder buffer 2000 samples long, while the *appconfig.h* for Flash target contains a PC master software recorder buffer only 200 samples long. This is due to the limited internal RAM memory size.
- The *appconfig.h* for the DSP56F805EVM and DSP56F807EVM contains the definition of a switch driver, while the *appconfig.h* for the DSP56F803EVM does not

The *appconfig.h* file can be divided into two sections. The first defines which components of the SDK libraries are included in the application; the second overwrites the standard settings of components during their initialization.

8.3 Drivers Initialization

Each on-chip DSP or EVM board periphery is accessible through a driver. The driver initialization of all peripherals used is described in this section. For detailed descriptions of drivers, see the **Targeting Motorola DSP5680x Platform** documentation.

To use a driver, follow these steps:

- Include the driver support in the *appconfig.h*
- Fill in the configuration structure in the application code for specific drivers, depending on the driver type
- Initialize the configuration setting in the *appconfig.h* for specific drivers, depending on the driver type
- Call the *open* (create) function

The *ioctl* function call provides access to individual driver functions.

8.4 Interrupts

The SDK services the interrupt routines calls and automatically clears the interrupt flags. The user defines the callback functions called during the interrupts. The callback functions are assigned during driver initialization. Callback function assignment is defined as an item in the initialization structure. Some drivers define the callback function in the *appconfig.h* file.

8.5 PC Master Software

PC master software was designed to provide a debugging, diagnostic and demonstration tool for development of algorithms and applications. It consists of a component running on a PC and a part running on the target DSP, connected by an RS-232 serial port. A small program is resident in the DSP that communicates with PC master software to parse commands, return status information to the PC, and processes control information from the PC. PC master software executing on the PC uses Microsoft Internet Explorer as the user interface.

PC master software is a part of the Motorola Embedded SDK and may be selectively installed during the SDK installation.

To enable PC master software operation on the DSP target board application, the following lines must be added to the *appconfig.h* file:

```
#define INCLUDE_SCI          /* SCI support */
#define INCLUDE_PCMASTER     /* PC master software support */
```

It automatically includes the SCI driver and installs all the necessary services. The baud rate of the SCI communication is 9600Bd. It is set automatically by the PC master software driver.

Part of the PC master software is also a recorder, which is able to sample the application variables at a specified sample rate. The samples are stored to a buffer and read by the PC via an RS232 serial port. The sampled data can be displayed in a graph or the data can be stored. The recorder behaves like a simple on-chip oscilloscope with trigger/pretrigger capabilities. The size of the recorder buffer and the PC master recorder time base must be defined in the *appconfig.h* file:

```
#define  PC_MASTER_REC_BUFF_LEN         200    /* recorder buffer
                                                 length (words) */
#define  PC_MASTER_RECORDER_TIME_BASE   0x4001 /* 1ms sampling
                                                 period */
```

The recorder routine must be called periodically in the loop in which you want to take the samples. The following line must be added to the loop code:

```
pcmasterdrvRecorder(); /* PC master recorder routine call */
```

A detailed description of PC master software is provided in the **PC Master Software User Manual**.

The actions controlled by PC master software are:

- Take over the PC remote control
- Start/Stop control
- Motor speed set point

Variables read by PC master software by default and displayed to the user are:

- Required speed
- Actual motor speed
- PC remote control mode
- Start/Stop status

- Drive Fault status
- DCBus voltage level
- Identified power stage boards
- System status

The profiles of required and actual speed can be seen in the speed scope window.

9. DSP Usage

Table 9-1 shows how much memory is used to run the 3-phase AC induction motor vector control application. The PC master software recorder buffer is set to zero and a majority of the DSP memory is still available for other tasks.

Table 9-1. RAM and FLASH Memory Usage for SDK2.4 and CW4.1

Memory	Available DSP56F803 DSP56F805	Available DSP56F807	Used Application + Stack (DSP56F805)	Used Application without PC master software, SCI (DSP56F805)
Program FLASH	32K x 16bit	60K x 16bit	13989 x 16bit	9247 x 16bit
Data FLASH	4K x 16bit	8K x 16bit	399 x 16bit	399 x 16bit
Program RAM	512 x 16bit	2K x 16bit	125 x 16bit	125 x 16bit
Data RAM	2K x 16bit	4K x 16bit	1139 + 352 stack x 16bit	876 + 352 stack x 16bit

10. References

[1] Bose, K. B. (1997). *Power Electronics and Variable Frequency Drives,* IEEE Press, ISBN 0-7803-1061-6, New York.

[2] Caha, Z.; Cerny, M. (1990). *Elektricke pohony,* SNTL, ISBN 80-03-00417-7, Praha.

[3] Subrt, J. (1987). *Elektricke regulacni pohony II,* VUT Brno, Brno.

[4] Vas, P. (1998). *Sensorless Vector and Direct Torque Control*, Oxford University Press, ISBN 0-19-856465-1, New York.

[5] Motorola**,** Inc. (2000). *DSP56800 Family Manual*, DSP56F800FM/D, Rev. 1.

[6] Motorola**,** Inc.(2001). *DSP56F80x User's Manual***,** DSP56F801-7UM/D, Rev. 3.0.

[7] Motorola**,** Inc. (2001). *DSP Evaluation Module Hardware User's Manual,* DSP56F805EVMUM/D, Rev. 3.0.

[8] Motorola**,** Inc. (2001). *DSP Evaluation Module Hardware User's Manual,* DSP56F803EVMUM/D, Rev. 3.0.

[9] Motorola**,** Inc. (2001). *DSP Evaluation Module Hardware User's Manual,* DSP56F807EVMUM/D, Rev. 0.

[10] Motorola, Inc. (2000). *3-Phase AC/BLDC High-Voltage Power Stage User's Manual,* MEMC3PBLDCPSUM/D.

[11] Motorola Software Development Kit documentation available on the web page: **www.motorola.com**

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/

MOTOROLA

AN1930/D