Application Note

Vector Control of a 3-Phase AC Induction Motor Using the Z8FMC16100 MCU

AN024703-1111

Abstract

The 3-phase alternating current (AC) induction motors are mechanically simple, rugged, highly reliable, lower in cost per horsepower than DC motors and capable of more torque and efficiency than single-phase AC motors. A 3-phase AC induction motor can be controlled by varying its inputs according to a mathematical model of the rotor flux field in a complex vector space (vector control). Vector control provides efficient and accurate control of the motor's speed and torque.

Until now, vector control has been the domain of digital signal processors (DSPs), digital signal controllers (DSCs) and a few 32-bit and 16-bit microcontrollers. Constant cost pressure and increased consumer expectations have driven design engineers to seek minimal hardware solutions that extract maximum performance from motors used in consumer goods.

This application note demonstrates how Zilog's Z8FMC16100 MCU can implement efficient, cost-conscious vector control of an AC induction motor.

Note: The source code file associated with this application note, AN0247-SC01.zip, is available for download on zilog.com. This source code has been tested with version 5.0.0 of ZDSII for Z8 Encore! XP-powered MCUs. Subsequent releases of ZDSII may require you to modify the code supplied with this application note.

Z8FMC16100 Series Flash Microcontrollers

The Z8FMC16100 Series of Flash MCUs is based on Zilog's advanced 8-bit eZ8 CPU core and is optimized for motor control applications. It supports control of single- and multiphase variable-speed motors. Target applications are consumer appliances, HVAC, factory automation, refrigeration and automotive applications, among others.

To rotate a 3-phase motor, three AC voltage signals must be supplied and phase-shifted 120 degrees from each other. To control a 3-phase motor, the MCU must provide six Pulse Width Modulation (PWM) outputs. The Z8FMC16100 Series Flash MCU features a flexible PWM module with three complementary pairs, or six independent PWM outputs, supporting deadband operation and fault protection trip input. These features provide multiphase control capability for various motor types and ensure safe operation of the motor by providing immediate shutdown of the PWM pins during a fault condition.

Discussion

An electric motor consists of a *stator*, a stationary frame in which a rotating component, or *rotor*, is mounted on a shaft and bearings. In a 3-phase AC induction motor, the stator is laced with three sets of inductor windings energized by three AC voltage inputs that are phase-offset 120 degrees from each other, producing a rotating field of magnetic flux inside the motor. This field creates an induced magnetic flux in short passive windings on the rotor. The stator field exerts a magnetic force on the rotor flux field, resulting in torque on the output shaft.

In a 3-phase motor control application, the input to the motor is produced by a 3-phase inverter bridge. A bridge contains three complementary source/drain transistor pairs which connect either ground or high-voltage DC to each of its three outputs in response to digital control signals from the microcontroller. The microcontroller uses PWM on the bridge control signals to generate three approximately-sinusoidal AC waveforms on the bridge outputs, with the required 120-degree phase offset.

The duty cycle of each microcontroller PWM output is varied to control the period and amplitude of the generated AC signal, which in turn determines the speed and torque of the motor. The microcontroller senses the resulting AC current and motor speed and periodically adjusts the AC signal waveform, increasing or decreasing torque to maintain the appropriate motor speed. For more information about motor control, see documents listed in Appendix A. References on page 27.

Theory of Operation

In this application, the Z8FMC16100 microcontroller's PWM module is configured as three complementary output pairs. Each output pair controls one complementary source/drain transistor pair in the inverter bridge. The PWM module is configured to insert a 0.6µs deadband between ON states to prevent leakage that could occur if one transistor in a pair turns on before the other is fully off.

Each PWM output pair produces a stream of complementary on/off pulses to activate the corresponding source or drain transistor in the inverter bridge. The voltage of each bridge output varies with the source/drain pulse duty cycle.

The period of each PWM cycle is configured to be 50µs and the PWM module generates an interrupt request at the end of each cycle. The PWM interrupt service routine (ISR) leaves the PWM interrupt disabled while it executes and is tuned to execute within 200µs to 250µs. Therefore, the PWM ISR is executed every five PWM cycles, or 250µs.

The PWM ISR controls all of the ongoing application program functions after initialization. The primary goal of the ISR is to update the duty cycle value for each PWM channel as required to produce the appropriate AC waveforms at the inverter bridge outputs.

The PWM duty cycle values are derived from a feedback loop based on rotor speed and rotor flux position. The stator current waveform is determined by sampling two phases of the inverter output current and reconstructing the third phase.

The rotor speed is sampled by a counter/timer configured to measure the period of a magnetic position sensor on the motor. The measured speed is periodically compared to the requested speed received through UART from an external controller. The resulting speed command value is used to create a rotation reference frame expressed as a two-phase direct/quadrature (DQ) vector.The resulting vector is regulated to produce the necessary flux and torque amplitude in the output vector.

An inverse Park transform is used to rotate the reference frame relative to the rotor position. Then an inverse Clarke transform is used to convert the rotated two-phase reference frame back into a 3-phase expression. At the same time, the vector's position is classified into one of six states, each corresponding to one sixth of the vector circle.

When the ISR calculates duty cycles for the three PWM channels, one phase is left unmodulated (ON or OFF for the whole PWM cycle) and the flux vector is encoded by modulating the other two phases. This reduces switching losses by one third. The two phases to be modulated depend on the flux vector's state position. After the PWM space vector calculation is complete, the corresponding PWM registers are updated. For more details about the software implementation, see Software Implementation on page 11.

Description of Components

This design uses the following major components:

- Z8FMC16100 microcontroller
- Insulated-gate bipolar transistors (IGBTs)
- I^2C interfaced 8-bit digital-to-analog converter (DAC)
- Washing Machine Motor

Z8FMC16100 Microcontroller

This application employs the Z8FMC16100 MCU's external precision oscillator, 10-bit analog-to-digital converter (ADC), I/O and UART features to demonstrate the washing machine's Vector Control Command Interface (GUI). User commands are transmitted to the Z8FMC16100 MCU as serial data to, in theory, allow very complex control commands to be transmitted. This user interface consists of a GUI for speed control, a PI loop, and start and stop motor commands. The speed of the motor is plotted in user-defined time intervals.

Additionally, two 6N137 single-channel optocouplers transmit UART command signals while electrically isolating the GUI from the high-voltage modules. For a discussion about this GUI, please see the Application Hardware section on page 5.

Insulated-Gate Bipolar Transistors

Six IXYS IGBTs (part number IXGA30N60C3C1) are used under PWM control to generate the required 3-phase AC motor drive current from a high-voltage DC supply. Although these IGBTs are rated for momentary short-circuit operation, the Z8FMC16100 MCU's PWM deadband feature is used to reduce leakage when the IGBTs are switched.

Three IXYS high-side and low-side gate drivers (part number IX2113) are used to convert the Z8FMC16100 MCU's CMOS PWM outputs into signals with the voltage required to drive the IGBT gate inputs.

Digital-to-Analog Converter

The Z8FMC16100 MCU's I^2C interface is used with a DAC5574 digital-to-analog-converter to generate up to four analog test outputs based on internal program values. As a result, the internal values are allowed to be directly compared to the motor controller's output waveform on an oscilloscope.

Motor

The motor is a 3-phase AC induction motor used in washing machine applications, as illustrated in Figure 1.

Figure 1. Washing Machine Application

The application motor is illustrated in Figure 2. The motor has the following specifications:

- Rated power: 500 W
- RPM max: 15,000 rpm
- Max current: 2.8 Arms
- Continuous current: 1.5 Arms
- DC Bus Voltage: 350 VDC

The motor windings are connected in the Wye configuration for this application.

The back side of the motor has an integrated tachogenerator, which consists of an 16-pole magnet mounted on the shaft and a single-phase winding mounted on the housing. This produces a sine wave signal whose voltage and frequency are proportional to the motor speed. The interface to the Z8FMC16100 MCU is a simple transistor circuit that squares off the sine wave into 3.3 V digital pulses going to Timer 0 in Capture Restart mode. This yields 8 pulses per revolution.

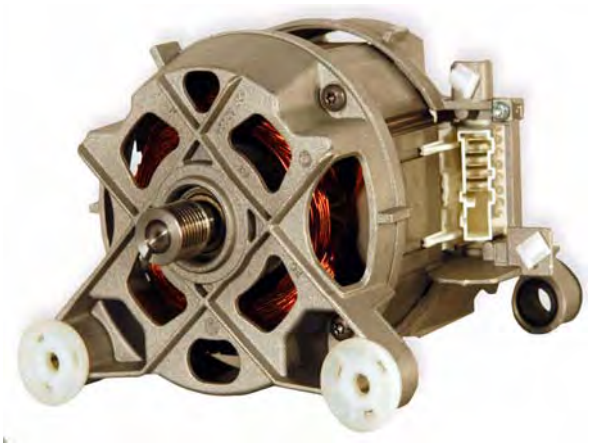

Figure 2. Washing Machine Motor with Tachogenerator

Application Hardware

The application electronic is controlled with a Vector Control Command Interface (GUI) to start and stop the motor, control its speed, and observe the PI loop. The speed of the motor is plotted in user-definable time intervals. Figure 3 presents all application hardware and connections.

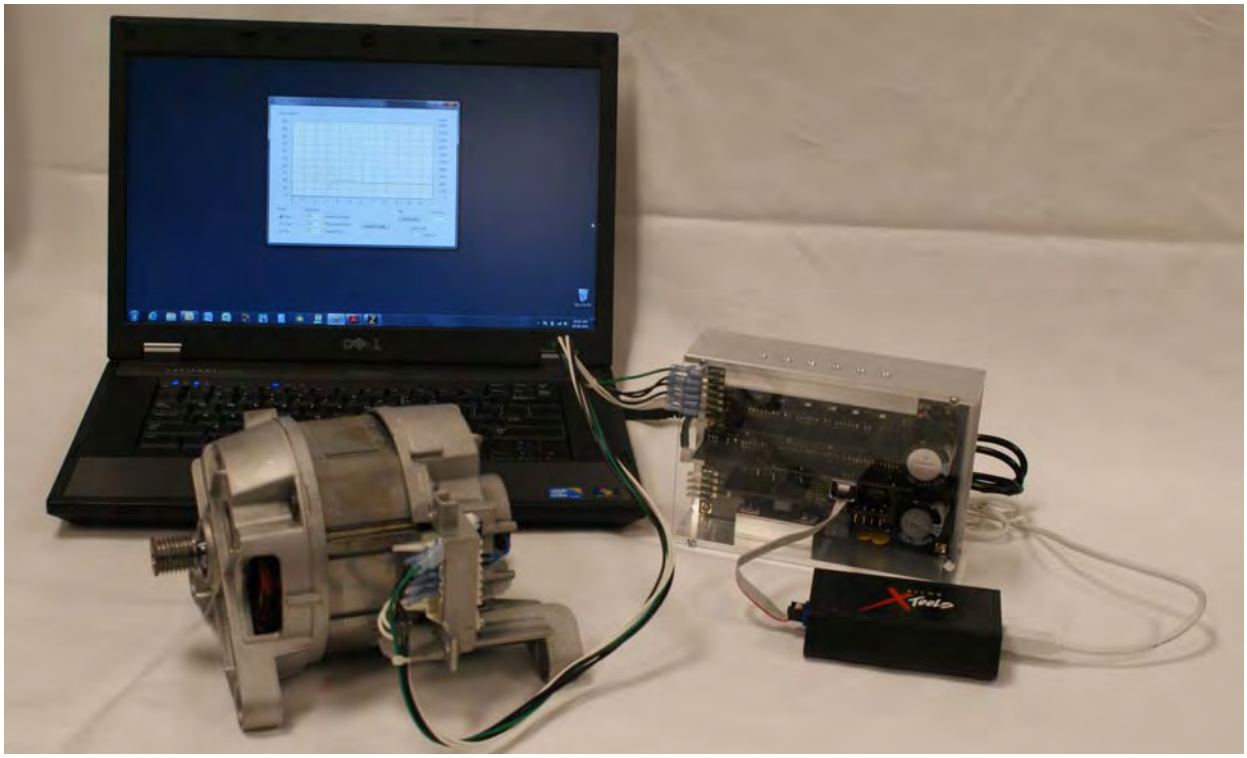

Figure 3. Modular Controller and Vector Control GUI

Figure 4 offers a closer view of the Vector Control GUI seen in Figure 3.

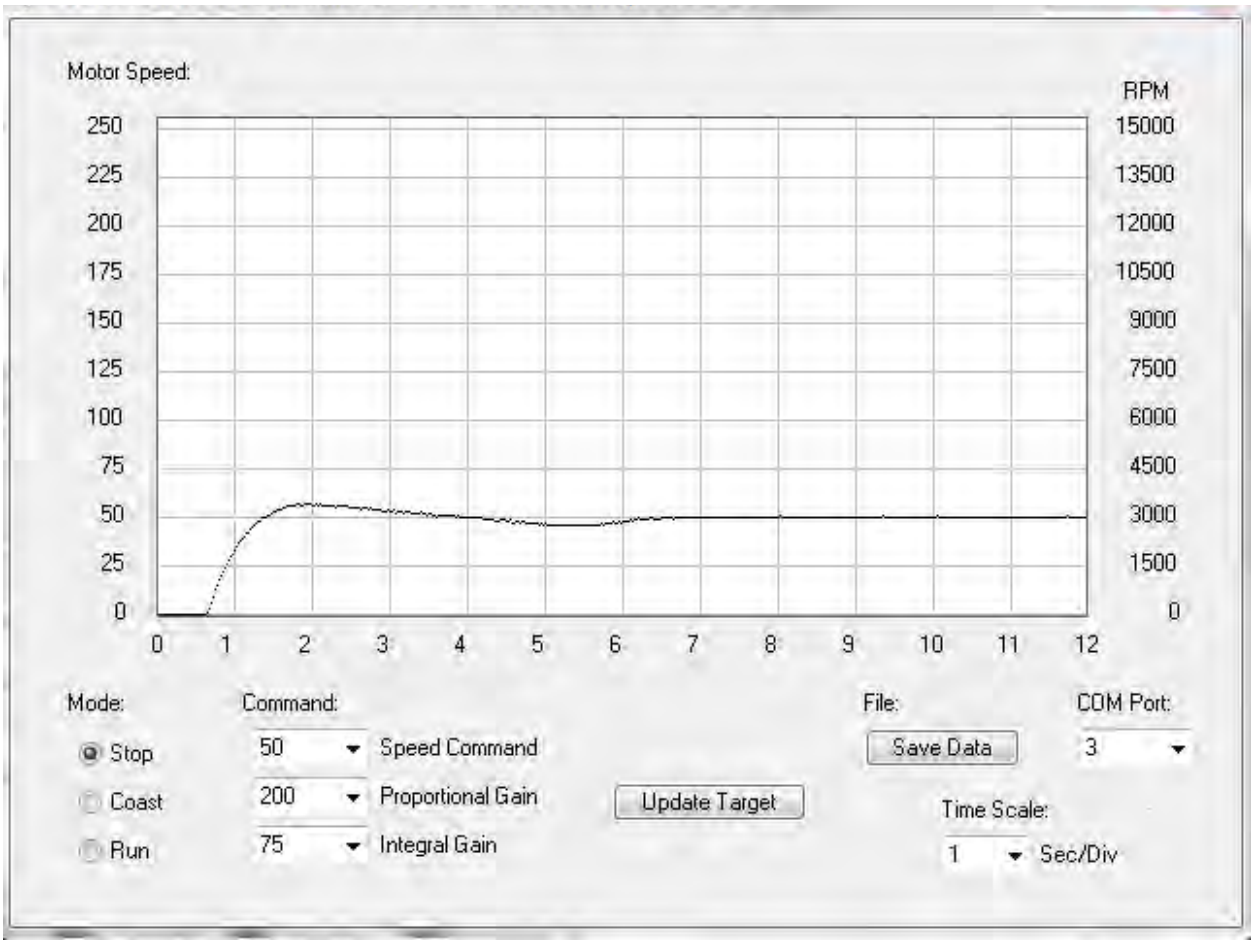

Figure 4. Vector Control Command Interface

Figure 5 shows a block diagram of the application hardware design, showing the logical position of each module in the controller.

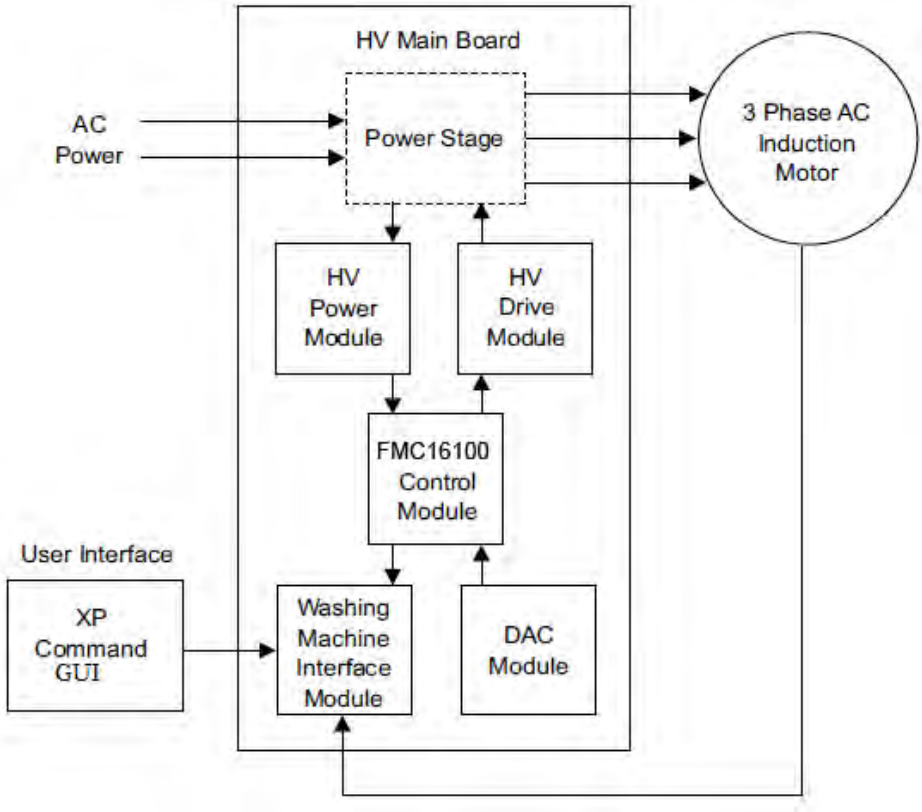

Figure 5. Hardware Block Diagram

The following sections briefly describe each of the modules that comprise the application hardware. The schematics are shown in Appendix C. Schematic Diagrams on page 29.

High-Voltage Main Board and Power Stage

The main board serves as the connector backplane for all the other controller modules. It also provides the high-voltage power stage that converts the single-phase AC supply to the 3-phase motor control output under PWM control. The schematic for the main board module is illustrated in Figure 17 on page 29.

The power stage section rectifies and filters the single-phase AC supply to provide a high-voltage DC bus supply. This section also contains the six IGBTs used under PWM control to generate the 3-phase AC motor drive current.

The MCU is powered after the AC supply is connected, but a normally-open relay withholds DC power from the IGBTs until the MCU has started and the application software has completed its necessary startup functions. The application then uses an I/O port output to enable a simple transistor driver that closes the relay, powering the IGBTs.

Z8FMC16100 Control Module

The schematic for the MCU module is shown in Figure 18 on page 30. This module contains the Z8FMC16100 MCU, a 20MHz ceramic resonator, a 6-pin DBG interface and the off-chip components of the high-voltage current sense and overcurrent detection circuits.

High-Voltage Gate Drive Module

The schematic for the gate drive module is shown in Figure 19 on page 31.This module contains three IX2113 high-side and low-side gate drivers that convert the Z8FMC16100 MCU's PWM outputs into the voltage required to drive the IGBT gate inputs.

High-Voltage Power Supply Module

The schematic for the power supply module is shown in Figure 20 on page 32. This module uses an LNK304G IC to convert rectified AC from the power stage to generate the application's 120mA, 12V supply. This 12V supply is then regulated by a UA78M33 IC to provide the 3.3V supply.

Figure 6 illustrates the overall current sense circuit, including some power stage components and components internal to the MCU. V_{REF} is the internal reference voltage of the MCU analog-to-digital converter and the full scale input range. At zero current (zero voltage across *Rsense*), the output of the Op Amp is at 1/2 V_{REF}, which centers the current waveform at one-half of the full input range of the ADC.

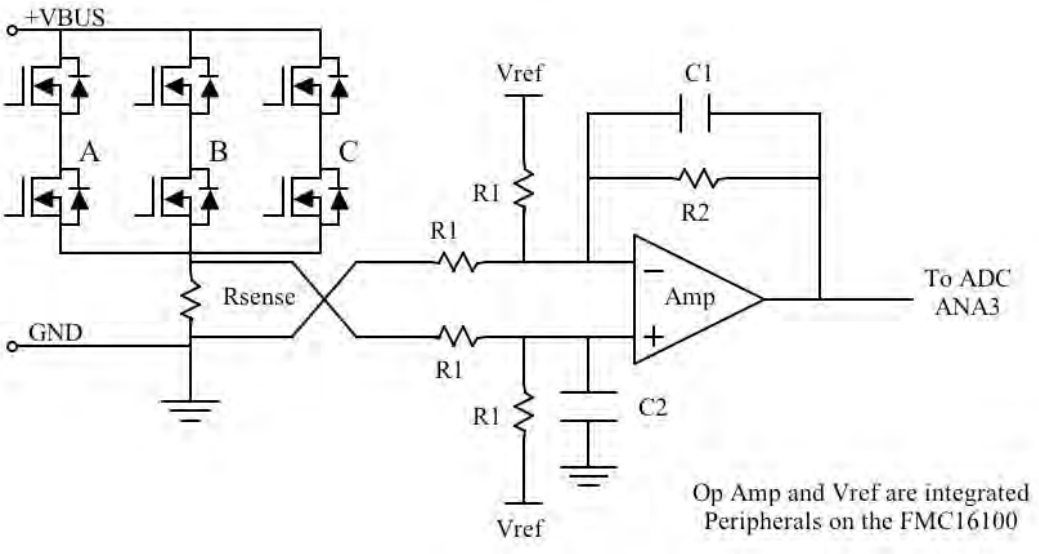

Figure 6. Current Sense Circuit

R1 = 10KΩ

R2 = 50KΩ

Gain = 5

V_{REF} = 2V

C1 = unused

C2 = 100pF

$$Rsense = \frac{V_{REF}}{2 \times Gain \times I\_max} = \frac{2V}{2 \times 5 \times 10A} = 0.02\Omega$$

The resistive divider also forms a Low Pass Filter (LPF) to take out the diode reverse recovery current spike in the current sense signal. Because this spike usually occurs in the MHz frequency range, a single-pole LPF is sufficient set at approximately 300kHz. Because the Operational Amplifier circuit already has an LPF, this issue is more important for the following comparator circuit.

The cutoff frequency is calculated using the following equation:

$$fc = \frac{1}{2\pi \times R1 \,||\, R1 \times C2} = \frac{1}{2\pi \times 5k\Omega \times 100pF} = 318kHz$$

Figure 7 illustrates the overall schematic for the overcurrent sense circuit. It includes power stage components and components internal to the MCU.

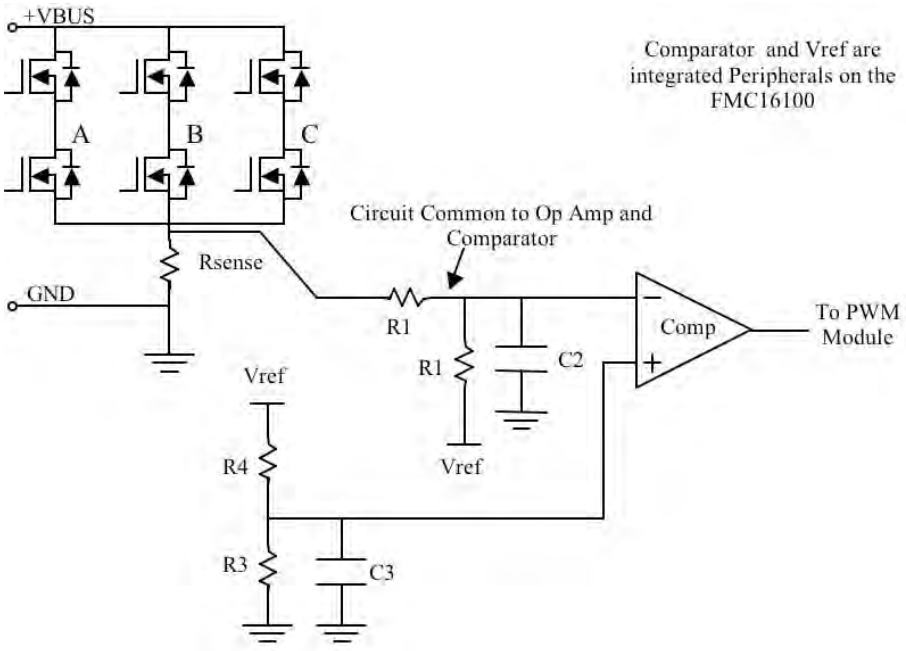

Figure 7. Overcurrent Sense Circuit

R3 and R4 set the overcurrent trip threshold. Because it is a DC value, a bypass capacitor (C3) of 0.1µF is used.

For R4 = 10, R3 is selected as:

$$R3 = R4 \times \frac{(V_{REF} + I\_max \times Rsense)}{(V_{REF} - I\_max \times Rsense)} = 10\ k\Omega \times \frac{(2V + 10A \times 0.02\Omega)}{(2V - 10A \times 0.02\Omega)} = 12.2\ k\Omega \approx 12.4\ k\Omega$$

Washing Machine Interface Module

This module provides the speed control interface between the modular controller and the washing machine. It consists of the following parts:

- Optoisolated UART command interface
- Motor tachometer/turn sensor

The optoisolator section ensures the electrical isolation of the user command interface from the high-voltage module and also serves to multiplex the Z8FMC16100 MCU's RXD and TXD signals onto a single isolated DATA signal that connects to the remote Vector Control Command Interface (GUI). Power and ground for the user side of the optoisolator section are provided by the GUI.

The tachometer/turn sensor section converts the analog rotor-sense signal from the AC motor to a logic-level DC pulse stream suitable for the Z8FMC16100 MCU's timer/counter input.

DAC Test Output Module

This module contains a DAC5574IDGS DAC whose I^2C serial interface is driven by the Z8FMC16100 MCU. The controller software transmits selected internal values to the DAC, which converts those digital values to one of four analog outputs. This allows, for example, the internal reconstructed waveform to be compared on an oscilloscope to the actual AC output from the power stage.

Vector Control Command Interface

The external Vector Control GUI provides user control to modify the speed and PI loop settings. The speed is plotted in user-defined time intervals, which can be seen in the GUI in Figure 4 on page 6.

Software Implementation

Figure 8 shows a functional block diagram of the vector control application.

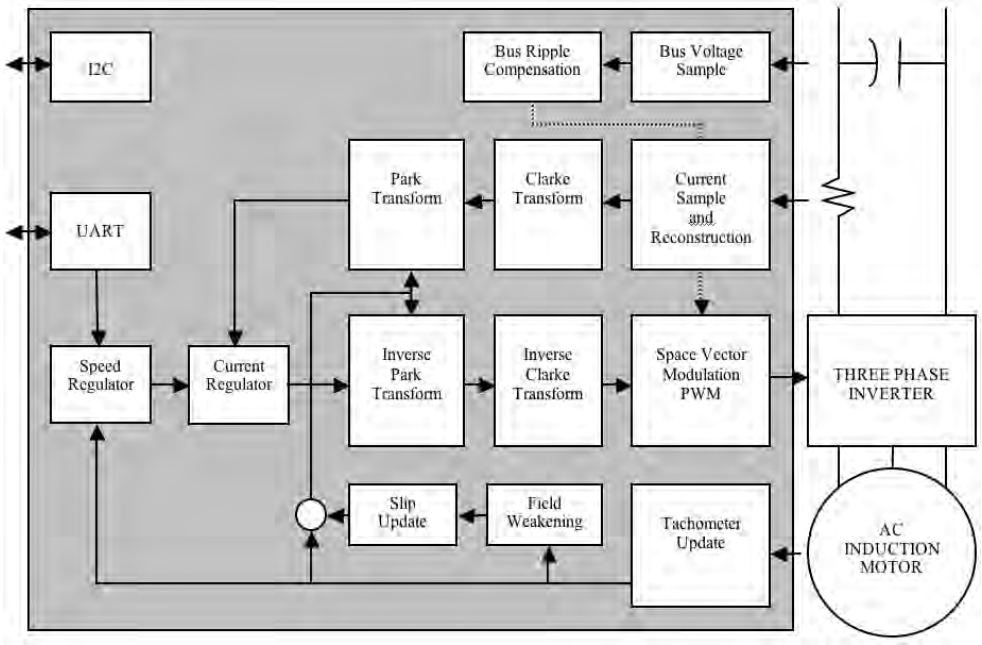

Figure 8. Functional Block Diagram

Variables and Declarations

Declarations set the PWM frequency, the Timer 0 scaling for the tachogenerator and the modulation values. All variables are declared as global for speed of execution and simplicity of code.

All variables are unsigned. All signed values have separate variables of the form *Name* for positive and *nName* for negative contents. One of these variables is always zero. Using unsigned math speeds up execution but makes the algorithms longer in code.

The ADC inputs are treated as 8 bits and the PWM output is 8 bits. Internal computations generally have 4 extra bits of precision especially in the control loops that use PI (Proportional Integral) compensators, this prevents rounding from affecting integrator precision. For Inverse Park Transform (vector rotation) the concept of a quadrant is used to determine calculations on the unsigned values.

The program uses a 128-value sine look-up table, `Sine_Table[128]`. This table represents sine values from 0 to 90 degrees, or approximately 0.7-degree steps. Stepping through the table in the forward direction yields sine values from 0 to 90 degrees and stepping through the table in the reverse direction yields cosine values from 0 to 90 degrees. This along with knowledge of the quadrant of the angle of interest (Theta) allows full reconstruction of sine and cosine values for any angle between 0 and 360 degrees.

The polarity table, `Polarity_table[8]`, contains bit masks used to set the polarity bits in the PWMCTL1 register. A 1 bit causes the corresponding channel to start the next PWM cycle in the ON state. A 0 bit causes the channel to start in the OFF state.

The ripple compensation table, `ripple_table[256]`, contains precalculated bus voltage ripple compensation factors; see the Bus Ripple Compensation section on page 19.

Initialization

The following initializations are performed when `main()` is executed:

Option Bits

The options bits set the PWM polarity and the WDT default mode.

Oscillator

Oscillator initialization consists of sending an unlock sequence to the oscillator control register, then enabling the external oscillator circuit. After allowing a settling time of 50 ms, the oscillator control register is unlocked again.

Digital Output

PB7 is used as a digital test pin for code development. This allows testing of the speed of routines and to check out conditional branching of routines.

Comparator

Used for overcurrent shutdown.

Op Amp

For current sensing using a single current sense resistor. The Op Amp is used to offset the ground referenced current sense signal to 1 V (half of the 2 V reference); the voltage gain increases by a factor of 5. For a sense resistor value of 20 mΩ and a current of ±4 A gained by 5, a 1 V signal can swing ±0.4 V.

ADC

Used for sampling current and DC bus voltage.

Relay

The main relay is normally open; therefore, the high-voltage power supply is disabled while the microcontroller starts up. The relay initialization pauses for three seconds before closing the relay to ensure that the high-voltage bus capacitors can precharge through a resistor. After closing the relay, the program waits another second to allow the relay contacts to settle.

Current Calibration

After the peripherals are enabled and have settled out 8 samples of the current sense channel are taken with zero current. The average of these samples are used to zero out any offsets in the circuit.

UART

The UART is configured for 9600 baud, 1 start bit, 8 data bits and 1 stop bit. UART transmit and receive are tied together for single wire communication. The 3 wires are ground, data and 5V. The UART service routine updates every 2ms. A more elaborate communication link could be implemented with this hardware.

I^2C

I^2C is used for an external EEPROM for logging product data. For testing purposes, the I^2C interface is used to send data to a 4-channel DAC. In this implementation, a single channel of the DAC is updated every 250µs so that one parameter can be monitored in real time to provide a method for troubleshooting and code development and for field orientation tuning using an oscilloscope. To control the DAC, the Z8FMC16100 MCU is set up as the I^2C Master at 384 kbps, 2.6µs per bit. All data is sent to DAC channel A.

Timer0

Timer 0 is used in Capture Restart Mode, the encoder input rising edge is used to trigger the Capture. If the timer rolls over, the maximum timer count of `0xFFFF` is used, setting it to the minimum speed. If an additional ADC input is required, the same encoder signal can be routed via jumper J2 to PA0 to generate a port interrupt which can capture the timer value.

PWM Module

The 3-phase PWM module is set up in Edge Aligned PWM and the PWM polarity is controlled to Allow Space Vector Modulation and current sampling and reconstruction. Each PWM pair is set up in complementary mode with 12 clock cycles of deadband (0.6 µs). The PWM reload event triggers an ADC conversion; it is used for current reconstruction.

PWM Interrupt

The PWM reload interrupt is enabled. The PWM ISR constitutes the program control loop. PWM interrupts are disabled during the ISR until the ISR is complete.

Main Event Loop

After initializations are complete, program control drops into an infinite *do nothing* loop. All subsequent program functions are performed by the PWM interrupt service routine (ISR), so this ISR is, in effect, the main event loop. The PWM ISR is executed upon every fifth PWM cycle.

Current Sample

The PWM interrupt is configured to automatically start an ADC conversion that samples the high-voltage current, so the PWM ISR begins with an ADC conversion already in progress. The PWM interrupt is edge-aligned and the sample takes place before the PWM state change propagates to the inverter bridge gates. Thus, this first sample captures the current at the end of the just-completed PWM cycle.

After reading the first sample, the ISR immediately begins another ADC conversion. This second sample captures the current at the beginning of the new PWM cycle; see Figure 9. Phase currents are then reconstructed from these samples.

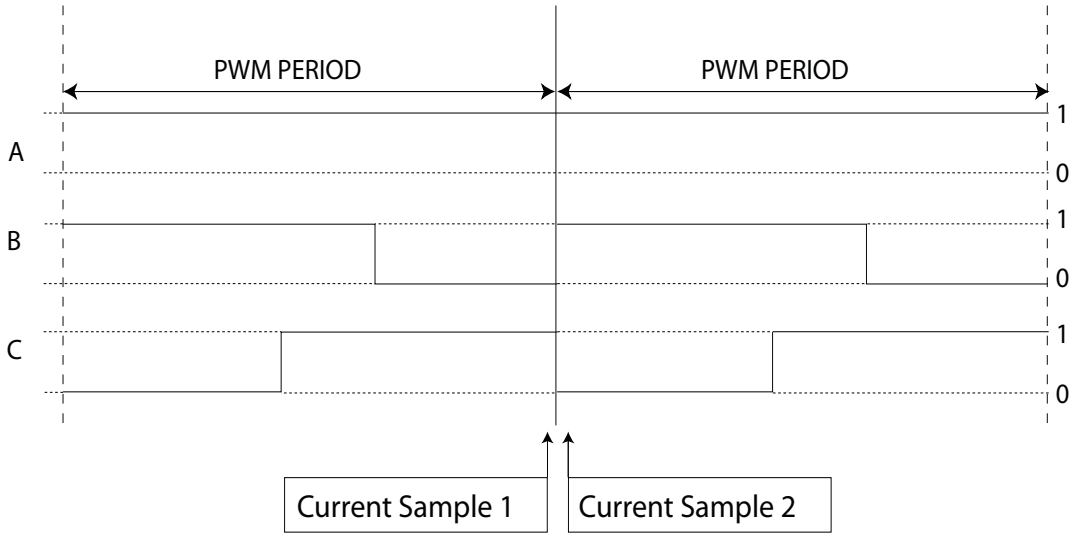

Figure 9. Sampling Space Vector PWM Current (State = 3)

I²C Send Address and Start Bit

Sends the address byte used by the DAC to establish communication. This action begins a side-process of the PWM ISR. I²C-related instructions are inserted at intervals to set up and transmit one test value per ISR loop to an external DAC.

Phase Current Reconstruction

Phase current reconstruction is based on the two current samples previously obtained plus the PWM space vector State. The State value is updated at the end of each ISR pass to reflect the stator flux vector angle arrived at in that pass. State positions are shown in Figure 10. The State determines how each current sample is interpreted, as shown in Table 1 on page 15. In this table, the variables `I_a`, `nI_a`, `I_b`, `nI_b`, `I_c` and `nI_c` represent the positive and negative current variables for phases A, B and C.

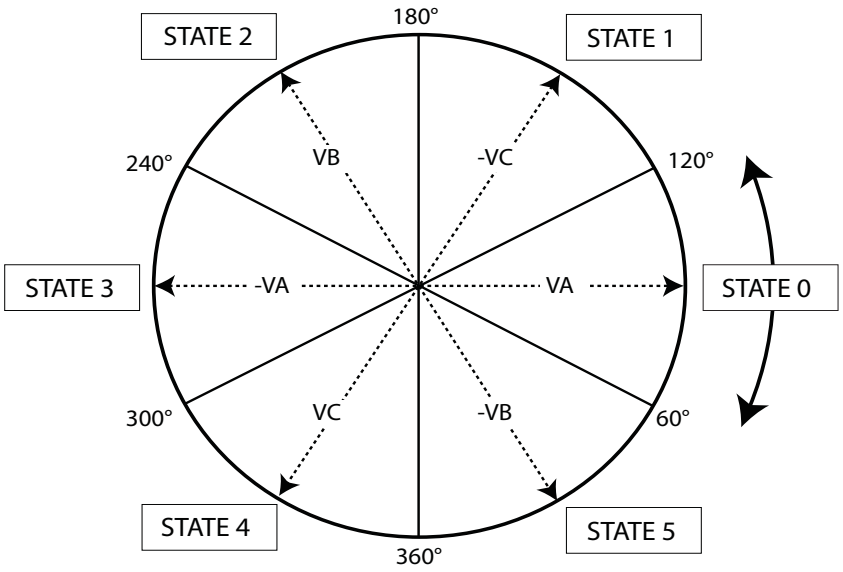

Figure 10. Space Vector PWM States

Table 1. Sample Interpretation by PWM State

PWM State	Current Sample 1 Variable	Current Sample 2 Variable
0	nI_c	nI_b
1	I_b	I_a
2	nI_a	nI_c
3	I_c	I_b
4	nI_b	nI_a
5	I_a	I_c
6	(Not Used)	
7	(Not Used)	

Using Kirchoff's current Law and assuming no zero sequence currents (zero sequence currents are only possible in an imbalanced Delta wound motor not in a Wye winding), the sum of three phase currents equals zero. The third current can be derived from the other two samples using the following equation:

I_a + I_b + I_c = 0

The oscillograph in Figure 11 compares a reconstructed waveform as output to the DAC at 1 A/330 mV (top trace) to the original waveform as read by a current probe at 1 A/10 mV.

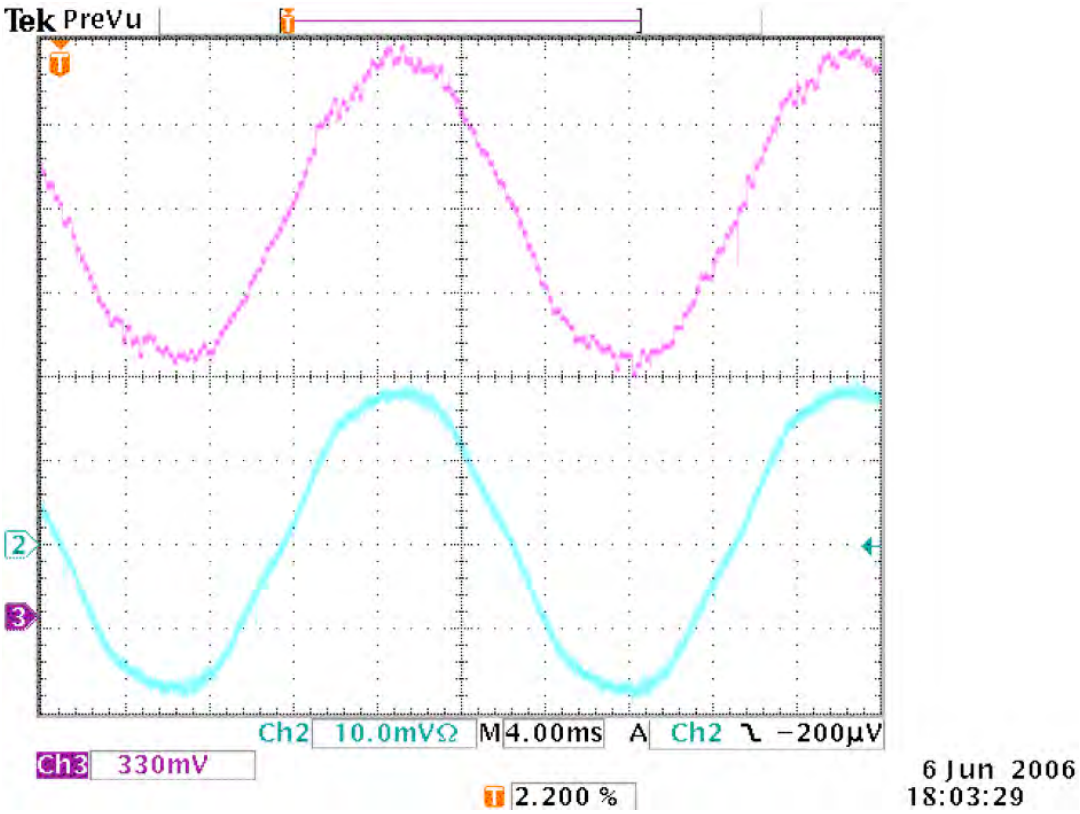

Figure 11. Phase Current Reconstruction

Test for End of PWM Period

This action begins another side-process of the PWM ISR. PWM interrupts remain disabled until the main loop ISR returns. However, at several times during the main loop, the program polls the IRQ0 register to see if a PWM request is pending. If so, the program clears the request and increments a counter. Near the end of the main loop, this event counter is used to delay completion of the process until four PWM request events have occurred, in addition to the one that began the loop. This process ensures a five-event main loop cycle (250µs total) without the requirement for precise timing of each main loop routine.

Clarke Transform

A Clarke transform produces a 2-phase (direct and quadrature) representation of the reconstructed 3-phase stator flux vector. Then a quadrant test is applied to the reference currents `I_ds` and `I_qs` to determine which quadrant the currents are in. This result is used later in the Park Transform to rotate the 2-phase vector. The equation for the Clarke Transform is:

$$I\_ds = I\_a - \frac{1}{2}(I\_\beta + I\_\chi)$$

$$I\_qs = \frac{2}{\sqrt{3}}(I\_\beta + I\_\chi)$$

Angle Update

The angle Theta is updated by the following equation:

Theta = Theta + Speed + Slip

The elements in the above equation can be defined as:

Theta = rotor flux angle relative to the stator phase A winding

Speed = speed (frequency) of the rotor in terms of the update rate of the control loop

Slip = slip speed (frequency) required to produce the proper rotor flux

Sine and Cosine Look-Up Table

Sine and cosine values are generated from an unsigned table of sine values from 0 to 90 degrees contained in a 128-value table (90 degrees ÷ 128steps = 0.7 degrees/step). Two values are taken from the table: one at the angle stored in the variable `Sin_1` and one at the 128-degree angle stored in the variable `Sin_2`.

I²C Send Control Byte to DAC

Sends the control byte used by the DAC to select the DAC output channel.

Park Transform: Vector Rotation

The Park Transform rotates the two phase currents referred to the stationary reference frame of the stator to the rotating reference frame of the rotor flux. This rotation is illustrated in Figure 12 on page 18.

The equations for the Park Transform are:

I_dr = I_ds × cos(Theta) + I_qs × sin(Theta)

I_qr = –I_ds × sin(Theta) + I_qs × cos(Theta)

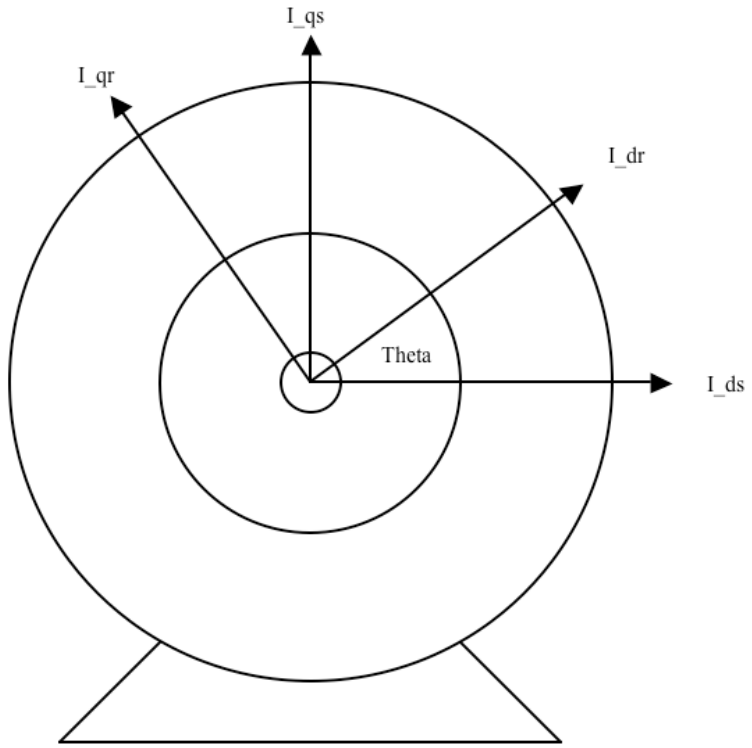

Figure 12. Park Transform

I²C Send MSB to DAC

Sends the 8-bit data used by the DAC. This internal value is sent to the DAC board to allow the display of internal values as voltages on the oscilloscope. The implemented routine sends data to one channel of the DAC updated at the loop rate to allow the real-time display of reference or feedback values.

I_d Regulator

A regulator that controls the direct component of the current vector; it is regulated by a simple PI regulator.

I_q Regulator

A regulator that controls the quadrature component of the current vector; it is regulated by a simple PI regulator.

Quadrant Test

A quadrant test is applied to the `I_dr` and `I_qr` reference currents to determine which quadrant the currents are in. This result is used later in the Inverse Park Transform to

rotate the 2-phase vector. The `I_dr` and `I_qr` reference currents are calculated in the `Speed_reg()` subroutine.

I²C Send LS Byte to DAC

The DAC offers only 8 bits of resolution; therefore it is a *don't care* byte required by the DAC.

Read Bus Voltage

The bus voltage is sampled through the microcontroller's ADC. Currently it is used for ripple compensation; however, it could also be included in field-weakening calculations or used for over-voltage shutdown protection.

Bus Ripple Compensation

This routine tracks changes in the bus voltage and looks up a precalculated ripple compensation factor which is inversely proportional to the ADC sample of the bus voltage. The PWM duty cycle is scaled by this factor to compensate for variations in the DC bus voltage so that a smaller factor value corresponds to a reduced PWM duty cycle which clamps the output waveform to a nominal voltage.

This application uses a nominal bus voltage of 225 VDC. At this voltage, the output factor is 255 (full scale, no compensation). The factor scales down to 0 at 375 VDC or higher, providing an overvoltage shutdown. Between 225 VDC and 375 VDC, the look-up table values are calculated using the following equation:

$$pwm\_correction = \frac{ADC\_vbus\_nom}{ADC\_vbus} \times 255$$

Inverse Park Transform—Vector Rotation

The Inverse Park Transform rotates the two phase voltages referred to the rotating reference frame of the rotor flux to the stationary reference frame of the stator; see Figure 13 on page 20. The equation for the Inverse Park Transform is:

$$V\_ds = V\_dr \times \cos(Theta) - V\_qr \times \sin(Theta)$$

$$V\_qs = V\_dr \times \sin(Theta) + V\_qr \times \cos(Theta)$$

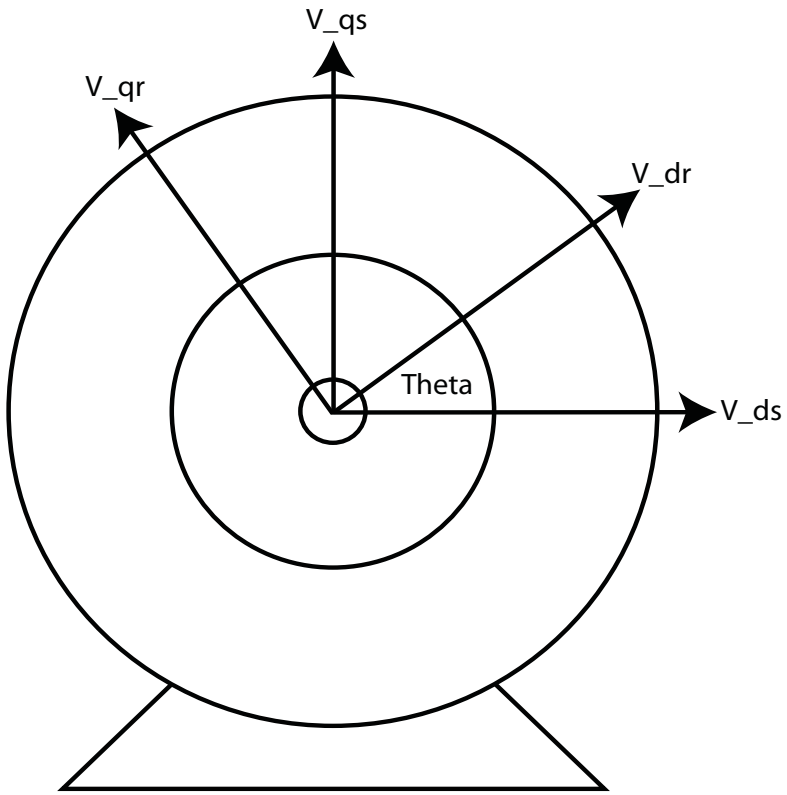

Figure 13. Inverse Park Transform

I²C Send Stop Bit to DAC

A stop bit is sent to terminate the I²C communication.

Inverse Clarke Transform

The Inverse Clarke Transform converts the two phase voltages of the rotated DQ reference frame to the ABC reference frame of the stator windings; see Figure 14 on page 21. The equations for the Inverse Clarke Transform are:

$$V\_a = V\_ds$$

$$V\_b = \frac{1}{2}(V\_ds) + \frac{2}{\sqrt{3}}(V\_qs)$$

$$V\_c = \frac{1}{2}(V\_ds) - \frac{2}{\sqrt{3}}(V\_qs)$$

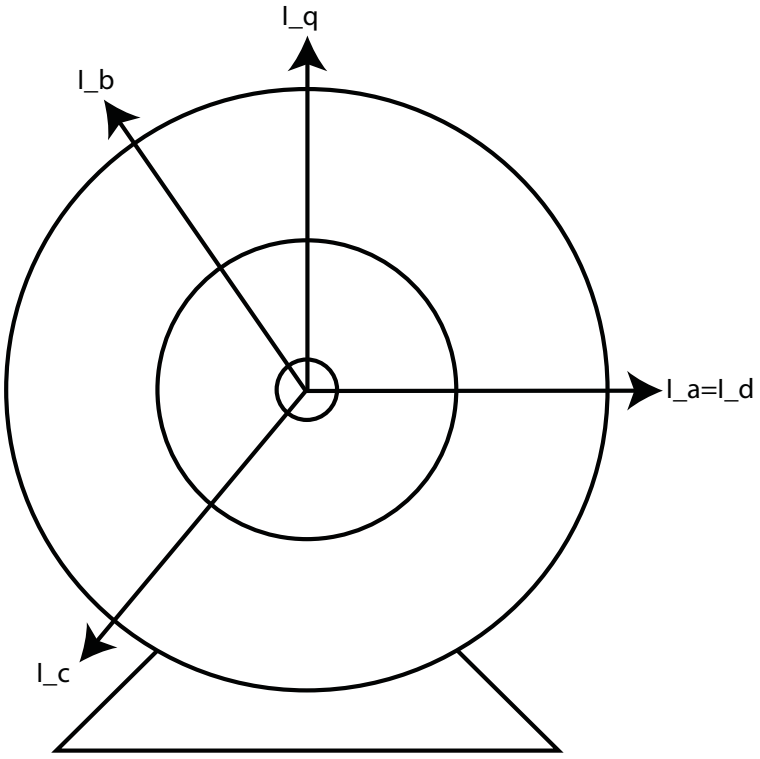

Figure 14. Inverse Clarke Transform

PWM Space Vector Modulation

Space vector modulation starts with the six states that represent the six voltage vectors VA, –VC, VB, –VA, VC and –VB. Table 2 shows the PWM duty cycle calculations used for each state. States 6 and 7 represent the unmodulated phase (OFF or ON).

Table 2. PWM Duty Cycle by State

	PWM Duty Cycle		
PWM State	Phase A	Phase B	Phase C
0	100%	100%– V_b	100%– V_c
1	V_a	V_b	0%
2	100%– V_a	100%	100%– V_c
3	0%	V_b	V_c
4	100%– V_a	100%– V_b	100%
5	V_a	0%	V_c
6	0	0	0
7	100%	100%	100%

Figure 15 illustrates an example of PWM space vector modulation at a PWM magnitude of 75 out of 125, referenced to DC Ground.

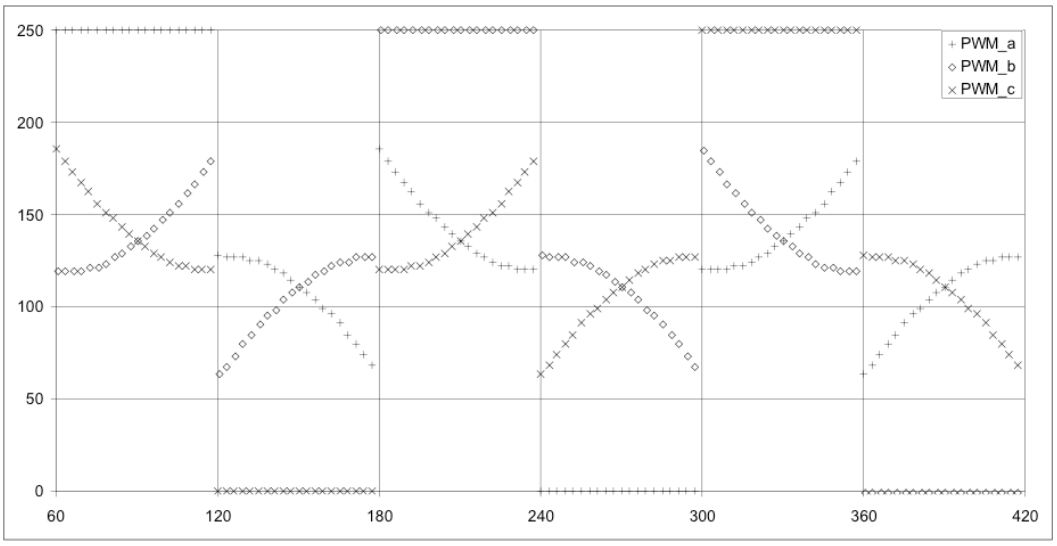

Figure 15. PWM Space Vector Modulation (Ground Reference)

Figure 16 illustrates an example of PWM space vector modulation at a PWM magnitude of 75 out of 125, referenced to Neutral.

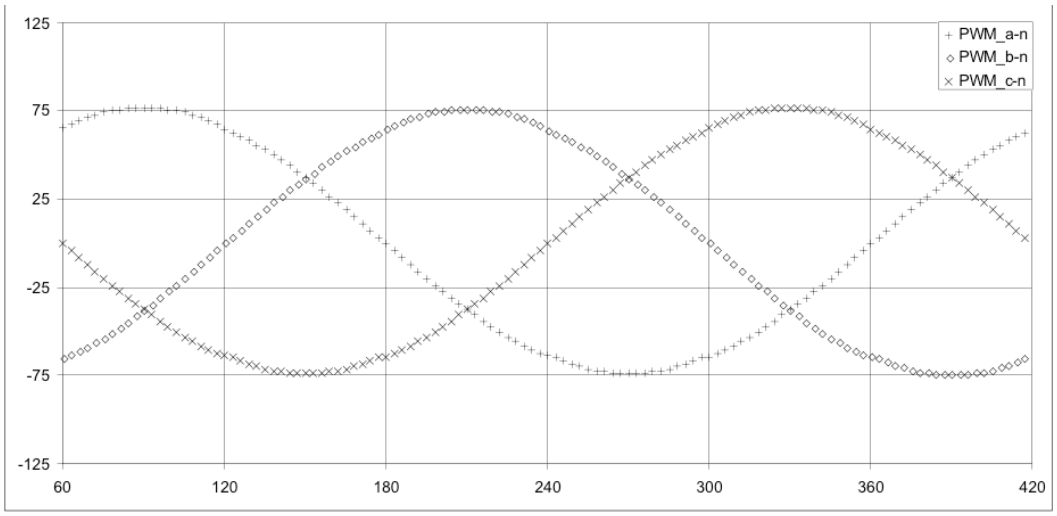

Figure 16. PWM Space Vector Modulation (Neutral Reference)

Watchdog Timer Refresh

Refreshes the Watchdog Timer once every loop update (250µs). A reset occurs if the oscillator or a fault in the code disrupts operation for more than 200ms.

Subfunctions

Subfunctions are subroutines that are not required to be updated every time the PWM ISR executes. There are ten subroutine calls arranged in a round-robin sequence so that only one call is executed per ISR loop. Therefore, each call is executed once every 2ms. Not all round robin slots are required for the currently implemented subfunctions. The remaining slots are available for future expansion.

Tachometer Update

The tachometer update routine looks at the latest capture value from Timer 0. The routine checks for a rollover; if it detects a roll-over, then the maximum counter value of `0xFFFF` is used to calculate speed. Additionally, a rate limit is placed upon the `Speed_step` variable to ensure that noise does not cause erratic speed readings. In the case of an optical encoder, the user can potentially encounter edge jitter when an encoder line is on the edge of the optical pickup and the motor is at idle; movement in the motor can cause fast edges that can be interpreted as sensing high speed. This phenomenon is filtered out by disallowing any jump in speed. The tachometer update function is divided into three subfunction calls to limit the execution time for each call.

Field Weakening

Currently not developed for this application.

Slip Update (Current Model)

Currently not developed for this application.

Speed Regulator

The speed regulator is a standard Proportional Integral (PI) regulator. The command arrives from the UART and the speed feedback arrives from the tachometer.

UART Communication

The UART communicates with the Vector Control GUI and simulates a washing machine control panel. The GUI contains entry fields to enter speed, PI loop values and to start or stop the motor. A speed graph is plotted at user-definable time intervals.

The GUI sends a single command byte that is read at each UART update. Bits 7:1 of the command byte are the speed command (MSB of the control module's ADC sample).

The Acknowledge byte from the motor controller is `AAh`; the UART baud rate is 9600 baud.

Software Metrics

Table 3 lists the execution time for each program initialization routine, executed once at startup.

Table 3. Initialization Execution Times

Routine	Time
init_osc	50ms
init_out	1µs
init_comp	1µs
init_amp	2µs
init_adc	2µs
init_relay	4s
init_current	32µs
init_uart	2µs
init_i2c	3µs
init_timer0	3µs
init_pwm	5µs
init_pwmint	1µs

Table 4 lists the execution time for each routine in the PWM ISR, executed every 250µs (4kHz loop).

Table 4. Main Loop (ISR) Execution Times

Routine	Time (µs)
Current Sample	5
I^2C Send Address and Start Bit	1
Current Reconstruction	14
Clarke Transform (3=>2 conversion)	18
PWM Period Test	1
Theta Update	6
Sine and Cosine Look-Up Table	3
I^2C Send Control Byte	1
Park Transform (vector rotation)	28
PWM Period Test	1
I^2C Send MSB	1
I_d Regulator	13
I_q Regulator	13
Quadrant Test	5

Table 4. Main Loop (ISR) Execution Times (Continued)

Routine	Time (µs)
I^2C Send LSB	1
Bus Voltage Sample	1
Bus Ripple Compensation	3
PWM Period Test	1
Inverse Park Transform (Vector Rotation)	28
PWM Period Test	1
I^2C Send Stop Bit	1
Inverse Clarke Transform (2=>3 Conversion)	18
PWM Period Test	1
PWM Space Vector Modulation	33
PWM Period Test	1
WDT refresh	1
Sub Function (stepped through list)	26
Subtotal	224
End of PWM Period Test	Variable
Total	250

Table 5 lists the execution time for each subfunction routine, executed every 2ms.

Table 5. Subfunction Execution Times

Subroutine	Time (µs)
Tachometer Update 1 of 3	7
Tachometer Update 2 of 3	26
Tachometer Update 3 of 3	2
Field Weakening	Unused
Slip Update	Unused
Speed Regulator	26
UART Communication	4

Summary

The purpose of this application is to demonstrate how a Z8FMC16100 microcontroller is used for the efficient rotor flux field-oriented vector control of a 3-phase AC induction motor. The following features of the Z8FMC16100 MCU make it particularly suited to motor control applications:

- The eZ8 processor core runs at up to 10MIPS with a 20MHz clock.

- The integrated PWM interface module provides the necessary timing and logic outputs for synthesizing 3-phase voltage in motor control applications.
- The 2.5µs successive approximation register (SAR) ADC can be triggered by PWM events for sampling and reconstructing phase currents. The integrated Op Amp amplifies and provides offset adjustment of a single ground-referenced sense resistor, centered to one-half of the built-in ADC reference. The integrated comparator provides for fast overcurrent shutdown.
- The integrated communication peripherals (UART, I^2C and SPI) provide for system-level communication. For example, the UART in this application provides a highly capable Vector Control Command Interface, yet is simple enough to be optically coupled for electrical isolation.

In addition to the Z8FMC16100 MCU's native advantages, the software design in this application uses the following techniques to manage the processing task:

- To make time for the waveform calculations, the main loop operates at one fifth of the 20kHz PWM chop rate (five 50µs cycles, 250µs per loop) to preserve most of the advantages of a high-frequency PWM rate while still producing a good approximation of the appropriate output waveform.
- To free up additional time for main loop calculations, low-priority subfunctions are called in a 10-position round-robin schedule so that each subfunction is executed only once every 2.5ms. The lengthy tachometer update operation is further divided into three subfunction calls to limit execution time per call.
- Converting current measurements to a 2-phase vector for the rotation transform reduces calculation time compared to rotating a 3-phase vector.
- All signed values are expressed as pairs of unsigned variables. This pairing requires extra code to perform an if-then-else routine for each calculation; however, the eZ8 processor's short-range branch instructions are efficient enough so that the unsigned calculations can execute faster than the equivalent library routines, which are written for general-case signed arithmetic.

The example application and techniques described in this document should prove helpful for anyone who intends to develop motor control applications based on the Z8FMC16100 microcontroller.

Appendix A. References

For more information about the topic of controlling motors, plus details about the Z8FMC16100 MCU and its core eZ8 CPU, see the references listed in Table 6.

Table 6. List of References

Topic	Document Name
eZ8 CPU	eZ8 CPU User Manual (UM0128)
Z8FMC16100 MCU	Z8FMC16100 Series Product Specification (PS0246)
Motor Control	Hsu, P.,. *A Short Course on Vector Control*. San Jose State University, pp. 49–54, 1996.
	Garcia, G., Stephan R. & Watanabe E. *Comparing the Indirect Field-Oriented Control with a Scalar Method*. IEEE Trans. Ind. Electronics, Vol. 41 No. 2, pp. 201–207, April 1994.
	Trzynadlowski, M., Andrzej *Control of Induction Motors*. Academic Press, pp. 102–105, 2001.
	Ned Mohan, *Short Course on Electric Drives: Understanding Basics to Advanced Control & Encoder-Less Operation*. University of Minnesota, 2005 (A recording of the Internet-based short course presented on May 12, 2005 by Professor Mohan and edited to fit on a DVD.)

Appendix B. Glossary

Definitions for terms and abbreviations used in this application note that are not commonly known are listed in Table 7.

Table 7. Glossary

Term/Abbreviation	Definition
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
IGBT	Insulated-gate bipolar transistor
ISR	Interrupt Service Routine
PI	Proportional plus Integral
PWM	Pulse Width Modulation
Rotor	Rotating cage and windings of AC motor
Stator	Stationary windings of AC motor

Appendix C. Schematic Diagrams

Figures 17 through 22 present the schematic diagrams for the vector control application modules.

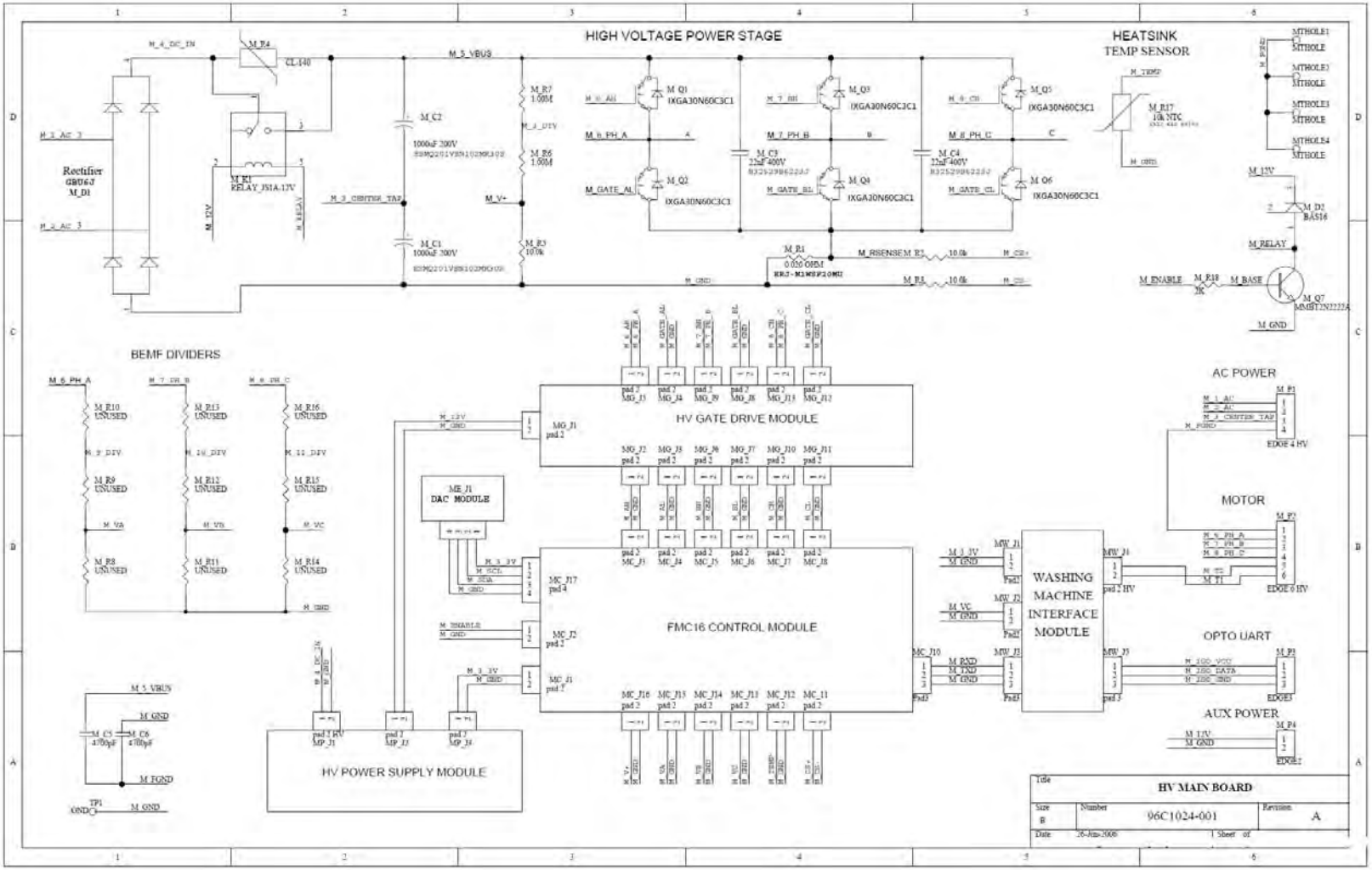

Figure 17. High-Voltage Main Board Schematic

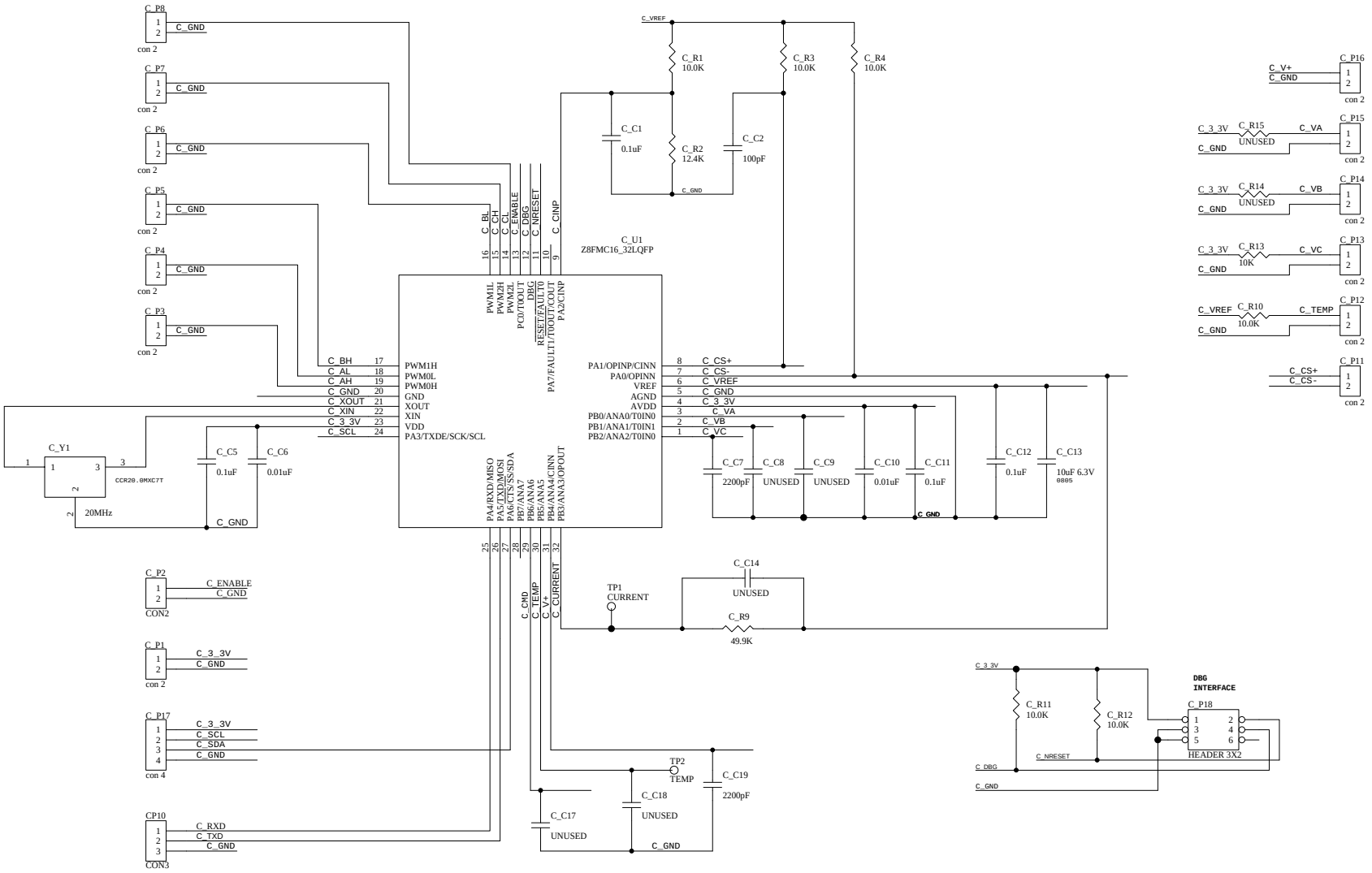

Figure 18. Z8FMC16100 Control Module Schematic

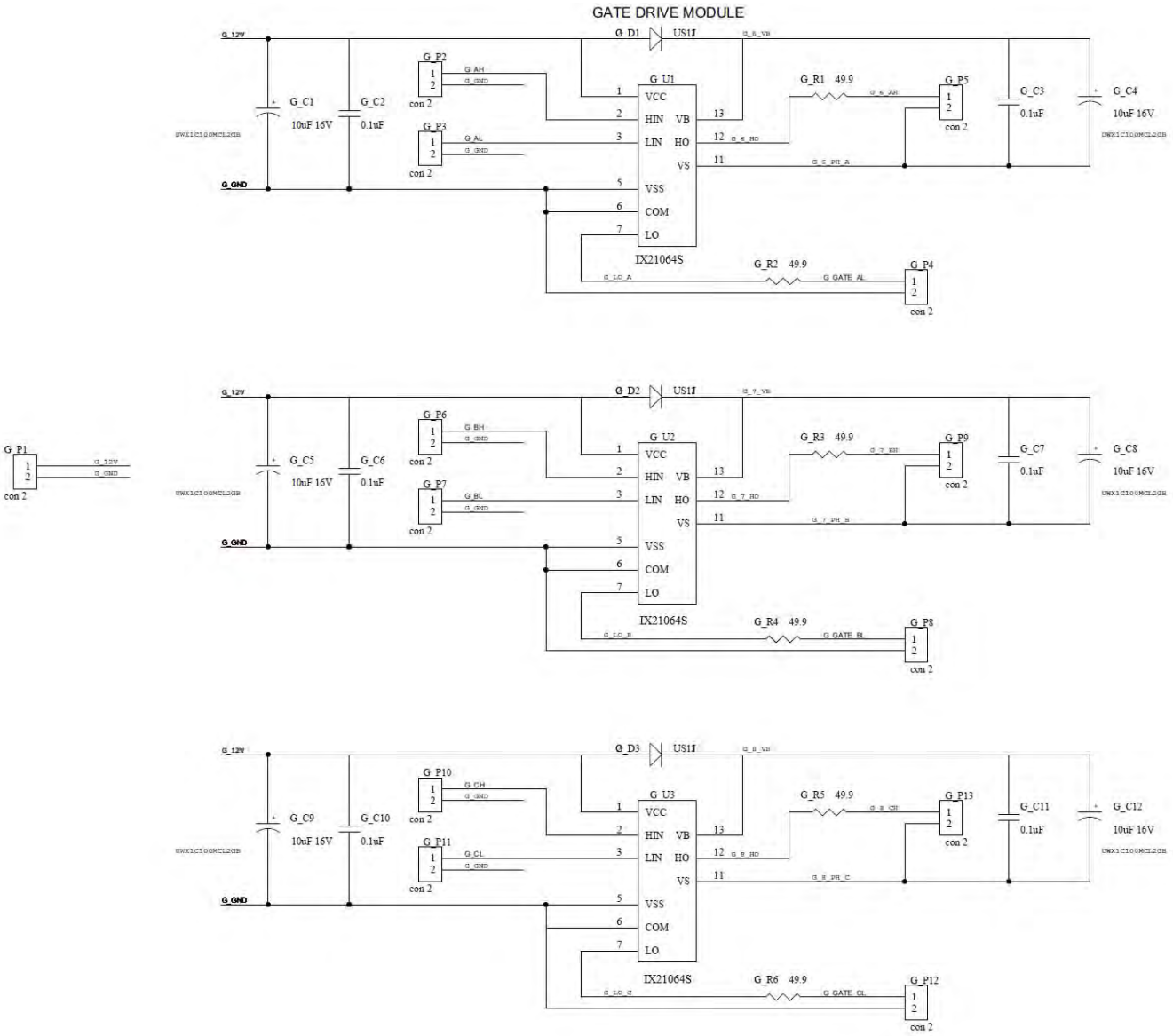

Figure 19. HV Gate Drive Module Schematic

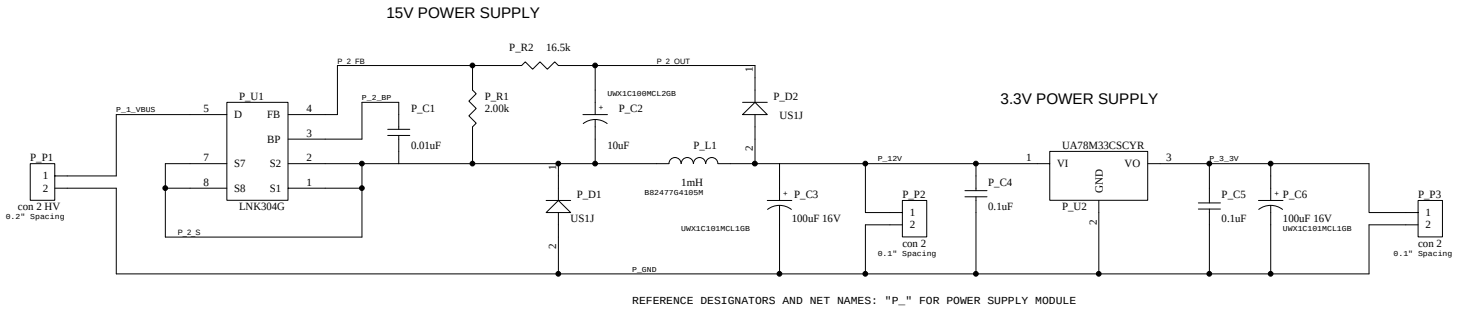

Figure 20. HV Power Module Schematic

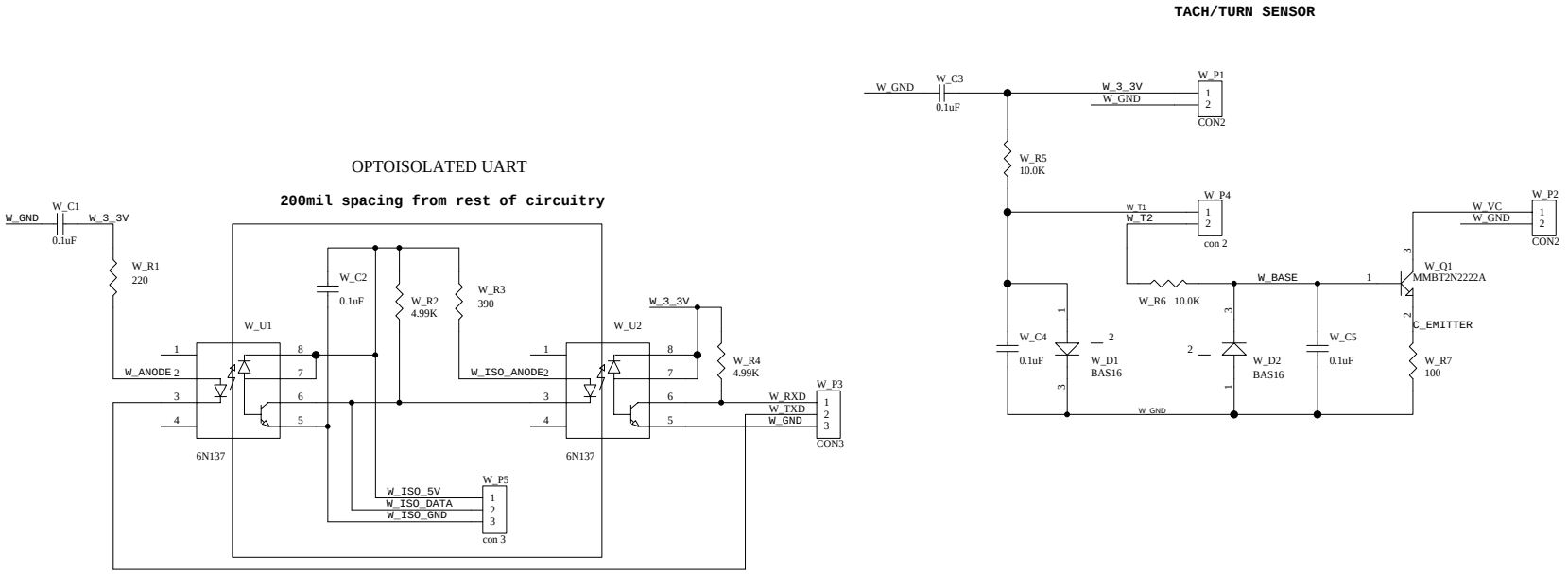

Figure 21. Washing Machine Interface Module Schematic

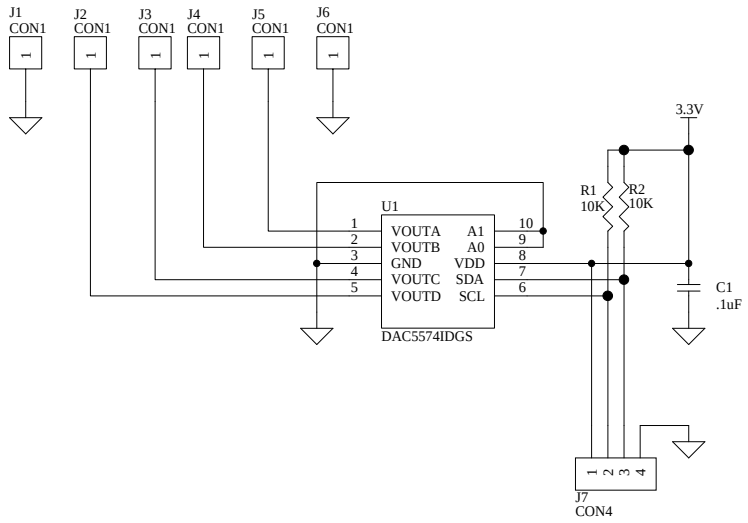

Figure 22. DAC Module Schematic

Appendix D. Flowcharts

Figure 23 illustrates the overall program flow. After `main()` initializes the application, it drops into an infinite loop. Meanwhile, the PWM timer generates an event every 50µs. The PWM ISR is timed to execute on every fifth PWM event, once every 250µs. The ISR calls one of ten low-priority subfunctions on every loop cycle; therefore, each subfunction is called once every 2.5ms.

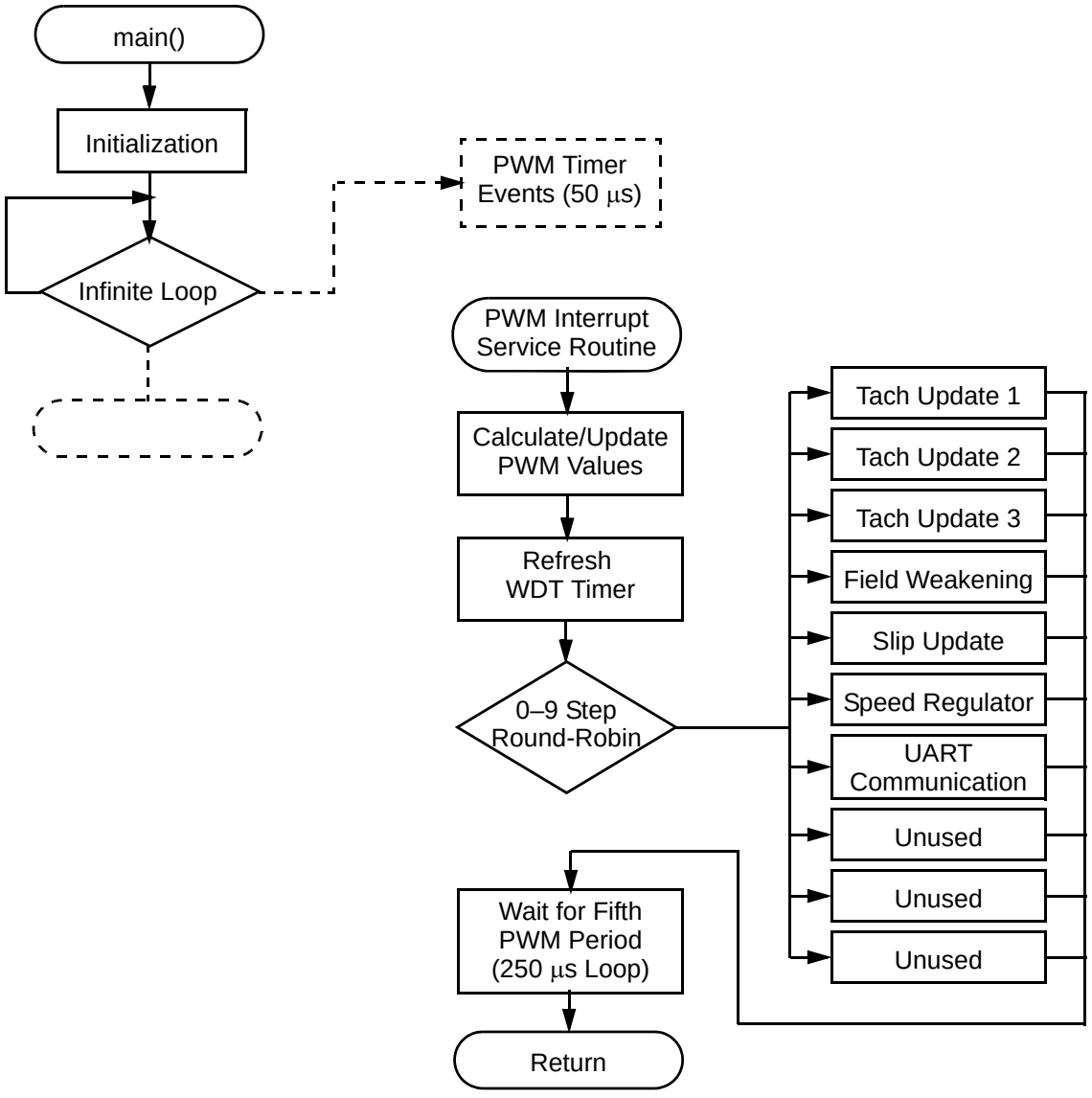

Figure 23. Flowchart for the Application

Figure 24 illustrates the PWM ISR main loop.

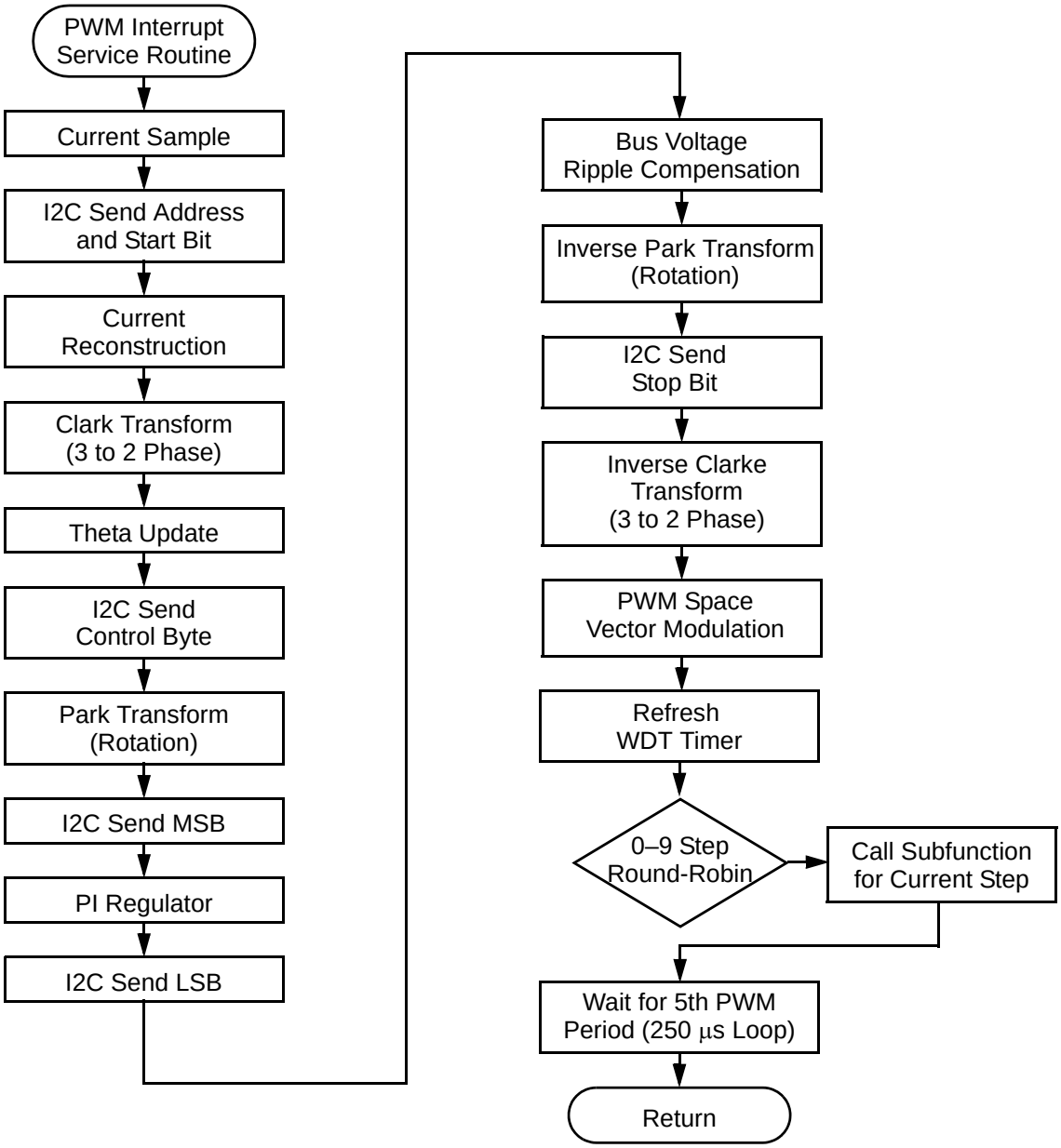

Figure 24. Flowchart for the PWM ISR

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.