MOTOROLA

Order by AN1958/D
(Motorola Order Number)
Rev. 0, 9/2003

3-Phase AC Motor Control with V/Hz Speed Closed Loop Using the 56F800/E

Design of a Motor Control Application Based on Processor Expert

Contents

1. Introduction

This application note describes the design of a 3-phase AC induction motor drive with Volts per Hertz control in closed-loop (V/Hz CL). It is based on Motorola's 56F800/E hybrid microcontrollers, which are ideal for motor control applications. The system is designed as a motor control system for driving medium-power, 3-phase AC induction motors. The part is targeted toward applications in both the industrial and home appliance industries, such as washing machines, compressors, air conditioning units, pumps, or simple industrial drives.

The drive introduced here is intended as an example of a 3-phase AC induction motor drive. The drive serves as an example of AC V/Hz motor control system design using Motorola hybrid controller with Processor Expert™ (PE) support.

This document includes the basic motor theory, system design concept, hardware implementation, and software design, including the PC master software visualization tool inclusion.

2. Motorola Hybrid Controller Advantages and Features

The Motorola 56F800/E families are ideal for digital motor control, combining the DSP's calculation capability with the MCU's controller features on a single chip. These controllers offer a rich dedicated peripherals set, such as Pulse Width Modulation (PWM) modules, Analog-to-Digital Converter

(ADC), timers, communication peripherals (SCI, SPI, CAN), on-board Flash and RAM. Several parts comprise the family: 56F80x with different peripherals and on-board memory configurations. Generally, all are suited for motor control.

A typical member of the 56F800 family, the 56F805, provides the following peripheral blocks:

- Two Pulse Width Modulators (PWMA & PWMB), each with six PWM outputs, three current status inputs, and four fault inputs, fault-tolerant design with dead time insertion; supports both center- and edge- aligned modes
- Two 12-bit, Analog-to-Digital Converters (ADCs), supporting two simultaneous conversions with dual 4-pin multiplexed inputs; 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 MPIO lines
- Serial Peripheral Interface (SPI), with configurable 4-pin port, or four additional MPIO lines
- Computer Operating Properly (COP) timer
- Two dedicated external interrupt pins
- Fourteen dedicated multiple purpose I/O (MPIO) pins and 18 multiplexed MPIO pins
- External reset pin for hardware reset
- JTAG/on-chip emulation (OnCE™)
- Software-programmable, phase lock loop-based frequency synthesizer for the hybrid controller core clock

The Pulse Width Modulation (PWM) block offers high freedom in its configuration, enabling efficient control of the AC induction motor.

The PWM block has the following features:

- Three complementary PWM signal pairs, or six independent PWM signals
- Features of complementary channel operation
- Dead time 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 reference signals
- 15 bits of resolution
- Half-cycle reload capability
- Integral reload rates from one to 16
- Individual software-controlled PWM outputs
- Programmable fault protection
- Polarity control
- 20-mA current sink capability on PWM pins
- Write-protectable registers

The PWM outputs are configured in the complementary mode in this application.

3. Target Motor Theory

3.1 3-phase AC Induction Motor Drives

The AC induction motor is a workhorse with adjustable speed drive systems. The most popular type is the 3-phase, squirrel-cage AC induction motor. It is a maintenance-free, less noisy and efficient motor. The stator is supplied by a balanced 3-phase AC power source.

The synchronous speed n_s of the motor is calculated by:

$$n_s = \frac{120 \times f_s}{p} \quad [rpm] \qquad \text{(EQ 3-1.)}$$

where f_s is the synchronous stator frequency in Hz, and p is the number of stator poles. The load torque is produced by slip frequency. The motor speed is characterized by a slip s_r:

$$s_r = \frac{(n_s - n_r)}{n_s} = \frac{n_{sl}}{n_s} \quad [-] \qquad \text{(EQ 3-2.)}$$

where n_r is the rotor mechanical speed and n_{sl} is the slip speed, both in rpm. **Figure 3-1** illustrates the torque characteristics and corresponding slip. As can be seen from **EQ 3-1** and **EQ 3-2**, the motor speed is controlled by variation of a stator frequency with the influence of the load torque.

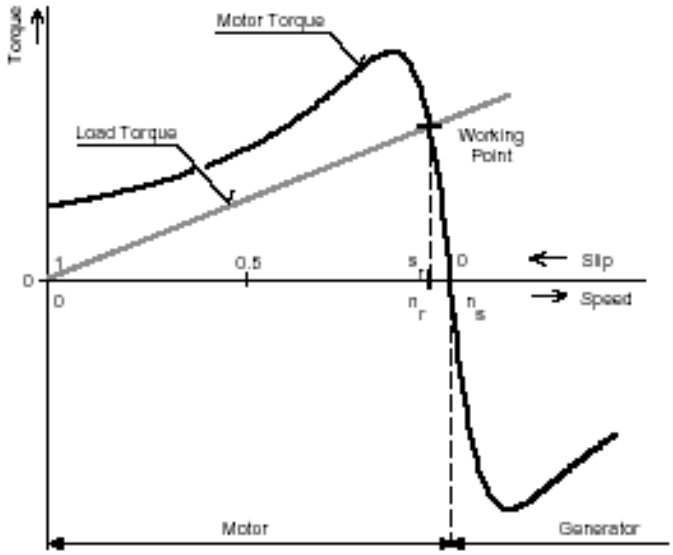


Figure 3-1. Torque-Speed Characteristic at Constant Voltage and Frequency

In adjustable speed applications, the AC motors are powered by inverters. The inverter converts DC power to AC power at required frequency and amplitude. The typical 3-phase inverter is illustrated in **Figure 3-2.**

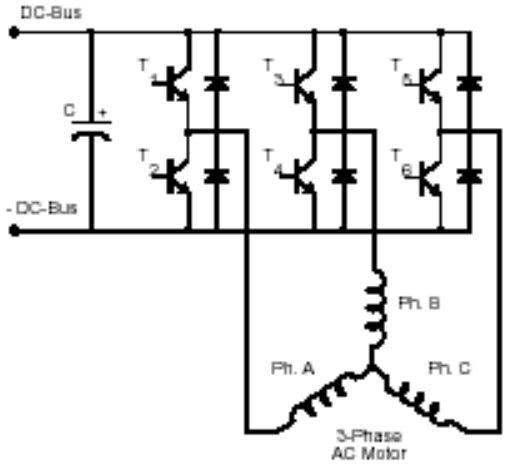


Figure 3-2. 3- Phase Inverter

The inverter consists of three half-bridge units; the upper and lower switches are controlled complementarily, which means that 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 3-3.** The 3-phase voltage waves are shifted 120^{o} to each other and thus a 3-phase motor can be supplied.

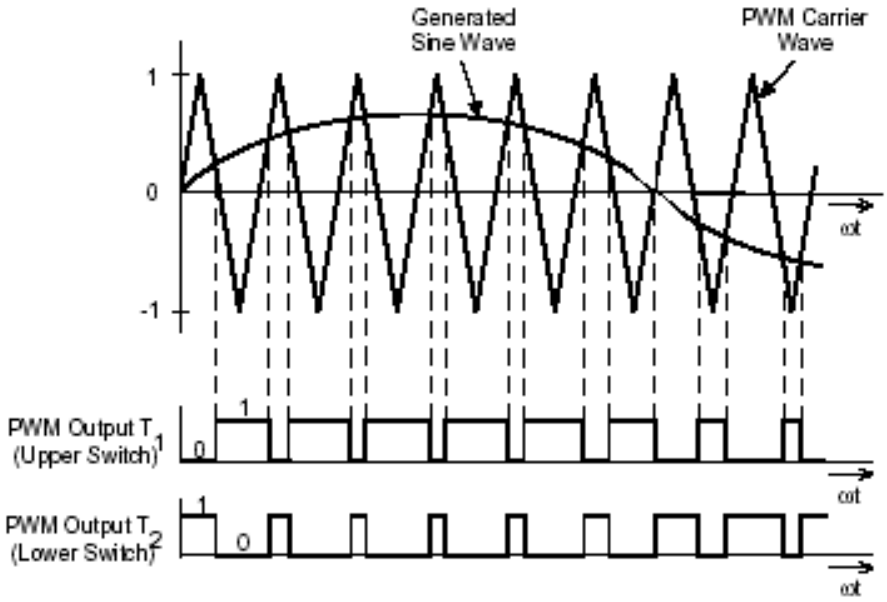


Figure 3-3. Pulse Width Modulation

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, resulting in low-power losses. However, the saturation temperature sensitivity limits the MOSFET application in high-power applications.

An Insulated Gate Bipolar Transistor (IGBT) is a bipolar transistor 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.

3.2 Volts per Hertz Control

The Volts per Hertz control method, the most popular technique of Scalar Control, controls the magnitude of such variables as frequency, voltage or current. The command and feedback signals are DC quantities, and are proportional to the respective variables.

The purpose of the Volts per Hertz control scheme is to maintain the air-gap flux of AC induction motor in constant, achieving higher run-time efficiency. In steady-state operation, the machine air-gap flux is approximately related to the ratio V_s/f_s, where V_s is the amplitude of motor phase voltage and f_s is the synchronous electrical frequency applied to the motor. The control system is illustrated in **Figure 3-4.** The characteristic is defined by the base point of the motor. Below the base point, the motor operates at optimum excitation due to the constant V_s/f_s ratio. Above this point, the motor operates under-excited because of the DCBus voltage limit.

A simple closed-loop Volts per Hertz speed control for an induction motor is the control technique targeted for low-performance drives. This basic scheme is unsatisfactory for more demanding applications, where speed precision is required.

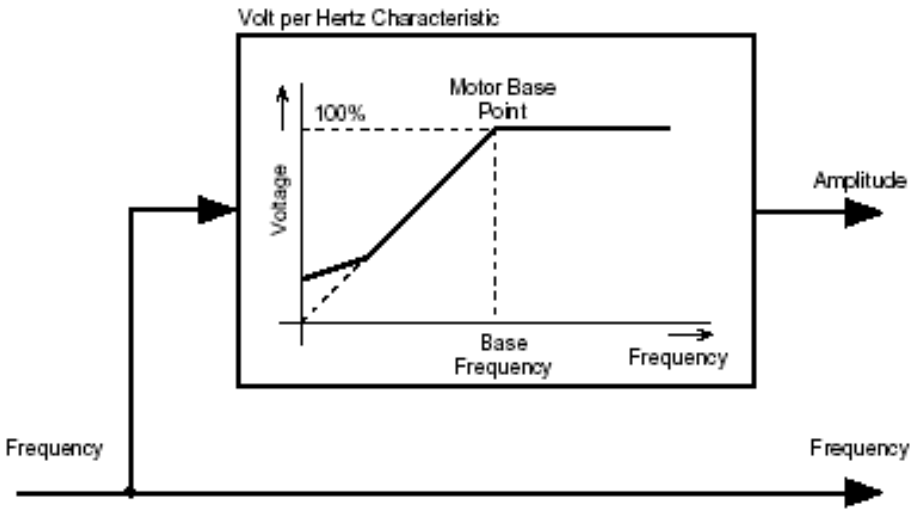


Figure 3-4. Volts per Hertz Control Method

3.3 Speed Closed-Loop System

To improve system performance, a closed-loop Volts per Hertz control was introduced. In this method, a speed sensor measures the actual motor speed and the system takes this input into consideration. A number of applications use the closed-loop Volts per Hertz method because of its simple and relatively good speed accuracy, but it is not suitable for systems requiring servo performance or excellent response to highly dynamic torque/speed variations.

Figure 3-5 illustrates the general principle of the speed PI control loop.

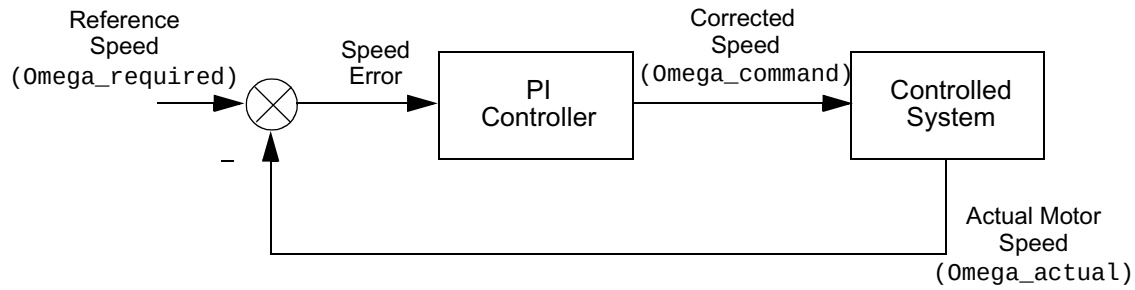


Figure 3-5. Closed Loop Control System

The speed closed-loop control is characterized by the measurement of the actual motor speed. This information is compared with the reference speed while the error signal is generated. The magnitude and polarity of the error signal correspond to the difference between the actual and required speed. Based on the speed error, the PI controller generates the corrected motor stator frequency to compensate for the error.

In an AC V/Hz closed-loop application, the feedback speed signal is derived from the incremental encoder using the Quadrature Decoder. The speed controller constants have been experimentally tuned according to the actual load.

4. System Design Concept

The system is designed to drive a 3-phase AC induction motor. The application meets the following performance specifications:

- Targeted for 56F800/E EVM platforms
- Running on 3-phase ACIM motor control development platform at variable line voltage 115 - 230V AC
- Control technique incorporates
  - — motoring and generating mode
  - — bi-directional rotation
  - — V/Hz speed closed-loop
- Manual interface (Start/Stop switch, Up/Down push button speed control, LED indication)
- PC master software interface (motor start/stop, speed set-up)
- Power stage identification
- Overvoltage, undervoltage, overcurrent, and overheating fault protection

The AC drive introduced here is designed as a system that meets the general performance requirements in **Table 4-1**.

Table 4-1. Motor / Drive Specification

Motor Characteristics	Motor Type	Four poles 3-Phase, star-connected, squirrel cage AC motor (standard industrial motor)
	Speed Range	< 5000rpm
	Base Electrical Frequency	50Hz
	Max. Electrical Power	180 W
	Delta Voltage (rms)	200V (Star)
Drive Characteristics	Transducers	IRC -1024 pulses per revolution
	Speed Range	<2250 rpm @ 230 V <1200 rpm @ 115 V
	Line Input	230V / 50Hz AC 115V / 60Hz AC
	Maximum DCBus Voltage	400V
	Control Algorithm	Closed-Loop Control
	Optoisolation	Required
Load Characteristic	Type	Varying

The hybrid controller runs the main control algorithm and generates 3-phase PWM output signals for the motor inverter according to the user's interface input and feedback signals.

A standard system concept is chosen for the drive, illustrated in **Figure 4-1.** The system incorporates the following hardware boards:

- Power supply rectifier
- 3-phase inverter

- Feedback sensors:
  - Speed
  - DCBus voltage
  - DCBus current
  - Temperature
- Optoisolation
- Evaluation board

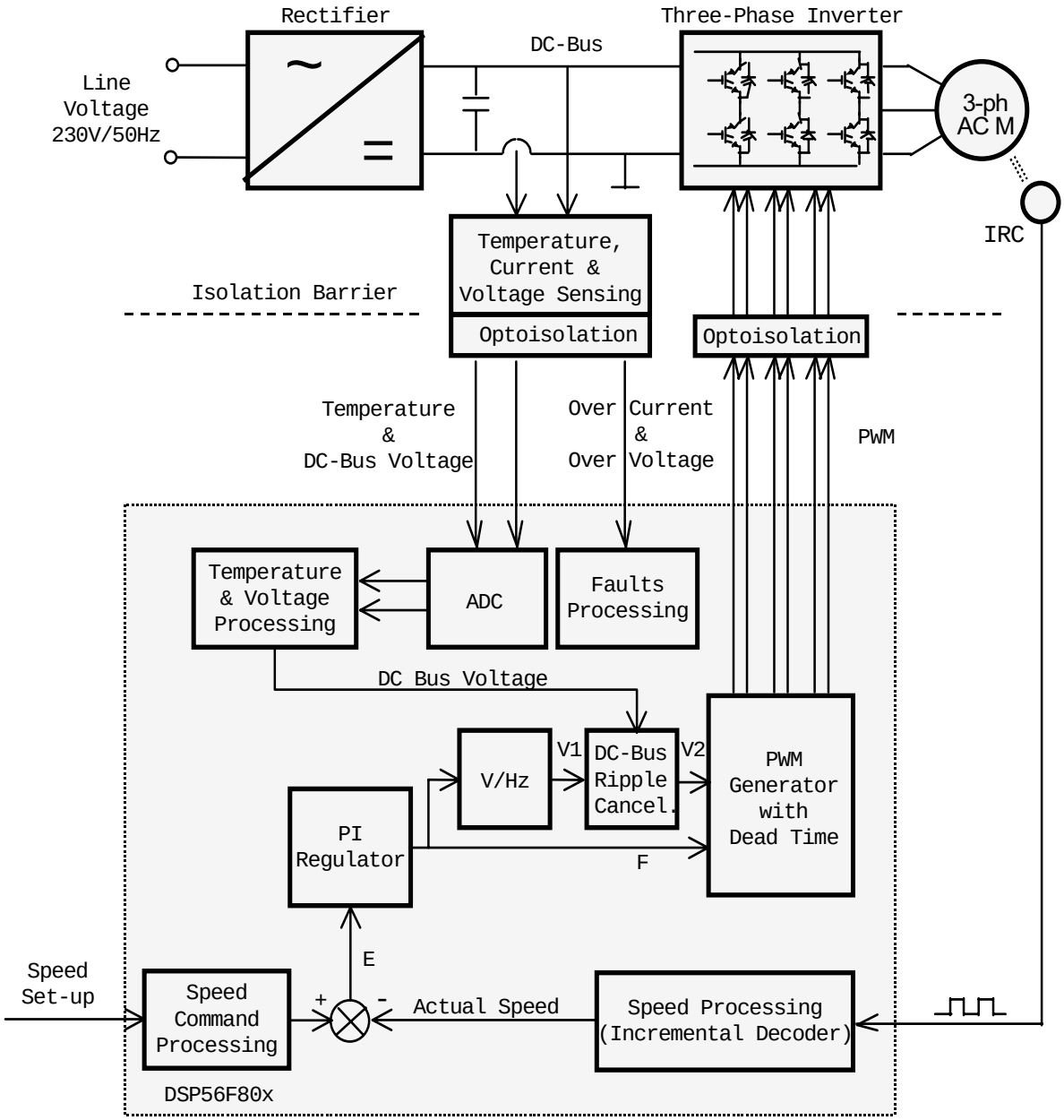


Figure 4-1. System Concept

The Control Process:

When the start command is accepted, using the Start/Stop switch, the state of the inputs is periodically scanned. According to the state of the control signals (Start/Stop switch, speed up/down buttons or PC master software set speed), the speed command is calculated using an acceleration/deceleration ramp.

The comparison between the actual speed command and the measured speed generates a speed error, E. The speed error is brought to the speed PI controller, which generates a new corrected motor stator frequency. With the use of the V/Hz ramp, the corresponding voltage is calculated and then DCBus ripple cancellation function then eliminates the influence of the DCBus voltage ripples to the generated phase voltage amplitude. The PWM generation process calculates a 3-phase voltage system at the required amplitude and frequency, including dead time. Finally, the 3-phase PWM motor control signals are generated.

The DCBus voltage and power stage temperature are measured during the control process. They protect the drive from overvoltage, undervoltage, and overheating. Both undervoltage protection and overheating are performed by ADC and software, while the DCBus overcurrent and overvoltage fault signals are connected to PWM fault inputs.

If any of the above-mentioned faults occurs, the motor control PWM outputs are disabled to protect the drive and the fault state of the system is displayed in PC master software control page.

5. Hardware

5.1 System Outline

The motor control system is designed to drive the 3-phase AC motor in a speed-closed loop.

Software is targeted for these hybrid controllers and evaluation modules (EVMs):

- 56F805
- 56F8346

The hardware set-up depends on the evaluation module (EVM) module used.

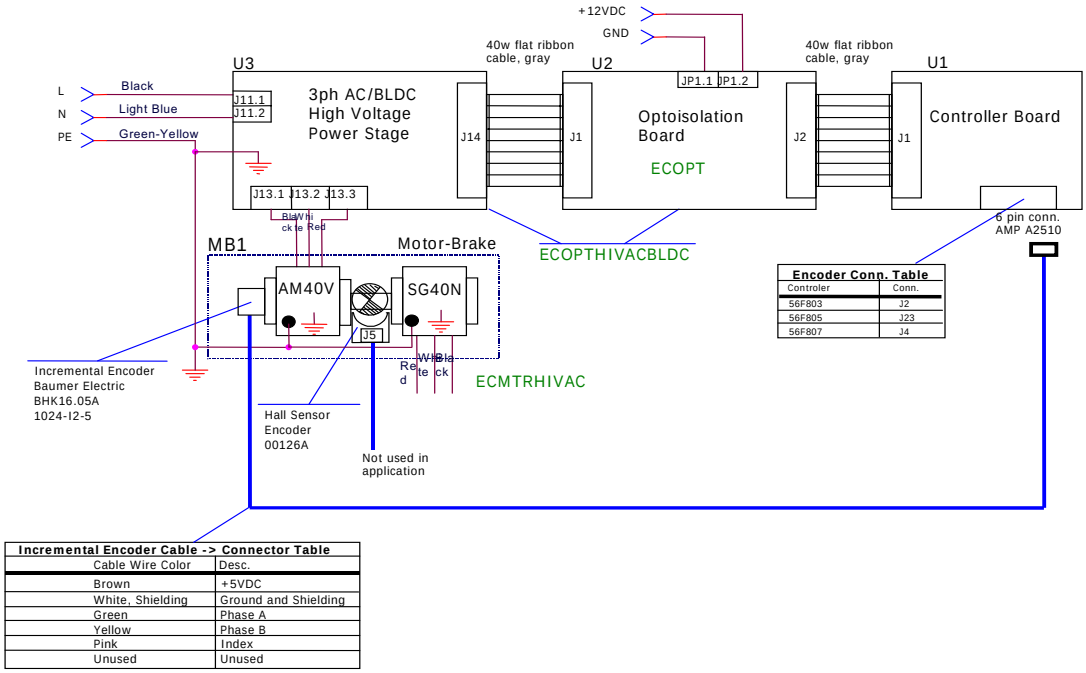
The software can run only on the high-voltage hardware set described in **Section 5.2**.

Other power module boards will be denied, due to the board identification build in the software. This feature protects misuse of hardware module.

The hardware set-up is shown in **Figure 4-1,** but it can also be found in the documentation for the device being implemented.

5.2 High-Voltage Hardware Set

The system configuration is shown in **Figure 5-1.**



Encoder Conn. Table

Controler	Conn.
56F803	J2
56F805	J23
56F807	J4

Incremental Encoder Cable -> Connector Table

Cable Wire Color	Desc.
Brown	+5VDC
White, Shielding	Ground and Shielding
Green	Phase A
Yellow	Phase B
Pink	Index
Unused	Unused

Figure 5-1. High-Voltage Hardware System Configuration

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

- U1 - Controller board:
  - Supplied as: 56F80x or 56F83xx EVM
  - Described in: the 56F80x or 56F83xx **Evaluation Module Hardware User's Manual** for the device being implemented
- U2 - 3-phase AC/BLDC high-voltage power stage
  - Supplied in kit with optoisolation board, Order # ECOPTHIVACBLDC
  - described in: Described in: **3-Phase Brushless DC High-Voltage Power Stage,** Order # MEMC3BLDCPSUM/D
- U3 - Optoisolation board
  - Supplied with 3-phase AC/BLDC high-voltage power stage, Order # ECOPTHIVACBLDC

  or

  - Supplied alone, Order # ECOPT
  - Described in: **Optoisolation Board User's Manual**
- MB1 motor-brake AM40V + SG40N
  - Order # ECMTRHIVAC

Warning: It is strongly recommended that you use optoisolation (optocouplers and optoisolation amplifiers) during development to avoid any damage to the development equipment.

Note: A detailed description of individual boards can be found in the comprehensive user's manual for each board. The manual incorporates the schematic of the board, a description of individual function blocks, and a bill of materials. Individual boards can be ordered from Motorola as a standard product; see **Section 9** for information.

6. Software Design

This section describes the design of the drive's software blocks and includes data flow and state diagrams.

6.1 Data Flow

The drive's requirements dictate that the software gather values from the user interface and sensors, process them, and generate 3-phase PWM signals for the inverter.

The control algorithm of the closed-loop AC drive is described in **Figure 6-1.** The control algorithm's processes are described in the following subsections. The detailed description is given to the subroutine's 3-phase PWM calculation and Volts per Hertz control algorithm.

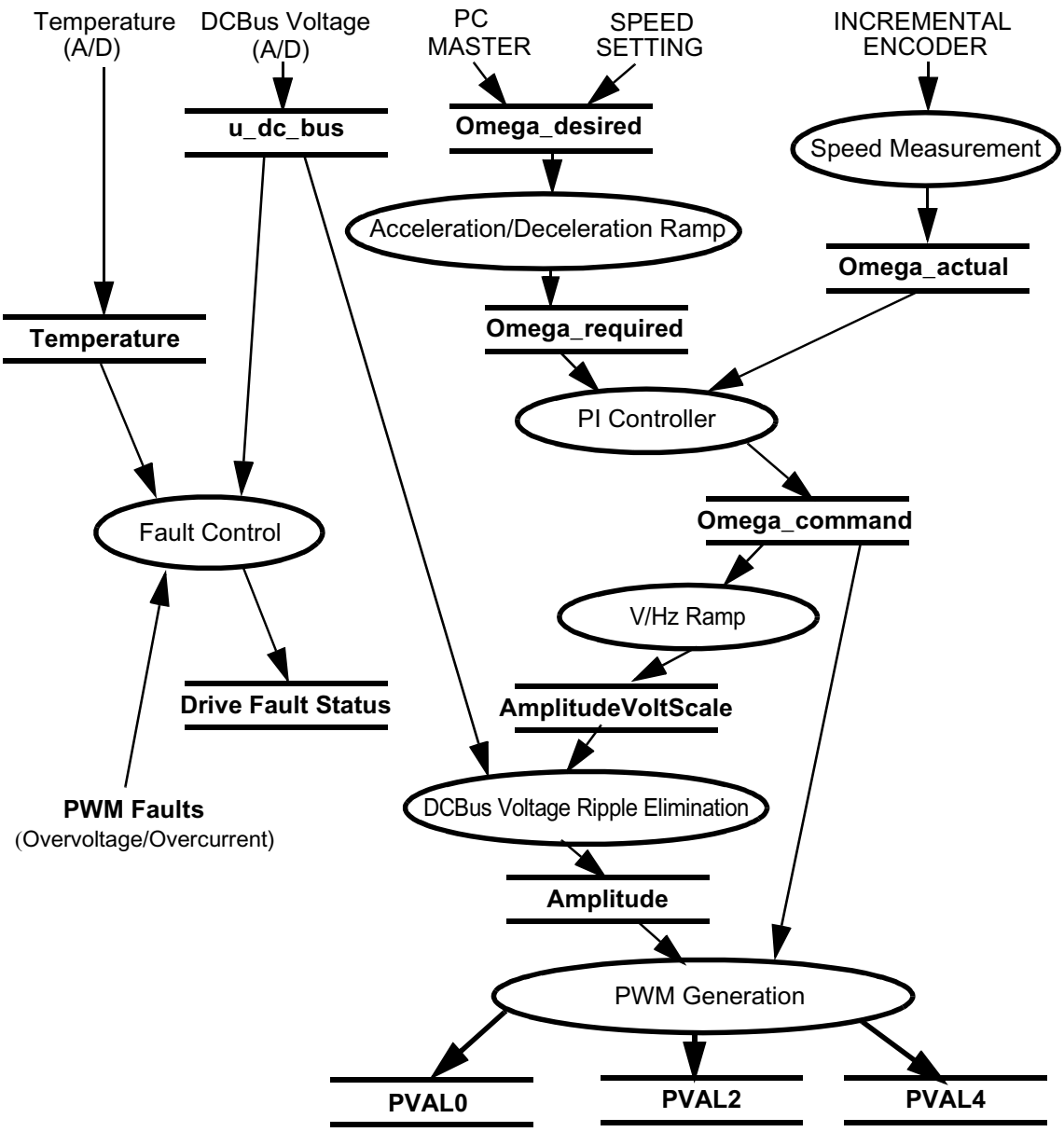


Figure 6-1. Data Flow

6.1.1 Acceleration/Deceleration Ramp

The process calculates the new actual speed command based on the required speed and according to the acceleration/deceleration ramp. The desired speed is determined either by push buttons or by the PC master software.

During deceleration, the motor can work as a generator. In the generator state, the DCBus capacitor is charged and its voltage can easily exceed its maximum voltage. Therefore, the voltage level in the DCBus link is controlled by a resistive brake, operating in case of overvoltage.

The process input parameter is *Omega_desired,* the desired speed.

The process output parameter is *Omega_required,* used as an input parameter of the PWM generation process.

6.1.2 Speed Measurement

The speed measurement process uses the on-chip Quadrature Decoder. The process output is *MeasuredSpeed,* and is only used as an information value in PC master software.

6.1.3 PI Controller

The PI controller process takes the input parameters, actual speed command *Omega_required,* and actual motor speed, measured by a incremental encoder *Omega_actual,* then calculates a speed error and performs the speed PI control algorithm. The output of the PI controller is a frequency of the first harmonic sine wave to be generated by the inverter, *Omega_command*.

6.1.4 V/Hz Ramp

The drive is designed as a Volts per Hertz drive, which means the control algorithm keeps the constant motor's magnetizing current (flux) by varying the stator voltage with frequency. A commonly used Volts per Hertz ramp of a 3-phase AC induction motor is illustrated in **Figure 6-2.**

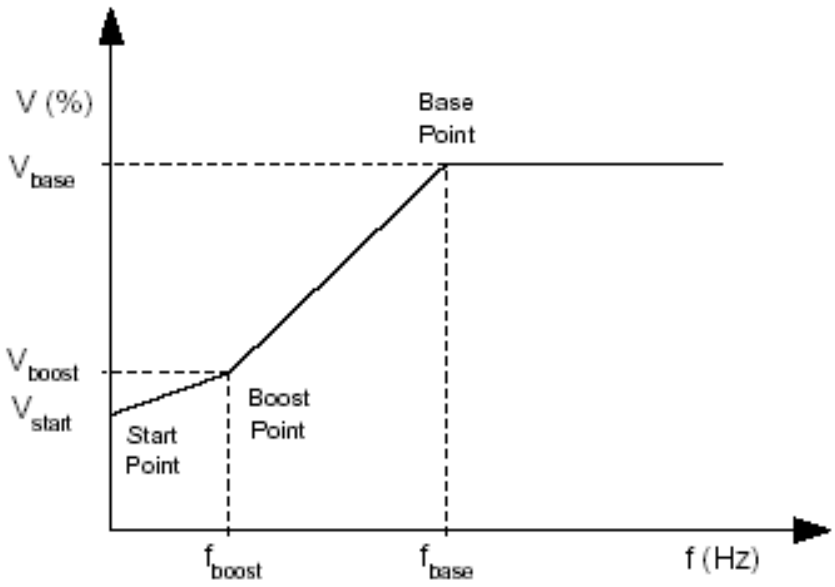


Figure 6-2. Volt per Hertz Ramp

The Volts per Hertz ramp is defined by following parameters:

- Base point - defined by f_{base} (usually 50Hz or 60Hz)
- Boost point- defined by $V_{boost}$and f_{boost}
- Start point - defined by V_{start} at zero frequency

The ramp profile fits to the specific motor and can be easily changed to accommodate different motors.

Process Description

This process provides voltage calculation according to Volts per Hertz ramp.

The input of this process is generated by desired inverter frequency, *Omega_required.*

The output of this process is *AmplitudeVoltScale,* a parameter required by DCBus voltage ripple elimination process.

6.1.5 **DCBus** Voltage Ripple Elimination

Process Description

The voltage ripple elimination process eliminates the influence of the DCBus voltage ripples to the generated phase voltage sine waves. In fact, it lowers the 50Hz or 60Hz acoustic noise of the motor. Another positive aspect due to this function is the generated phase voltage, which is independent of the level of DCBus voltage, making the application easily adapted to power supply systems worldwide.

The process is performed by the *mcgenDCBVoltRippleElim* method of the *MC_WaveGenerate* bean, converting the phase voltage amplitude (*AmplitudeVoltScale*) to the sine wave amplitude (*Amplitude*), based on the actual value of the DCBus voltage (*u_dc_bus*) and the inverse value of the modulation index (*ModulationIndexInverse*).

The *modulation index* is the ratio between the maximum amplitude of the first harmonic of the phase voltage (in voltage scale) and half of the DCBus voltage (in voltage scale), which is defined by the following formula:

$$m_i = \frac{U^{(1)}_{phasemax}}{\frac{1}{2} \cdot u_{DCBus}} = \frac{2}{\sqrt{3}} \qquad \text{(EQ 6-1.)}$$

The modulation index is specific to a given 3-phase generation algorithm; in this application, it is 1.27.

Note: The result of the modulation index is based on the 3rd harmonic injection PWM technique.

The first chart in **Figure 6-3** demonstrates how the *Amplitude* (in scale of generated sine wave amplitude) is counter-modulated to eliminate the DCBus ripples. The second chart delineates the duty cycles generated by one of the 3-phase wave generation functions. The third chart contains symetrical sine waves of the phase-to-phase voltages actually applied to the 3-phase motor.

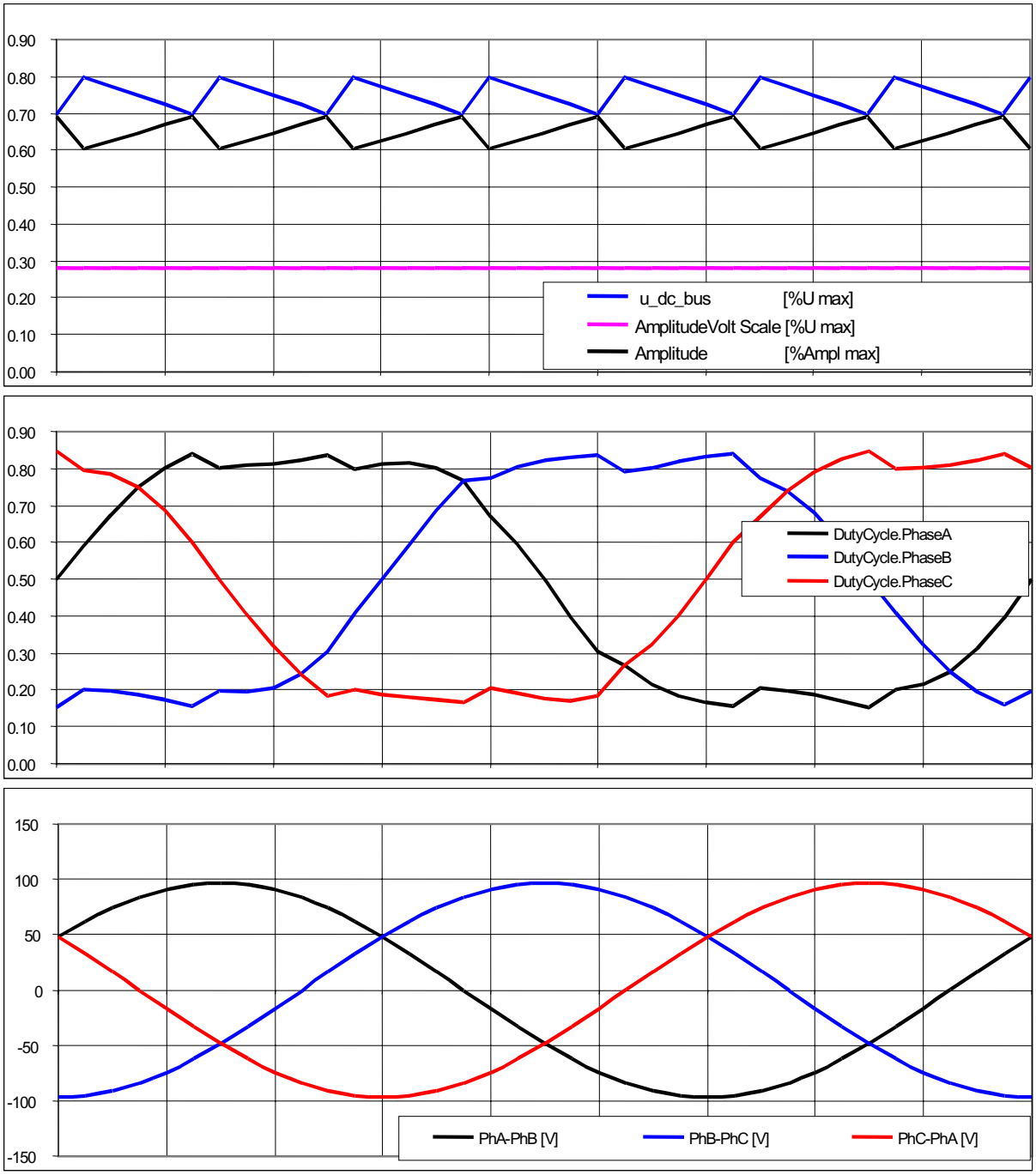


Figure 6-3. 3-Phase Waveforms with DCBus Voltage Ripple Elimination

6.1.6 PWM Generation

Process Description

This process generates a system of 3-phase sine waves with addition of third harmonic component shifted 120^{o} to each other using *mcgen3PhWaveSine3rdHIntp* function from the motor control function library.

The function is based on a fixed-wave table describing the first quadrant of sine wave stored in data memory of the controller. Due to symmetry of sine function, data in other quadrants are calculated using the data of first quadrant, which saves data memory. The sine wave generation for Phase A, simplicity, is explained in **Figure 6-4.** Phases B and C are shifted 120^{o} with respect to Phase A.

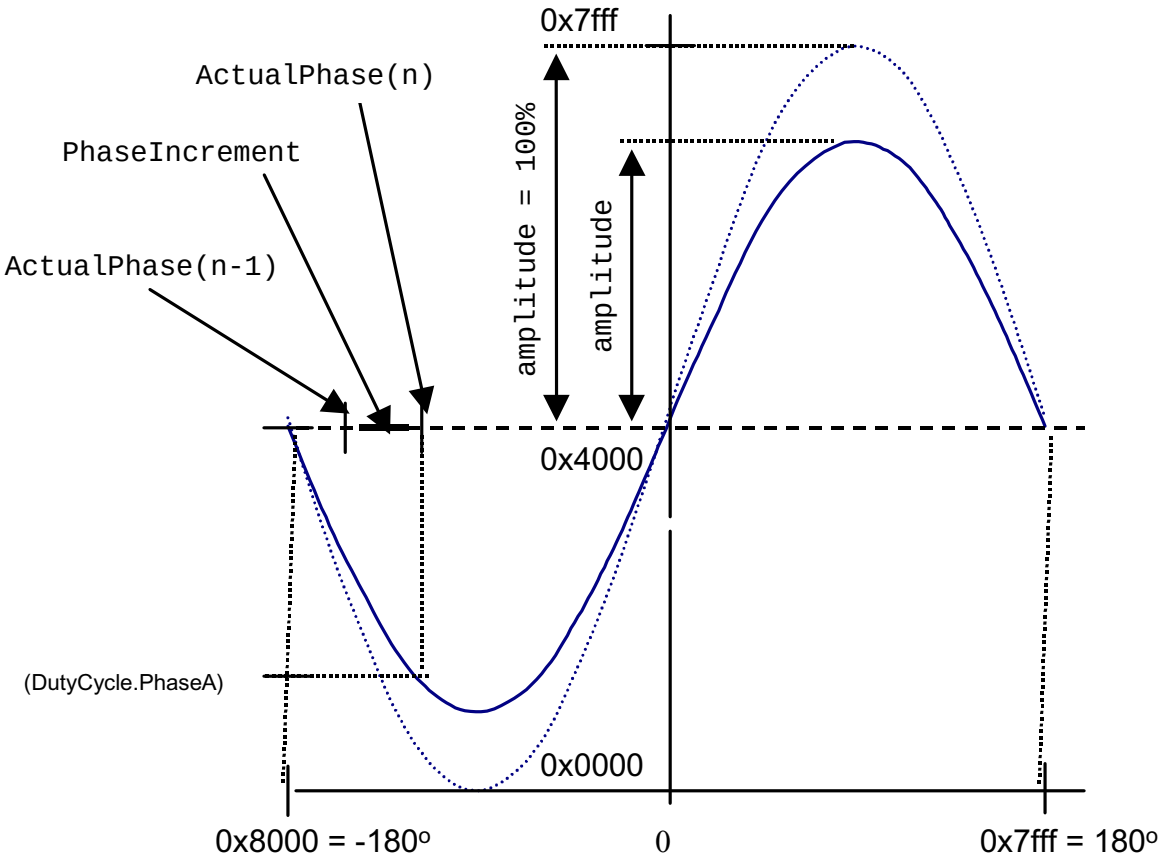


Figure 6-4. Sine Wave generation

Each time the waveform generation function is called, *ActualPhase* from previous step is updated by *PhaseIncrement*, and, according to the calculated phase, the value of sine is fetched from the sine table by the function *tfr16SinPIxLUT,* from the **Trigonometric Function Library** in Processor Expert. It's then multiplied by amplitude and passed to the PWM. An explanation of the 3-phase waveform generation with 3rd harmonic additionis found in the following formulas:

$$PWMA = \frac{1}{\sqrt{3}} \cdot Amplitude \cdot \left(\sin\alpha + \frac{1}{6} \cdot \sin 3\alpha \right) + 0.5$$

$$PWMB = \frac{1}{\sqrt{3}} \cdot Amplitude \cdot \left(\sin(\alpha - 120^0) + \frac{1}{6} \cdot \sin 3\alpha \right) + 0.5 \qquad \text{(EQ 6-2.)}$$

$$PWMC = \frac{1}{\sqrt{3}} \cdot Amplitude \cdot \left(\sin(\alpha - 240^0) + \frac{1}{6} \cdot \sin 3\alpha \right) + 0.5$$

Where PWMA, PWMB and PWMC are calculated, duty cycles passed to the PWM driver and amplitude determine the level of phase voltage amplitude.

The process performed in the PWM reload callback function, *pwm_Reload_A_ISR*, is accessed regularly at the rate given by the set PWM reload frequency. This process is repeated often enough to compare it to the wave frequency. Wave length comparisons are made to generate the correct wave shape. Therefore, for 16kHz PWM frequency, it is called each fourth PWM pulse; thus, the PWM registers are updated in a 4kHz rate (every 250µsec).

Figure 6-5 shows the duty cycles generated by the *mcgen3PhWaveSine3rdHIntp* function when *Amplitude* is 1 (100%).

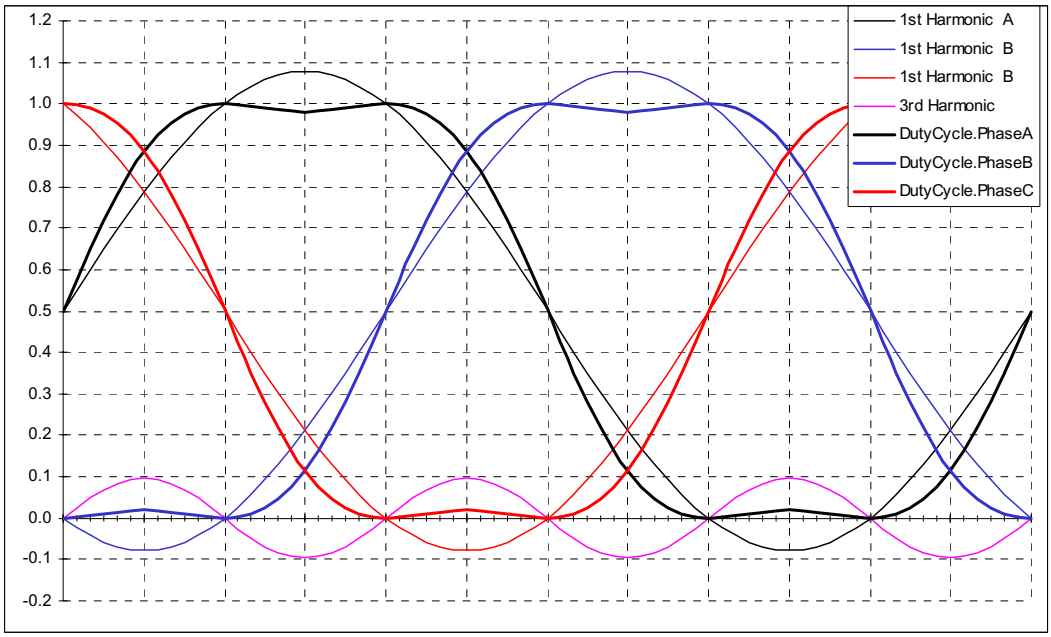


Figure 6-5. 3-Phase Sine Waves with 3rd Harmonic Injection, *Amplitude* = 100%

Figure 6-6 defines the duty cycles generated by the *mcgen3PhWaveSine3rdHIntp* function when *Amplitude* is 0.5 (50%).

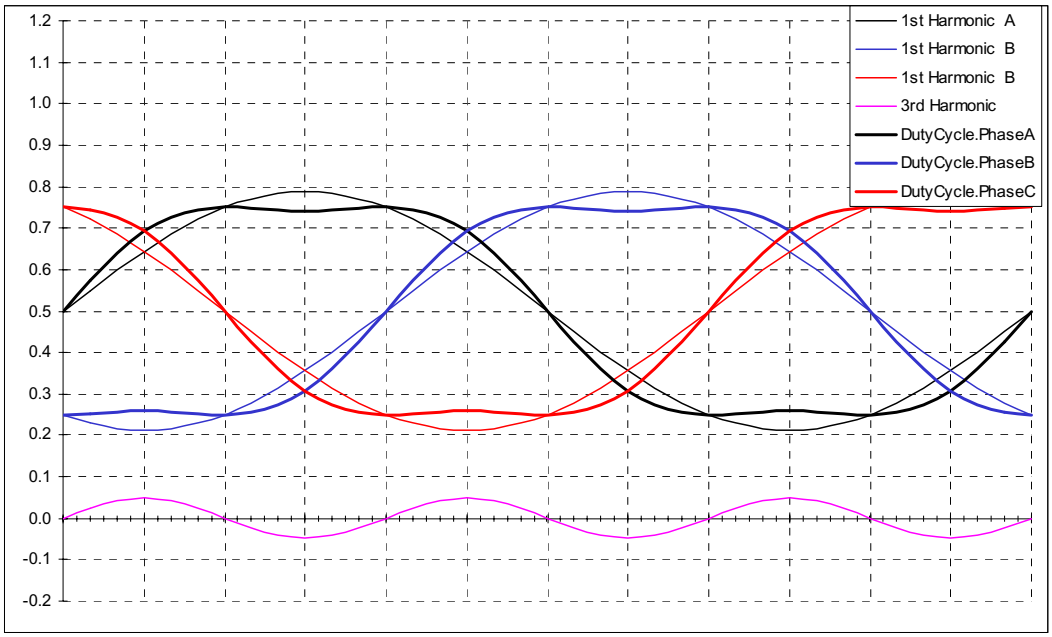


Figure 6-6. 3-Phase Sine Waves with 3rd Harmonic Injection, *Amplitude* = 50%

Input process:

- *Amplitude* is obtained from the DCBus ripple elimination process
- *Omega_required* is obtained from acceleration/deceleration ramp process

Output process:

Results calculated by the *mcgen3PhWaveSine3rdHIntp* function are passed directly to the PWM value registers using the PWM driver.

6.1.7 Fault Control

This process is responsible for fault handling. The software accommodates five fault inputs: overcurrent, overvoltage, undervoltage, overheating and wrong identified hardware.

Overcurrent: If overcurrent occurs in the DCBus link, the external hardware provides a rising edge on the controller's fault input pin, FAULTA1. This signal immediately disables all motor control PWM outputs (PWM1 - PWM6) and sets the *DC_Bus_OverCurrent* bit of *DriveFaultStatus* variable.

Overvoltage: If overvoltage occurs in the DCBus link, the external hardware provides a rising edge on the controller's fault input pin, FAULTA0. This signal immediately disables all motor control PWM outputs (PWM1 - PWM6) and sets the *DC_Bus_OverVoltage* bit of *DriveFaultStatus* variable.

Undervoltage: The DCBus voltage sensed by ADC is compared with the limit set in the software. If undervoltage occurs after a period defined by UNDERVOLTAGE_COUNT, all motor control PWM outputs are disabled, and the *DriveFaultStatus* variable is set to *DC_Bus_UnderVoltage.*

Overheating: The temperature of the power module sensed by ADC is compared with the limit set in the software. If overheating occurs after a period defined by OVERHEATING_COUNT, all motor control PWM outputs are disabled and the *DriveFaultStatus* variable is set to *OverHeating.*

Wrong Hardware: In the wrong hardware (for example, a different power module or missing optoisolation board) is identified during initialization, the *DriveFaultStatus* variable is set to *Wrong_Hardware.*

If any of these faults occur, the program run into infinite loop and waits for reset. The fault is signaled by user LEDs on the controller board and on the PC master software control screen.

7. Software implementation

This project is implemented using Processor Expert plug-in and Embedded Beans™ technology in the CodeWarrior Integrated Devolopment Environment (IDE). Processor Expert is designed for rapid application development of embedded applications on many platforms.

7.1 Embedded Beans

Embedded Beans are design components which encapsulate functionality of basic elements of embedded systems such as the controller's core; on-chip peripherals; stand-alone peripherals; virtual devices; and pure software algorithms. Embedded Beans allow access to these facilities via simple and uniform interface of properties, methods and events. Additional information can be found in Processor Expert help.

Table 7-1 lists the beans used in implementing the 56F805 application.

7.2 Bean Modules

Each peripheral on the hybrid controller chip or on the EVM board is accessible through a bean. Processor Expert generates the source code modules containing the implementation of methods controlling the hardware which provides the bean's functionality.

The following steps are required to generate the code:

- Add the beans to your project (Processor Expert tab of CodeWarrior's project panel)
- Set up the beans according to the hardware configuration
- Generate the source code
- Use beans' methods in your code (generated functions for bean's methods are named *beanName_MethodName*)

You should not modify generated modules; generated code can be found in the *Files* folder of CodeWarriors's project panel.

User modules, which are meant to be modified by the user, are found in the *User Modules* folder, where other user modules could also be added. The *Events* module (*events.c*) contains the handling routines for beans' events, those caused by interrupt, for example. The *Main* module (*projectName.c*) contains the function *main()* and all necessary declarations and variables.

All enabled methods are generated into appropriate bean modules during the code generation process. All Methods of each bean inserted into the project are visible as a subtree of the bean in the *Project* panel (Processor Expert tab of CodeWarrior's project panel). **Figure 7-1** and **Figure 7-2** show the list of beans inserted in the project for 56F805 and 56F8346 implementations. To see a detailed list of beans, see **Table 7-1** and **Table 7-2**.

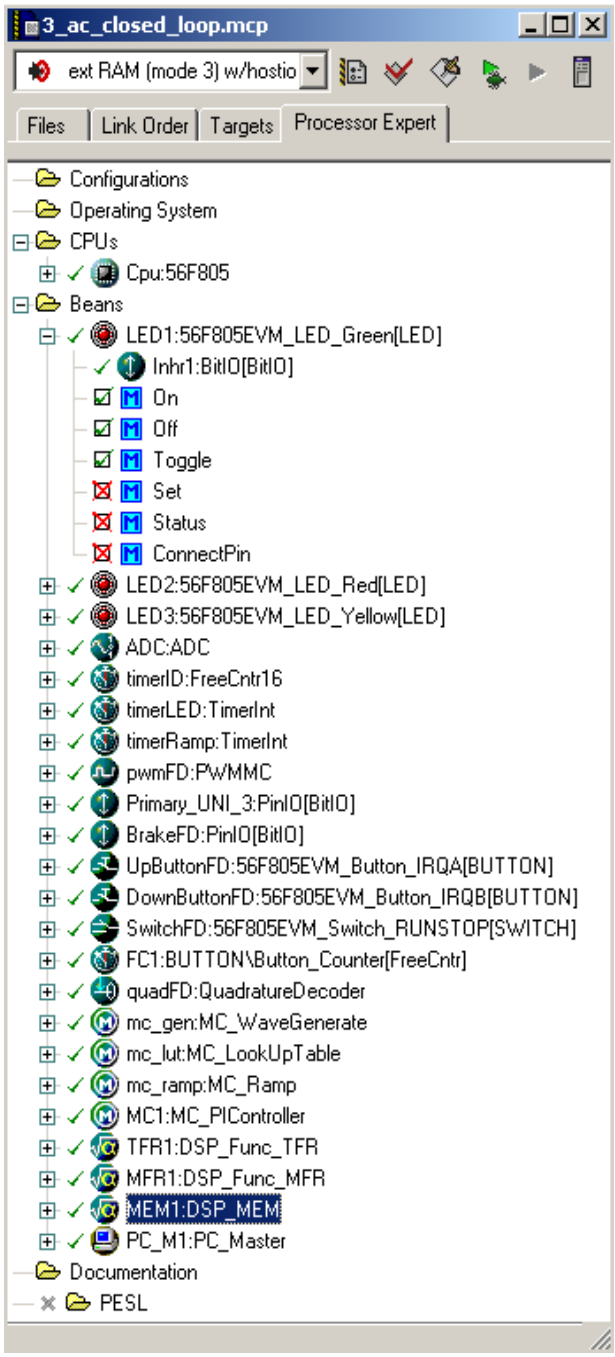


Figure 7-1. Beans Used in Implementing the 56F805

Table 7-1. Beans Used in Implementing the 56F805

Bean name	Bean type	Description
PC_Master	PC_Master	Provides serial communication with the PC master software running on a PC using the SCI0 on-chip device. Communication speed is 9600bps.
UpButtonFD, DownButtonFD	Button	Buttons are used for motor speed settings, with a 75rpm step. Buttons are connected to $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$ inputs. Events: • *OnButton* - Invoked by pressing the button
LED1, *LED2*, *LED3*	LED_Green, LED_Red, LED_Yellow	Beans control the LEDs on the EVM connected to pins GPIOB0, GPIOB1 and GPIOB2. Methods: • *On* - Switches the LED on • *On* - Switches the LED on • *Toggle* - Reverses the state of the LED
SwitchFD	Switch_RUNSTOP	Sets the application state to RUN/STOP. The bean uses the GPIOD5 input bit. Methods: • *GetVal* - Reads the switch state
Adc	ADC	Measures the DCBus voltage and temperature. The bean uses channels AD0 (voltage) and ADA5 (temperature) of the ADCA on-chip device. Methods: • *Measure* - Start of the measurement (length of the measurement is 1.7us) • *GetChanValue* - Reading of the measured value from given channel • *SetHighChanLimit* - Sets the upper limit value for a given channel • *SetLowChanLimit* - Sets lower limit value for a given channel Events: • *OnEnd* - End of conversion. This event watches the values of voltage and temperature. When the values overrun the allowed range, the motor is disconnected. • *OnHighLimit* - Switches brake on • *OnLowLimit* - Switches brake off
FC1	FreeCounter	This timer bean watches the 100ms minimum delay between presses of the Up and Down buttons. The bean uses the on-chip device TMRD23.

Table 7-1. Beans Used in Implementing the 56F805

Bean name	Bean type	Description
TimerLED	TimerInt	This timer controls the LED's blinking. It uses the on-chip timer TMRA0. Events: • *OnInterrupt* - Toggles the LED on and off
TimerID	FreeCntr16	The timer used to identify hardware. Methods: • *Enable* - Starts the timer • *Disable* - Disables the timer
TimerRamp	TimerInt	The timer controlling the acceleration/decceleration ramp. Methods: • *Enable* - Starts the timer Events: • *OnInterrupt* - Sets the speed as needed
PwmFD	PWMMC	Controls the state of the PWM module for motor applications. Frequency of the PWM output is 16kHz; dead time is 2.5µs. Pulse width is controlled by the application. Methods: • *Enable* - Enables PWM output • *Disable* - Disables PWM output • *SetDuty* - Sets duty on the appropriate PWM channel • *Load* - Updates control registers • *Swap* - Swaps PWM pairs • *OutputPadEnable* - Enables signal output onto CPU pins • *OutputPadDisable* - Disables signal output onto CPU pins Events: • *Onreload* - This event: — Measures actual speed (*MeasuredSpeed*) — Eliminates DCBus voltage ripples (*mcgenDCBVoltRippleElim* function) — Calculates waveform generator (*mcgen3PhWaveSine3rdHIntp* function) — Updates PWM value registers — StartsADC conversion • *OnFault0* - Switches the motor off and sets the Overvoltage error flag • *OnFault1* - Switches the motor off and sets the Overcurrent error flag

Table 7-1. Beans Used in Implementing the 56F805

Bean name	Bean type	Description
Primary_UNI_3	PinIO	Accesses the *Primary Serial COM* signal of *Primary UNI connector.* Uses the GPIOD7 input bit. Methods: • *GetVal* - reads the input value
BrakeFD	PinIO	.Accesses the signal for switching brake on/off. Methods: • *SetVal* - Switches brake on • *ClrVal* - Switches brake off
QuadFD	QuadratureDecoder	Computes the position and speed of the motor. Uses the Quad Decoder0 on-chip device. Methods: • *CoeficientCalc* - Computes parameters' settings • *GetScalePositionDifference* - Computes position and speed
mc_gen	MC_WaveGenerate	Algorithm used for determining the state of phases for motor control.
mc_lut	MC_LookUpTable	Look-up table algorithm. Methods: • *lutGetValue* - Gets the desired value from the table
MC_ramp	MR_Ramp	Sets the acceleration/decceleration ramp. Methods: • *rampGetValue* - Gets the ramp value
MC1	MC_PIController	Computes motor speed using the PI controller. Methods: • *controllerPItype1* - PI controller algorithm

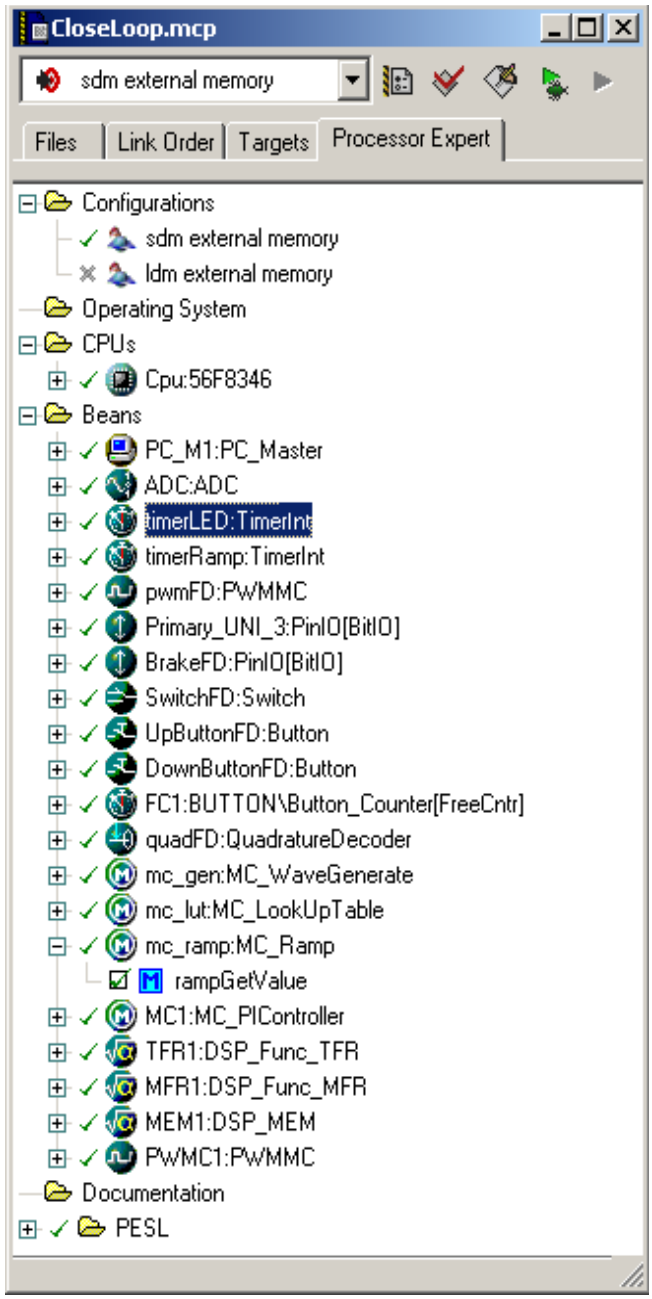


Figure 7-2. Beans Used in Implementing the 56F8346

Table 7-2. Beans Used in Implementation the 56F8346

Bean name	Bean type	Description
PC_Master	PC_Master	Provides serial communication with the PC master software running on the PC using the SCI0 on-chip device. Communication speed is 9600bps.
UpButtonFD, DownButtonFD	Button	Buttons are used for motor speed settings, with a 75rpm step. Buttons are connected to GPIOE4 and GPIOE7 inputs. Events: • *OnButton* - Invoked by pressing the button
SwitchFD	Switch_RUNSTOP	Sets the applcation state to RUN/STOP. The bean uses GPIOE5 input bit. Methods: • *GetVal* - Reads the switch state
Adc	ADC	Measures the DCBus voltage and temperature. The bean uses channels AD0 (voltage) and ADA5(temperature) of the ADCB on-chip device. Methods: • *Measure* - Start of the measurement (length of the measurement is 1.7us) • *GetChanValue* - Reading of the measured value from a given channel • *SetHighChanLimit* - Sets the upper limit value for a given channel • *SetLowChanLimit* - Sets the lower limit value for a given channel Events: • *OnEnd* - End of conversion. This event watches the values of voltage and temperature. When the values overrun the allowed range, the motor is disconnected. • *OnHighLimit* - Switches brake on • *OnLowLimit* - Switches brake off
FC1	FreeCounter	This timer bean watches the 100ms minimum delay between presses of the Up and Down buttons. The bean uses the on-chip device TMRD23.
TimerLED	TimerInt	This timer controls the LED's blinking. It uses the on-chip timer TMRB0. Events: • *OnInterrupt* - Toggles the LED on and off

Table 7-2. Beans Used in Implementation the 56F8346

Bean name	Bean type	Description
TimerID	FreeCntr16	The timer used to identify hardware. Uses the TMRA0 on-chip device. Methods: • *Enable* - Starts the timer • *Disable* - Disables the timer
TimerRamp	TimerInt	The timer controlling the acceleration/decceleration ramp. Uses the TMRA1 on-chip device. Methods: • *Enable* - Starts the timer Events: • *OnInterrupt* - Sets the speed as needed
PwmFD	PWMMC	Controls the state of PWM module for motor-application. Frequency of the PWM output is 16kHz, Dead time is 2.5us. Pulse width is controlled by the application. Uses the PWM_B on-chip device. Methods: • Enable - Enables PWM output • *Disable* - Disables PWM output • *SetDuty* - *S*ets duty on the appropriate PWM channel • *Load* - Updates control registers • *Swap* - Swaps PWM pairs • *OutputPadEnable* - Enables signal output onto CPU pins • *OutputPadDisable* - Disables signal output onto CPU pins Events: • *Onreload* - This event: — Measures actual speed (*MeasuredSpeed*) — Eliminates DCBus voltage ripples (*mcgenDCBVoltRippleElim* function) — Calculate waveform generator (*mcgen3PhWaveSine3rdHIntp* function) — Updates PWM value registers — Starts ADC conversion • *OnFault0* - Switches the motor off and sets the Overvoltage error flag • *OnFault1* - Switches the motor off and sets the Overcurrent error flag

Table 7-2. Beans Used in Implementation the 56F8346

Bean name	Bean type	Description
Primary_UNI_3	PinIO	Accesses the *Primary Serial COM* signal of the *Primary UNI connector.* Uses the GPIOE6 input bit. Methods: • *GetVal* - Reads the input value
BrakeFD	PinIO	Accesses the signal for switching the break on and off on GPIOD7. Methods: • *SetVal* - Switches brake on • *ClrVal* - Switches brake off
QuadFD	QuadratureDecoder	Computes the position and speed of the motor. Uses the Quad Decoder1 on-chip device. Methods: • *CoeficientCalc* - Computes parameters' settings • *GetScalePositionDifference* - computes position and speed
mc_gen	MC_WaveGenerate	Algorithm used for determining the state of phases for motor control.
mc_lut	MC_LookUpTable	Look-up table algorithm. Methods: • *lutGetValue* - Gets the desired value from the table
MC_ramp	MR_Ramp	Sets the acceleration/decceleration ramp. Methods: • *rampGetValue* - Gets the ramp value
MC1	MC_PIController	Computes motor speed using the PI controller. Methods: • *controllerPItype1* - PI controller algorithm
PWMMC1	PWMMC	Sets the LED's state on the EVM. The bean uses LEDs connected to PWM outputs. Methods: • *Mask* - Switches LEDs on/off

7.2.1 Initialization

The *Main* routine calls the *PE_low_level_init()* function generated by Processor Expert. This function pre-sets the CPU internal peripherals to the initial state according to the beans' settings.

The main method provides the following application initialization:

- Sets the upper and lower limits of the A/D converter
- Starts A/D conversion
- Sets up aplication state
- Identifies connected hardware
- Establishes Quadrature Encoder settings for speed computation
- Sets up PI controller

The board identification routine identifies the connected power stage board by decoding the identified message sent from the power stage. If the wrong power stage is identified, the program goes to the infinite loop, displaying the fault status on the LED. The state can be left only by a reset.

7.3 State Diagram

The general state diagram incorporates the main routine entered from reset and interrupt states. The main routine includes the initialization of the CPU and the main loop. The main loop incorporates the initialization state, the application state machine and checks the state of the Run/Stop switch.

The interrupt states calculates the actual speed of the motor, PWM reload event, ADC service, Limit analog values handling, overcurrent and overvoltage PWM fault handler, and other tasks.

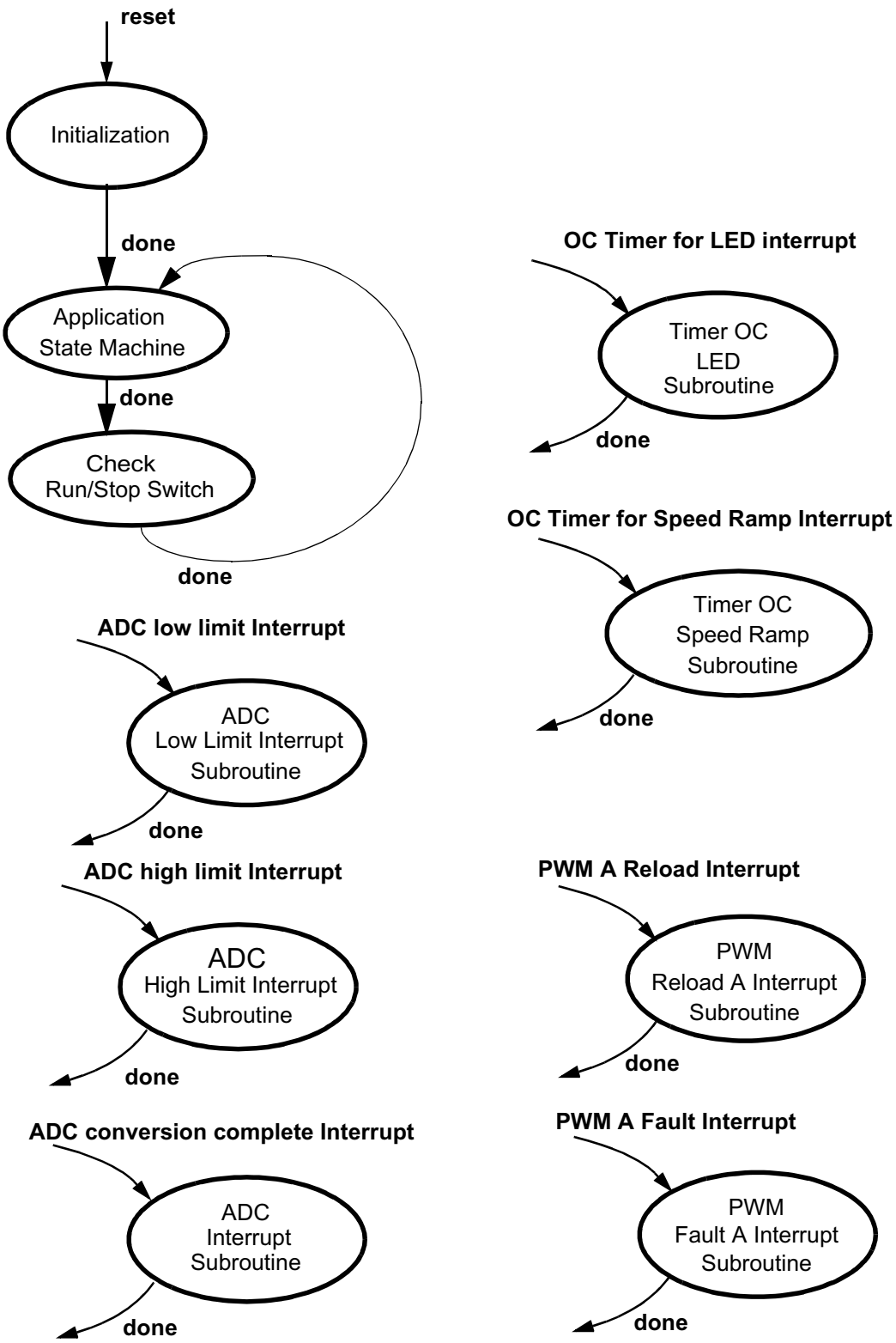


Figure 7-3. State Diagram - General Overview

7.3.1 Application State Machine

This state controls the main application functions, depicted in **Figure 7-4.**

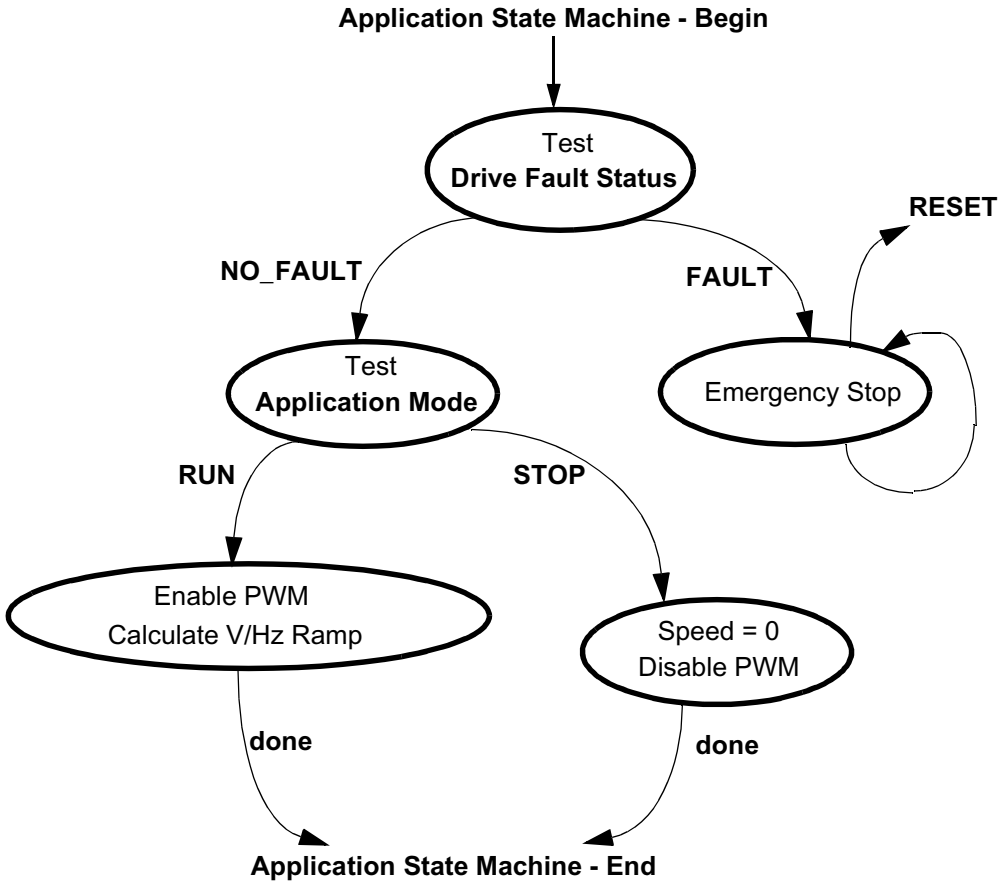


Figure 7-4. State - Application State Machine

7.3.2 Check Run/Stop Switch

In this state, the Run/Stop switch is checked according to the Application Mode setting and is set to RUN or STOP as directed.

Individual driver functions are provided by special PE function calls, found under the Processor Expert tab.

8. PC Master Software

PC master software was designed to provide a debugging, diagnostic and demonstration tool for the development of algorithms and applications. It runs on a PC connected to the EVM via an RS-232 serial cable. A small program resident in the controller communicates with the PC master software to parse commands, return status information to the PC and process control information from the PC. PC master software uses part of Microsoft's Internet Explorer as the user interface. To enable PC master software operation on the hybrid controller target board application, add the PC master software bean to the project and configure it. This automatically includes the SCI driver and installs all necessary

services. The SCI communication's default baud rate is 9600bd. It is set automatically by the PC master software driver and can be changed if needed. A detailed description of PC master software is provided in a dedicated User's Manual.

The 3-phase AC Motor V/Hz Speed Open Loop application utilizes PC master software for remote control from the PC. It enables the user to:

- Set the motor speed
- Set the close loop

PC master software reads and displays these variables to the user:

- Required and actual motor speed
- Application operational mode
- Start/stop status
- DCBus voltage
- Temperature
- Phase voltage amplitude

The PC master software Control Page is illustrated in **Figure 8-1**. Profiles of the required and actual speeds can be seen in the Speed Scope window.

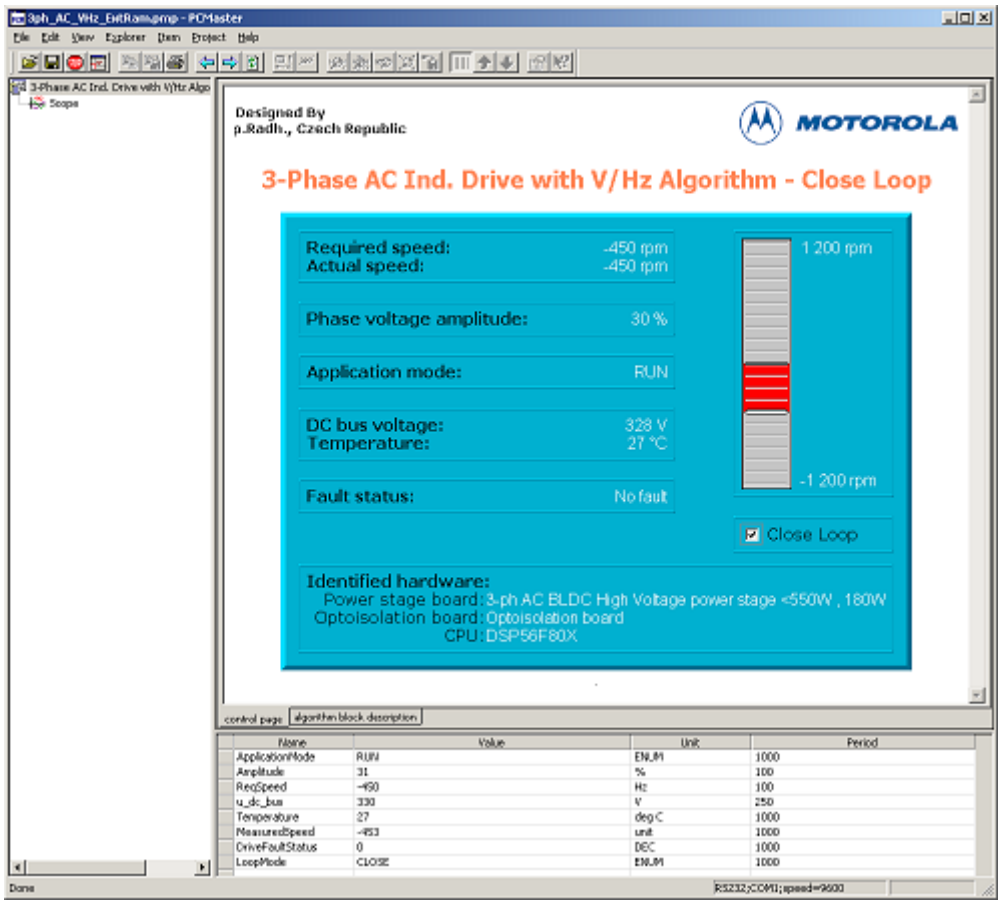


Figure 8-1. PC Control Window

9. References

DSP56F800 Family Manual, DSP56F800FM/D

DSP56F80x User's Manual, DSP56F801-7UM/D

DSP56F805 Evaluation Module Hardware User's Manual, DSP56F805EVM

DSP56800E Reference Manual, DSP56800ERM/D

MC56F8300 Peripheral User Manual, MC56F8300UM/D

MC56F8346 Evaluation Module Hardware User's Manual, MC56F8346EVMUM

3-Phase AC Induction Motor Control V/Hz Application - Closed Loop, 805ACIMTD/D, Motorola

3-Phase AC Induction Motor Control V/Hz Application - Closed Loop, 8346ACIMTD/D, Motorola

"Low Cost 3-phase AC Motor Control System Based On MC68HC908MR24." Motorola Semiconductor Application Note, AN1664, 1998.

Processor Expert Embedded Beans, Processor Expert Help

For more information, go to URL:
http://www.motorola.com/semiconductors

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://motorola.com/semiconductors

AN1958/D

Freescale Semiconductor, Inc.