
An Introduction to AC Induction Motor Control Using the dsPIC30F MCU

*Author: Steve Bowling
Microchip Technology*

INTRODUCTION

This application note demonstrates how you use the dsPIC30F MCU to control an AC Induction Motor (ACIM). The discussion is based on the dsPICDEM™ MC Motor Control Development System, but you can use your own hardware if you choose. The dsPICDEM MC Motor Control Development System has electrical isolation and is fully self-protected against faults. With these features, you can safely develop a motor control application and also avoid damage to hardware by software errors.

The code provided in this application note is a simple example that provides basic variable speed control of an ACIM. It will help you learn about the dsPIC30F architecture and the basics of ACIM control.

Recommended Hardware

The code presented in this application note can be run on the following Microchip equipment:

- MPLAB® ICD 2 In-Circuit Debugger and Device Programmer (Part# DV164005)
- dsPICDEM MC1 Motor Control Development Board (Part# DM300020)
- dsPICDEM MC1H 3-Phase High-Voltage Power Module (Part# DM300021)
- 3-Phase ACIM High-Voltage Motor (208/460V) (Part# AC300021)

If you prefer, you can supply your own 3-phase or single-phase ACIM. The recommended power range is 1/6-1/2 HP.

Documents for Further Reading

These application notes provide useful background information:

- AN887, "AC Induction Motor Fundamentals" (DS00887)
- AN889, "VF Control of 3-Phase Induction Motors Using PIC16F7X7 Microcontrollers" (DS00889)
- AN900, "Controlling 3-Phase AC Induction Motors Using the PIC18F4431" (DS00900)
- AN908, "Using the dsPIC30F for Vector Control of an ACIM" (DS00908)

BACKGROUND

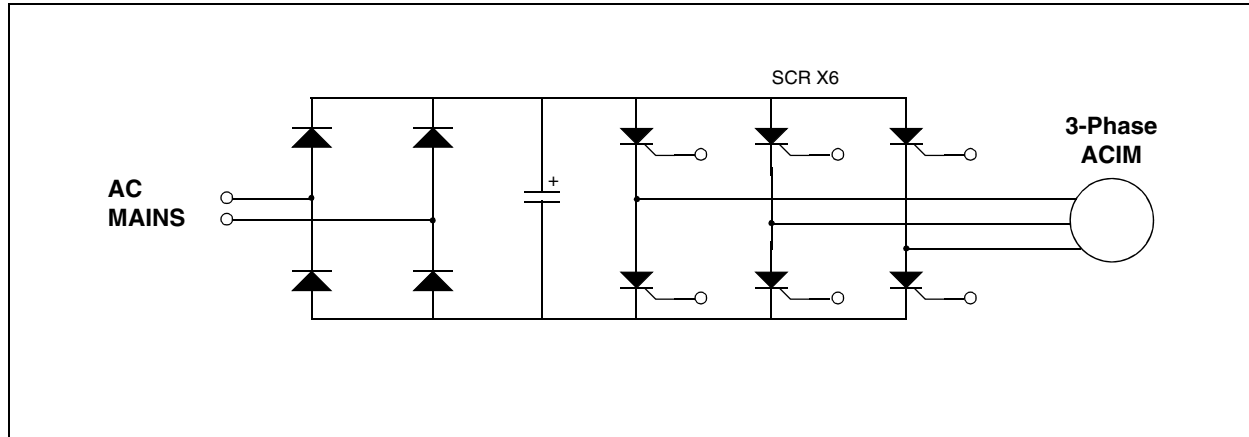
Variable speed ACIM drives have evolved from industrial control applications. Wound DC motors were preferred in the past because they were easier to control. The motor current is simply varied to adjust the torque output and motor speed. However, these DC motors do have some disadvantages. DC motors used in industrial applications need to be periodically overhauled to replace worn brushes and rotor windings. Since the rotor windings of a DC motor are inside the motor, it is more difficult to keep them cool.

An ACIM has a simple rotor construction and does not use brushes. For these reasons, an ACIM is more durable than a DC motor. The only mechanical components that need to be serviced are the rotor bearings. The rotor is much more tolerant of heat because it is simply a steel cage with magnetic laminations. The durability of the ACIM makes it an attractive choice.

Variable speed control of an ACIM is conceptually very simple. The frequency and amplitude of the drive voltage must be varied to change the motor speed. Early ACIM drives used SCR devices connected to the motor as shown in Figure 1. By firing each SCR at the appropriate time, a very crude sinusoidal voltage can be generated on the motor phases. These types of circuits are often called 'six-step' drives because there are six different ways that the SCR devices can be energized to produce motor currents. However, the high harmonic content of the six-step drive causes high heat dissipation and does not deliver good performance at low frequencies.

Semiconductor technology has drastically improved since the days of six-step drives. SCR devices are now replaced with MOSFET or IGBT devices that can be switched at relatively high frequencies with minimal power loss. These devices can be controlled using PWM signals to generate continuously variable drive voltages and currents.

FIGURE 1: SCR INVERTER CIRCUIT



SINGLE-PHASE ACIM vs. THREE-PHASE

Most motors intended for industrial applications will have three-phase windings. Three-phase power is widely available in an industrial environment. Residential environments usually only have single-phase power available, which presents a problem for ACIMs.

A three-phase motor is the best type to use for variable speed control. The three-phase motor gives good torque performance at all operating speeds. Single-phase motors can also be used, but they have limited performance in the low-speed range. Depending on the motor, there can be significant torque pulsations when a single-phase induction motor is run at low speeds.

A three-phase motor can generate a true rotating magnetic field in the stator windings when fed from a source of three-phase power. However, without some modification, an ACIM with a single stator winding cannot produce a rotating magnetic field that is capable of producing torque. This rotating field problem can be handled in different ways.

Shaded-pole motors have a pole structure that is formed from laminated iron pieces. A single coil is placed on the structure. A 'rotating' field is produced by placing shorting rings around the laminated pole pieces in two strategic locations. The shorting rings place offsets in the magnetic flux so that a rotating field can be created.

Another way to solve the rotating field problem is to use two electrical windings that are physically offset in the stator. This type of motor is called a split-phase ACIM. In most cases, one of the two windings has lower electrical impedance and is designated as the primary or 'run' winding. The second winding has higher impedance and is designated as the secondary, or 'start' winding.

In general, there are three types of split-phase motors. The first type has two windings, a centrifugal switch and a single pair of input terminals. These motors are typically used for fans and blowers. At start-up, both windings are connected in parallel. After the motor is near full speed, the centrifugal switch disconnects the start winding. Once the motor is spinning with sufficient speed, the motor can run without the start winding. This type of motor is not very efficient during starting, but a fan or blower will not present a large load at low speed.

The second kind of split-phase motor has two windings. A centrifugal switch and a capacitor are placed in series with the start winding. The capacitor provides phase shift, which improves the starting torque and reduces the starting current. The centrifugal switch disconnects the start winding (and capacitor) after the motor is near full speed. This type of motor is often called a 'capacitor-start' motor.

The third type of split-phase motor eliminates the centrifugal switch, but still has a capacitor in series with the secondary winding. The secondary winding is, therefore, never disconnected. This type of split-phase motor is often called a 'capacitor-run' motor and has the best performance of all split-phase motor types. The capacitor-run motor will have the best torque performance over the operating speed range. Of all the different types of single phase ACIMs, a capacitor-run motor makes the best choice for variable speed control.

THE INVERTER CIRCUIT

A variable speed ACIM application needs an inverter circuit that performs two functions. First, the incoming AC mains supply is rectified and filtered to produce a DC bus voltage. Second, the DC bus voltage must be transformed back into AC currents that will power the motor. The circuit in Figure 2 can be used to control a three-phase motor.

You may notice that I said ‘AC current’ and not ‘AC voltage’. We will use the motor control PWM available on the dsPIC[®] MCU to control the power transistors in the inverter circuit. If you were to put a scope probe on one of the three phase connections, you would just see a PWM signal with an amplitude approximately equal to the DC bus voltage. Since the motor windings are inductive, the incoming voltage will be integrated to produce a motor current that is proportional to the PWM duty cycle. If the PWM duty cycle is modulated, then AC currents of any arbitrary wave shape can be generated.

Several different types of inverter circuits could be used depending on the type of ACIM that you want to control. No matter what type of inverter circuit is used, you can think of each complementary pair of transistors across the DC bus as a single entity, controlled by a single PWM generator, as shown in Figure 3. Each complementary pair of transistors is connected to a motor phase winding.

The PWM is used like a digital-to-analog converter in order to produce motor currents of any desired wave-shape. A 50% PWM duty cycle is normally used as a zero-current reference point. If all PWM duty cycles are at 50%, then all phases will have the same applied average voltage and there will be zero average current flowing in the motor. If the PWM duty cycle is raised above 50%, then positive current will be generated in the winding. If the duty cycle is below 50%, then negative current will be generated in the winding.

FIGURE 2: THREE-PHASE INVERTER CIRCUIT

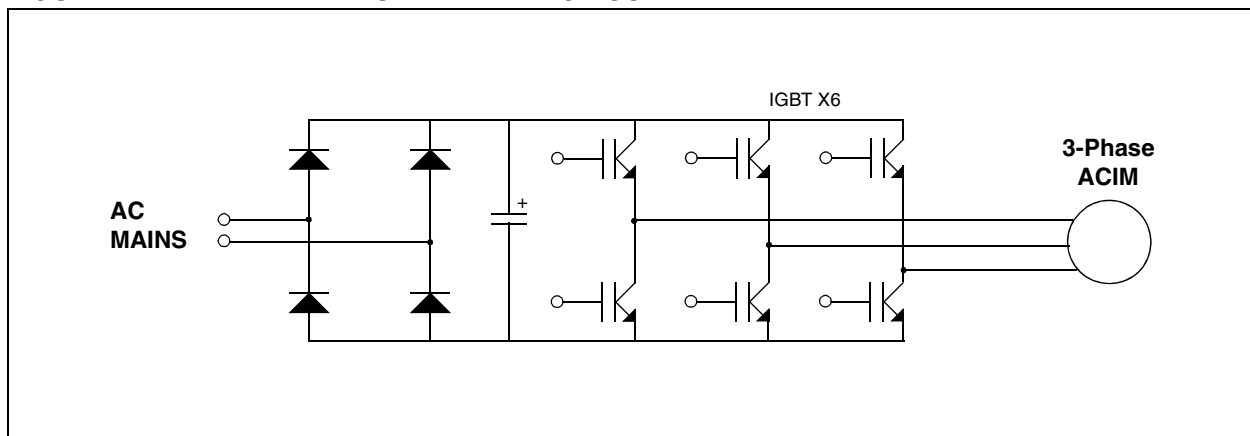
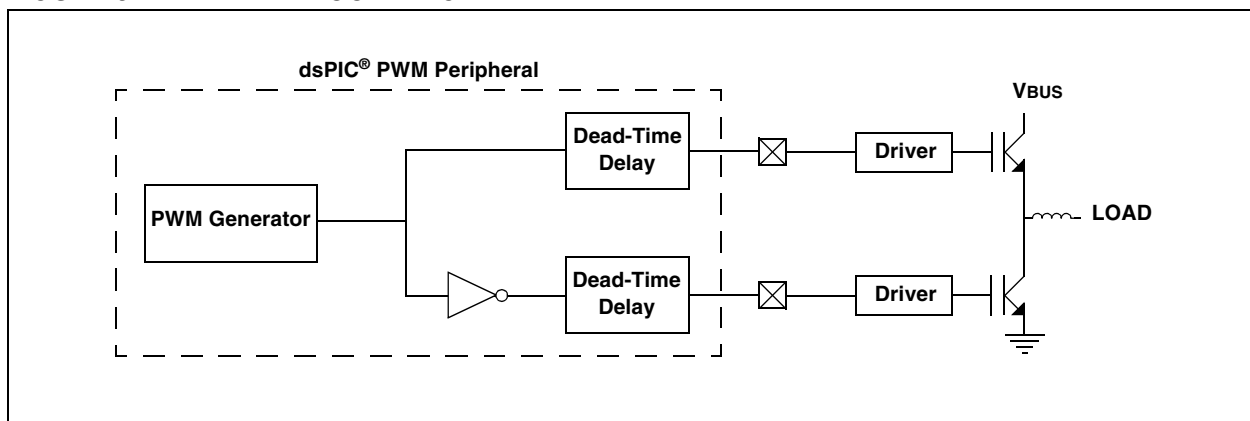


FIGURE 3: PWM BLOCK DIAGRAM



AN984

Inverter Circuit Options for Single-Phase Motors

If you wish to control a split-phase motor with a transistor inverter circuit, there are several different inverter topologies that you can use. If you want to eliminate the capacitor and implement the phase shift in software, the three-phase inverter circuit described above can be used, as shown in Figure 4. With an inverter driven secondary winding, a current with any arbitrary phase shift and amplitude can be generated on the secondary winding.

The split-phase motor could also be driven with an H-bridge inverter, as shown in Figure 5. This topology requires a run capacitor, but eliminates two of the inverter switches. The disadvantage of this circuit is that the motor direction is set by the location of the capacitor in the circuit. The H-bridge inverter circuit could also be used to drive a shaded-pole motor with a single winding.

Another way to drive a split-phase motor or a shaded-pole motor uses a voltage doubler circuit with an H-bridge inverter, as shown in Figure 6. In this circuit, the switching devices will see a higher DC bus voltage.

FIGURE 4: SPLIT-PHASE MOTOR DRIVE WITH 3-PHASE INVERTER

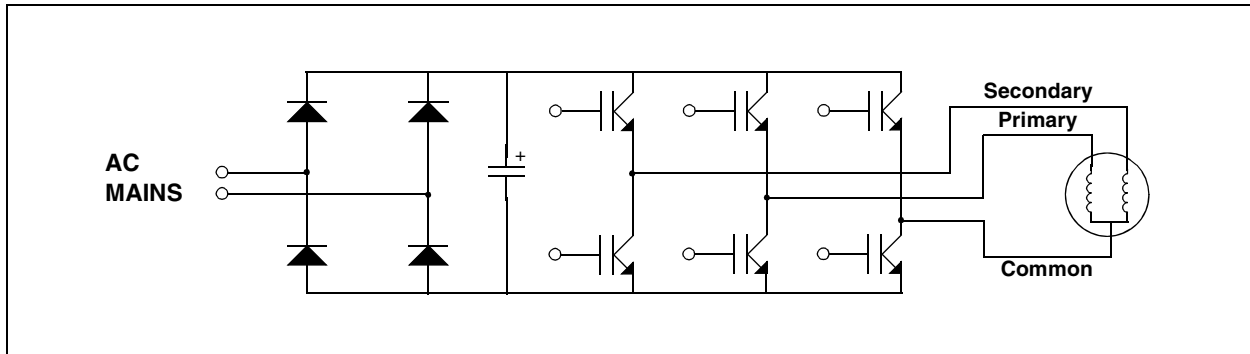


FIGURE 5: SPLIT-PHASE MOTOR DRIVE WITH H-BRIDGE INVERTER

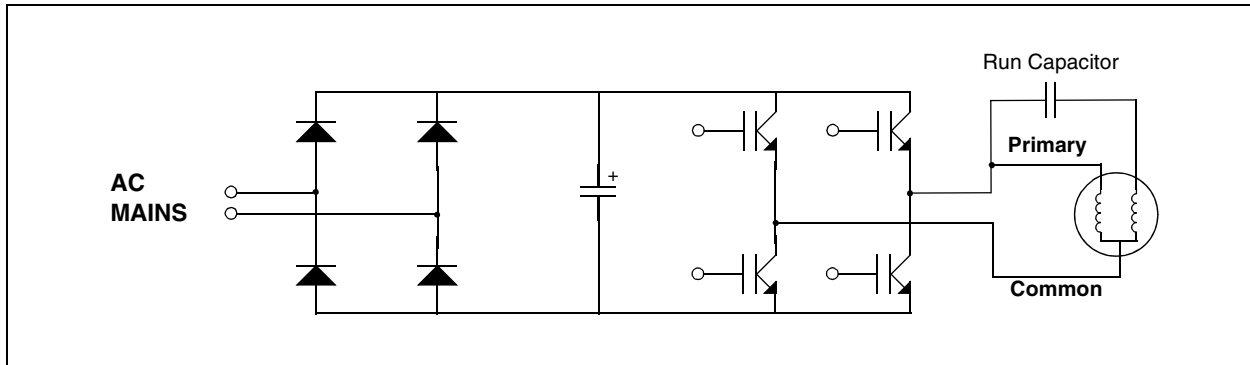
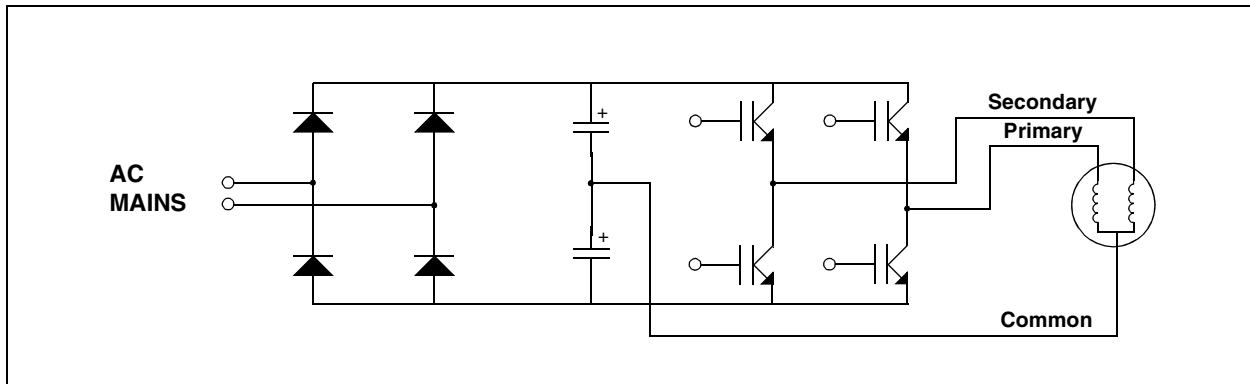


FIGURE 6: SPLIT-PHASE MOTOR DRIVE WITH VOLTAGE DOUBLER CIRCUIT



GENERATING A SINUSOIDAL WAVEFORM

The easiest way to generate a sinusoidal waveform is to use a look-up table. You could also calculate the sine value on the fly, but it's just not worth spending the CPU time to do this. A look-up table is used that contains all the points of a sine wave. The sine values are read from the table at periodic intervals, scaled to match the allowable range of duty cycles, and then written to the duty cycle registers.

A sine pointer variable is maintained in software that defines the present location in the table. This pointer has to be adjusted at periodic intervals, usually at the beginning of each PWM period. If a constant adjustment value is added to the pointer at each interval, the software will move through the table at a fixed frequency. The look-up table length is usually set to an even power of 2, such as 64, 128 or 256. This way, the software does not have to check the pointer value every time it is adjusted. The pointer can simply be allowed to roll over and reset to 0.

How Many Points in the Table?

This is a question often asked when creating a look-up table. There is not a specific answer to this question. Too few points in the table will cause a 'staircase' effect in the motor current waveform. The staircase effect will cause excessive motor current distortion, which causes higher heat dissipation. Too many points will use up valuable memory in the MCU. A good 'rule of thumb' is to divide the maximum desired modulation frequency into the PWM carrier frequency. A PWM carrier frequency, just outside the audible range, is usually chosen.

Assume that a 16-kHz PWM carrier is selected and the maximum modulation frequency is 60 Hz for a typical ACIM:

EQUATION 1:

$$\begin{aligned} \text{Number of Table Values} &= f_{\text{PWM}}/f_{\text{MODMAX}} \\ &= 16,000/60 \\ &= 267 \end{aligned}$$

For this example, a 256 value sine table would be sufficient. In practice, the code provided in this application note has 64 entry tables and provides good results.

Sine Table Pointer

Once the sine table size has been selected, the size of the sine table pointer variable can be chosen. Let's assume that a 256 value table is used for the sine values. At first glance, you might assume that an 8-bit pointer value would be sufficient. However, you will want the pointer value to be larger so that it will be easy to produce very low modulation frequencies.

For the code examples in this application note, a 16-bit sine table pointer is used. The pointer represents a full 360 degrees of angle, where 0x0000 = 0 degrees and 0xFFFF = 359.9 degrees. Each time a new value is needed from the look-up table, the upper 8 bits of the pointer variable are used as the pointer index. The lower 8 bits of the pointer variable can be viewed as fractional bits.

You will want to know the resolution of the modulation frequency. To find this resolution, you need to know how frequently the sine table pointer will be adjusted. For now, assume that this adjustment occurs every PWM period. Assuming a 16-kHz PWM frequency, the modulation frequency resolution would be:

EQUATION 2:

$$\begin{aligned} \text{Mod Frequency Resolution} &= f_{\text{PWM}}/2^{16} \\ &= 0.244 \text{ Hz/bit} \end{aligned}$$

So, this choice of angle resolution and PWM carrier frequency allows the modulation frequency to be adjusted in 0.244 Hz steps. Our variable speed application runs the motor at full speed with a modulation frequency of 60 Hz. To find the table pointer delta value that will provide a 60 Hz modulation frequency, use this formula:

EQUATION 3:

$$\begin{aligned} f_{\text{MOD}}/0.244 &= 60/0.244 \\ &= 246 \text{ bits} \end{aligned}$$

If the value, 246, is added to the sine table pointer at each PWM interrupt, then we will get a 60-Hz modulation frequency.

How to Create Multiple Sine Outputs with Phase Offsets

We will want to create multiple outputs with different phases to drive the ACIM. A phase offset can be established for a particular output by adding a constant offset value to the pointer.

THREE-PHASE OUTPUTS

Binary numbering works well for three-phase systems. Assuming a 16-bit pointer size is used, a value of 0x5555 provides a 120-degree offset and a value of 0xAAAA gives a 240-degree offset. The offset values are added to the sine table pointer at each PWM interrupt to provide two additional pointers for the 2nd and 3rd phase. Since 16-bit arithmetic is used, the sine table pointer just wraps around if an overflow occurs as a result of the offset addition.

SINGLE-PHASE OUTPUTS

If you want to drive a single motor winding using an H-bridge inverter configuration, then you can modulate one side of the bridge with a 0-degree phase offset and the other side with a 180-degree offset. An offset value of 0x8000 gives the 180-degree offset using a 16-bit sine table pointer. An offset of 90 degrees may be required to drive the auxiliary winding of a split-phase motor. In this case, just use a sine pointer offset value of 0x4000.

Scaling the Sine Table Pointer

To obtain index values to perform table look-ups from the sine data, the 16-bit sine pointers are right shifted to discard the 'fractional' bits described in the "**Sine Table Pointer**" section. If a 256-entry table is used, then you will only need the upper 8 bits to use as a look-up table index.

Scaling the Sine Table Look-up Value

Once look-up values are obtained from the table, they are multiplied by scaling values to determine the actual amplitude of the modulation output. More scaling details are provided in the "**PWM Modulation**" section.

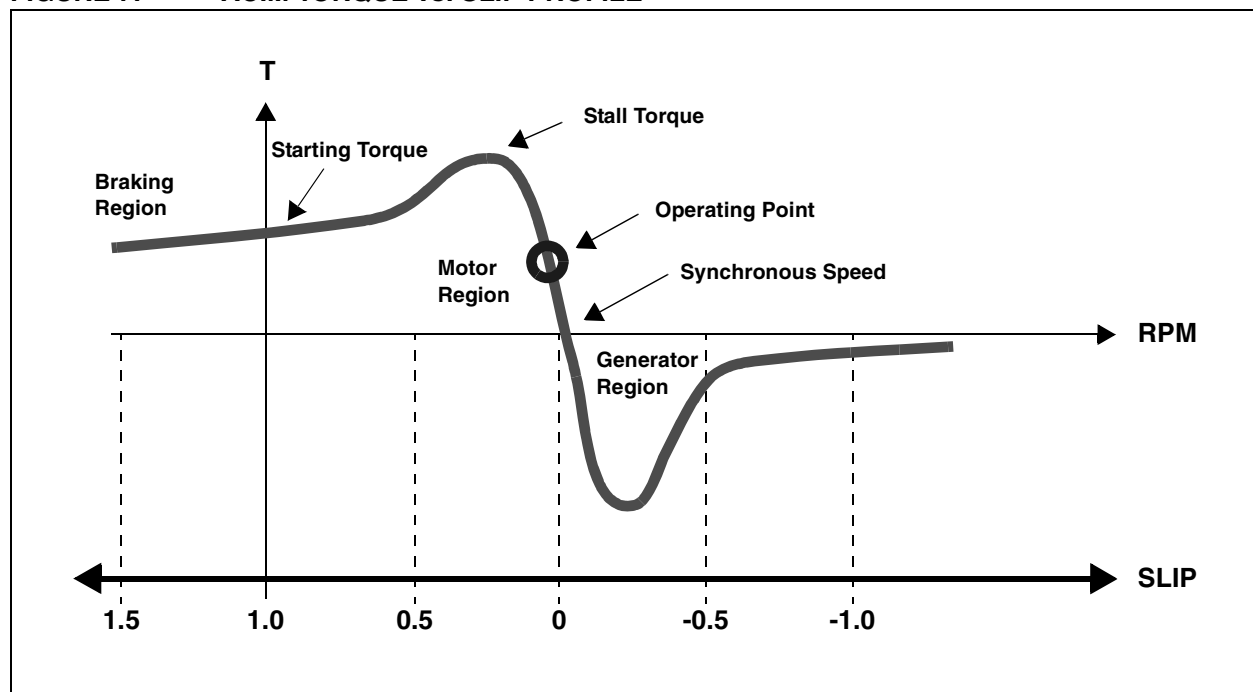
ACIM SLIP vs. TORQUE

An ACIM must have slip in order to operate. Slip is the difference in mechanical rotor speed compared to the speed of the rotating magnetic field generated by the stator. Slip can be expressed either as a ratio or as a frequency difference.

To demonstrate slip, assume a two-pole ACIM is energized with a 60-Hz AC input voltage. If the rotor turned synchronously with the applied input frequency, the rotor would spin at 60 revolutions per second, or 3600 RPM. Due to the slip, however, the rotor turns slightly slower than the synchronous speed of 3600 RPM. The slip is usually accounted for in the motor specifications. A two-pole motor with a synchronous speed of 3600 RPM might specify 3450 RPM as the nominal motor speed.

Figure 7 below shows how the motor torque and slip relate to each other. When energy is first applied to the motor, the rotor will be at rest. At this instant, slip will be at maximum value. The motor will have a finite start-up torque at this time. If the load on the motor does not exceed the available start-up torque, the motor begins to accelerate to a speed near the synchronous speed. The motor will then be operating at a point on the right side of the torque profile. When operating in this region, the motor has the ability to self-regulate its own speed within a limited range. When additional load is placed on the motor shaft, the motor speed naturally decreases. This decrease in speed causes the slip to increase, which increases the torque. As greater load is placed on the motor, the operating point moves to the left on the torque profile until the stall torque is reached. At this point, the rotor stalls.

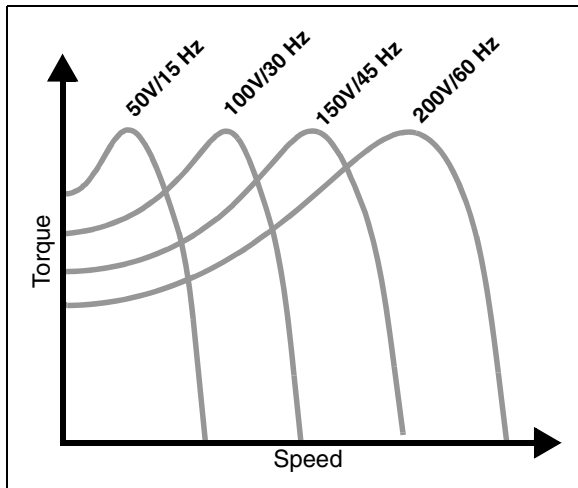
FIGURE 7: ACIM TORQUE vs. SLIP PROFILE



THE VOLTS-HERTZ PROFILE

ACIMs are usually designed with a fixed operating voltage and frequency in mind. The torque profile that was discussed in the prior section is applicable to one frequency. However, the self-regulation ability of the ACIM can be used to implement basic variable speed control. If the input frequency to the motor is changed, the synchronous speed of the motor also changes accordingly. The frequency change has the effect of moving the torque profile curve to the left or right. If the motor input frequency is continuously adjustable, a family of torque profile curves will be created, as shown in Figure 8. Since the motor is an inductive load, the reactance of the motor will decrease with a decrease in drive frequency. Therefore, if an adjustment in drive frequency is made, then the voltage should also be adjusted by a proportional amount. The relationship between frequency and voltage for a variable speed ACIM is called the Volts-Hertz, or VF profile.

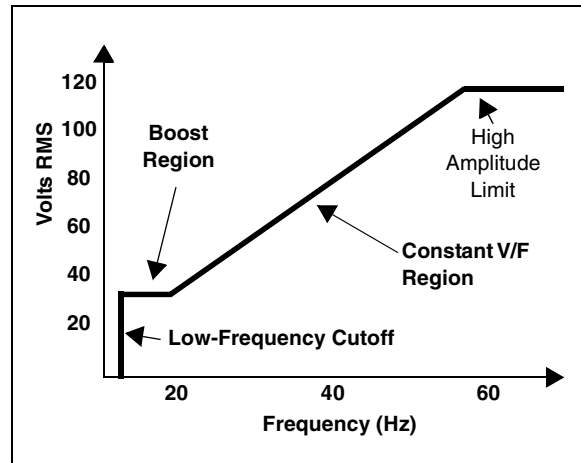
FIGURE 8: TORQUE PROFILES FOR VARIABLE FREQUENCY OPERATION



A linear relationship between voltage and frequency is most often used. The VF relationship required for a particular motor can be determined from the motor nameplate parameters. For example, assume that you have a motor designed to be operated at 230 VAC, 60 Hz. If you want to drive the motor at half the rated speed, then you should use half the input frequency AND half the input voltage. In this example, 115 VAC and 30 Hz would be approximate input values.

In practice, the VF profile can take any shape. The drive voltage can be boosted at certain operating frequencies to provide more torque at those speeds. For example, most VF profiles have a low-frequency boost region to help the motor start from zero speed. In this boost region, the voltage is limited to a minimum value. A typical VF profile is shown in Figure 9.

FIGURE 9: VF PROFILE EXAMPLE



SOFTWARE OPERATION

The source code supplied with this application note is a very basic assembly code example that allows you to get started with ACIM variable speed control.

ADC Sampling and VF Profile

The ADC is enabled to scan inputs AN7 and AN12, which are both connected to potentiometers on the dsPICDEM MC1 board. ADC conversions are triggered from the PWM module every 16 PWM cycles and the ADC interrupts the CPU after two sample/convert sequences. With this configuration, the ADC effectively samples AN7 and AN12 at a frequency of 500 Hz. The `ReadADC` subroutine is called at each ADC interrupt to read the conversion values and calculate the VF profile.

The potentiometer connected to AN7 sets the drive frequency. The 10-bit ADC result is right shifted by 2 and written to the `Frequency` variable. The frequency resolution is 0.244 Hz/bit, so the potentiometer can adjust the drive frequency up to 62 Hz with this scaling.

The AN12 potentiometer is used to set the voltage-to-frequency ratio, which determines the slope of the VF profile. The VF profile has maximum slope with the potentiometer set to full value. The ADC results for AN7 and AN12 are left shifted to convert them to fractional values, which simplifies the mathematics. In fractional mathematics, $0x7FFF = 0.999$. The two fractional values are multiplied using the fractional `MPY` instruction. The result of the multiply is a fractional value that can be used to scale the modulation voltage between 0 and 100%. This value is stored in the `Amplitude` variable.

Together, the `Frequency` and `Amplitude` values specify the input parameters for the sine wave generation. The value of `Amplitude` is limited, so that overmodulation will not occur in the PWM modulation routine.

In this application, the ADC and VF processing occurs at a 500-Hz rate to generate new voltage and frequency values. It should be noted that the VF processing only needs to be executed when the motor speed needs to be changed. This will be application dependent.

PWM Modulation

The PWM is configured for a 16 KHz carrier frequency, complementary outputs and 2 μ sec of dead time. The required amount of dead time depends on the power circuit that is used to drive the motor. A 64-entry sine wave look-up table is used to drive the motor.

The `Modulation` subroutine is called each PWM period to calculate the duty cycle for each of three motor phases. This subroutine is written to save and restore all working registers so that it can be reused in other applications.

The function first loads pointers to the sine look-up table and various variables and constants associated with the modulation routine. Although it is not an issue for this application, the sine table is stored in the program memory space to conserve RAM.

The old modulation angle, `Phase`, is added to the `Frequency` variable to get an adjusted modulation angle value. The 120-degree offset value, `0x5555`, is added to the adjusted angle value to get modulation angles for the 2nd and 3rd phase outputs. If you wish to drive a single-phase motor, these offset values could be modified. An offset value of `0x4000` would give a phase shift of 90 degrees. An offset of `0x8000` will provide a 180-degree phase shift.

After the three modulation angles are created, they are shifted to the right to discard all but the upper 6 bits. This is done because the sine table has 64 entries and only 6 bits are needed to form a pointer value. If you use a different size sine table, then the amount of shifting will need to be adjusted. After the right shift, the sine table pointers are left shifted by one to create a byte pointer. The table contains word values, so the pointer needs to be multiplied by 2.

The code that follows is repeated three times to get a duty cycle for each phase. Each of the three sine table pointers is added to the base program memory address of the sine table to form the actual table look-up address. A table read instruction is used to get a sine value from the table. Two multiplies and one addition are performed to calculate each duty cycle. The first multiply scales the sine look-up value for the desired modulation amplitude. The second multiply scales the sine look-up value to the allowable range of the PWM duty cycle values, which is based on the choice of PWM period. The PWM scaling factor is a value that represents 50% duty cycle. The scaling factor is then added to this result to provide a 50% duty cycle offset.

Sine Look-up Table

The sine wave data table values were calculated using a spreadsheet. The data are in 16-bit signed integer format, where `0x7FFF` represents +0.999 and `0x8000` represents -1.0. If desired, the data could be pre-scaled to the maximum PWM duty cycle to avoid one of the multiplication steps in the `Modulation` function.

SYSTEM SETUP

This section of the application note describes how to hook up the motor and dsPICDEM Motor Control Development System. The following discussion assumes you have the hardware listed in the “**Recommended Hardware**” section of this document.

dsPICDEM MC Board Setup

You will need to connect VR1 on your demo board. You can do this by borrowing the jumper on LK8 or LK9 from the CAN or RS-485 ports. Place the jumper on prototyping header J6, connecting VR1 to AN12.

How to Connect a 3-Phase AC Induction Motor to the Power Module

If you ordered your motor directly from a distributor, you will probably need to connect power wires to the motor. You will need four equal lengths of 12-16 AWG stranded wire and four wire nuts. It is best to keep the wire lengths less than 3 feet. Most three-phase ACIMs can be wired for two different operating voltages. There is usually a plate that can be removed from the back or side of the motor to access the wiring connections.

Follow the wiring diagram on the motor faceplate to install wires for the three power phases and the motor frame ground connection. Follow the faceplate wiring diagram for 208V operation. Strip and tin the ends of the wires that will attach to the power module.

Note: ALWAYS INSTALL A GROUND WIRE TO THE MOTOR FRAME TO AVOID THE RISK OF ELECTRICAL SHOCK.

Connect the three power wires from the motor to the R, Y and B terminals on the right side of the power module. It does not matter which power wires are connected to which power terminals, unless you need a specific direction of rotation. Be sure to connect the motor frame ground wire to the ground terminal on the right side of the power module.

How to Connect a Split-Phase AC Induction Motor to the Power Module

Split-phase ACIMs are designed with a primary winding and a secondary winding. Small split-phase ACIMs typically have power wires that are already installed. A split-phase ACIM will usually have three power wires: the primary winding, the secondary winding and a common wire that is connected to the other side of both windings.

You can figure out the function of each power wire using an ohmmeter. The secondary winding has a much higher resistance than the primary winding. When the ohmmeter is placed across the primary and secondary wires, you will measure the sum of the two winding resistances.

Once you have figured out what each wire does, label them ‘PRIMARY’, ‘SECONDARY’ and ‘COMMON’. Split-phase motors are designed to be driven with a capacitor inserted in series with the secondary winding, as shown in Figure 5. The capacitor inserts a phase shift in the secondary winding current so a rotating magnetic flux can be established from a single-phase voltage source. The capacitor specifications are provided by the motor manufacturer. If a capacitor is used, the motor can be connected to the power module, as shown in Figure 5. A split-phase ACIM can be driven by the power module without the secondary winding capacitor, if desired. The connection details for this configuration are shown in Figure 4.

Connection Steps

1. Connect an AC line cord to the terminals on the left side of the power module. You will need a cable that has an AC power plug on one end and stripped ends on the other. Use the following connections. Make sure the connections are secure.
 - Connect the green wire (ground) to the ground terminal
 - Connect the white wire to the N (neutral) terminal
 - Connect the black wire to the L (line) terminal

Note: You may have an AC power cord with a green wire, blue wire and a brown wire. If so, the green wire connects to the ground terminal, the blue wire connects to the N terminal and the brown wire connects to the L terminal.

2. Plug the dsPICDEM MC1 demo board into the power module. The board and power module mate using a 37-pin D-type connector. No cable should be used in between the board and power module.
3. Apply a source of 9 VDC power to the demo board.
4. Now, plug in the AC power cord to the HV power module. Look through the ventilation holes in the top of the power module. On the right side of the PCB, near the output terminals, there is a red LED that indicates high voltage is present on the DC bus. If you ever need to change the connections to the module, disconnect the power source and wait for this LED to extinguish.
5. Build the application note source code file, `acim_vhz.s` and program the dsPIC device. The device can be programmed using the MPLAB ICD 2 In-Circuit Debugger. Connect the ICD 2 communication cable to connector J4 on the left side of the dsPICDEM MC1 board. Make sure switch S2 on the board is placed in the 'ICD' position during device programming.
6. Before running the source code, ensure that VR1 and VR2 are turned to the minimum settings.

Input Voltage Considerations

Depending on your location, your source of AC power may vary in voltage. The power module can be supplied with any voltage up to 230 VAC. In some cases, you may have a mismatch between the AC line voltage and the design operating voltage for the motor. Many three-phase ACIMs are designed to operate with a 208-230 VAC input. You may only have a 120 VAC source of power available. The motor would still operate properly in this case, but with limited torque output. This will be no problem for the experiments described in subsequent sections. It simply means that for a given drive frequency, the drive voltage will be half the required value.

One way to solve the voltage mismatch problem is to limit the drive frequency range based on the limited voltage available. The `Frequency` variable that is calculated in the `ReadADC` subroutine can be divided by 2, which limits the frequency range to 31 Hz and thus, decreases the motor speed range by half. But, the motor will produce the proper amount of torque in that speed range.

EXPERIMENTS TO TRY

This section has some experiments that you can try. These experiments will give you a good understanding of how an ACIM responds to different voltages and frequencies in a variable speed application.

Experiment 1

In this experiment, you will observe the response of the motor to speed changes as a function of the VF profile slope.

1. Adjust VR1 to a moderate setting, between 25% and 50%, which sets the VF profile slope to a relatively low value.
2. Try changing the speed of the motor using VR2. You may notice that the motor speed changes are sluggish.
3. Now, try changing the value of VR1 and then changing the motor speed. You should find that motor speed changes occur much faster when the slope of the VF profile is increased.

You may discover that an overcurrent condition occurs if VR1 is set too high. A red LED will be lit on the power module to indicate this condition. If the overcurrent trip occurs, you can clear it by resetting the software. Pin RE9 is pulsed at start-up to provide a Fault Reset signal to the power module.

Experiment 2

In this experiment, you will observe how the motor torque changes when the frequency is held constant, but the amplitude is changed.

1. Start by setting VR2 to a very low value, less than 25%. This will set the modulation frequency to a relatively low value, which will run the motor at a low speed.
2. Initially set VR1 to zero and then slowly increase the setting while stopping the motor shaft with your hand. You should be able to feel the torque of the motor increase as VR1 is increased.

In this scenario, the motor is used in a variable torque mode. The frequency is held constant while the amplitude is changed.

CONCLUSION

This application note has presented a simple code example that you can use to implement variable speed control of an ACIM. For the details of an advanced ACIM application, please see AN908, "Using the dsPIC30F for Vector Control of an ACIM" (DS00908).

SOURCE CODE LISTING

Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the "Company") is intended and supplied to you, the Company's customer, for use solely and exclusively with products manufactured by the Company.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

```

;*****
;
;   Filename           :   acim_vhz.s
;
;*****
;   Notes:
;   =====
;   The A/D is enabled to sample two pots on the dsPICDEM-MC1 demo board
;   connected to AN7 and AN12.  VR1 is used to vary the V/Hz ratio of the
;   modulation.  VR2 is used to vary the modulation frequency.  By
;   experimenting with the two pot settings, you can find an optimal V/Hz
;   ratio to drive the motor.
;*****

.equ __30F6010, 1
.include "C:\pic30_tools\support\inc\p30f6010.inc"

.global __reset

;.....
;Configuration bits:
;.....

config __FOSC, CSW_FSCM_OFF & XT_PLL4      ;Turn off clock switching and
;fail-safe clock monitoring and
;use the XT osc and 4x PLL as
;system clock

config __FWDTP, WDT_OFF                    ;Turn off Watchdog Timer

config __FBORPOR, PBOR_ON & BORV_27 & PWRT_16 & MCLR_EN
;Set Brown-out Reset voltage and
;and set Power-up Timer to 16msecs

config __FGS, CODE_PROT_OFF               ;Set Code Protection Off for the
;General Segment

;.....
;Uninitialized variables in Near data memory (Lower 8Kb of RAM)
;.....

.section .nbss, "b"

; This variable is added to the 16-bit sine wave table pointer at each
; PWM period.  A value of 246 will provide 60 Hz modulation frequency
; with 16 KHz PWM
Frequency:.space 2

; This variable is used to set the modulation amplitude and scales the
; value retrieved from the sine wave table.  Valid values range from 0
; to 32767

```

AN984

Amplitude: .space 2

; This variable is the pointer to the sinewave table. It is incremented
; by the value of the Frequency variable at each PWM interrupt.

Phase: .space 2

.....
; Constants stored in Program space
.....

```
.section .sine_table, "x"  
.align256  
; This is a 64 entry sinewave table covering 360 degrees of the  
; sine function. These values were calculated using Microsoft  
; Excel and pasted into this program.
```

SineTable:

```
.hword 0,3212,6393,9512,12539,15446,18204,20787,23170,25329  
.hword 27245,28898,30273,31356,32137,32609,32767,32609,32137,31356,30273,28898  
.hword 27245,25329,23170,20787,18204,15446,12539,9512,6393,3212,0,-3212,-6393  
.hword -9512,-12539,-15446,-18204,-20787,-23170,-25329,-27245,-28898,-30273  
.hword -31356,-32137,-32609,-32767,-32609,-32137,-31356,-30273,-28898,-27245  
.hword -25329,-23170,-20787,-18204,-15446,-12539,-9512,-6393,-3212
```

.....
; Constants for this application
.....

```
; This constant is used to scale the sine lookup value to the valid range  
; of PWM duty cycles. This is based on the value written to PTPER. We will  
; PTPER = 230 for this application, which allows duty cycles between 0 and  
; 460. The sine table data is signed, so we will multiply the table data  
; by 230, then add a constant offset to scale the lookup data to positive  
; values
```

```
.equ PWM_Scaling, 230
```

```
; The pointer to the sign wave table is 16 bits. Adding 0x5555 to the  
; pointer will provide a 120 degree offset and 0xAAAA will give a 240  
; degree offset. These offsets are used to get the lookup values for  
; phase 2 and phase 3 of the PWM outputs.
```

```
.equ Offset_120, 0x5555
```

.....
; Code Section in Program Memory
.....

```
.text ; Start of Code section  
__reset:  
MOV #__SP_init, W15 ; Initialize the Stack Pointer  
MOV #__SPLIM_init, W0 ; Initialize the Stack Pointer Limit Register  
MOV W0, SPLIM  
NOP ; Add NOP to follow SPLIM initialization  
  
CALL _wreg_init ; Call _wreg_init subroutine  
; Optionally use RCALL instead of CALL  
  
call Setup ; Call the routine to setup I/O and PWM  
-----  
; Variable initialization  
-----  
  
clr Frequency  
clr Amplitude
```

```

;-----
; Main loop code
; The PWM interrupt flag is polled in the main loop
;-----

Loop:  btss   IFS2,#PWMIF           ; poll the PWM interrupt flag
      bra   CheckADC              ; if it is set, continue

      call  Modulation             ; call the sinewave modulation routine
      bclr  IFS2, #PWMIF          ; Clear the PWM interrupt flag

CheckADC:
      btss  IFS0,#ADIF
      bra  Loop

      call  ReadADC

      bra  Loop

;-----
; ADC processing subroutine
;-----
ReadADC:

      push.d W0
      push.d W4

      mov   ADCBUF0,W0            ; Read the ADC results into W0
      mov   ADCBUF1,W1            ; and W1.

      asr   W0,#2,W4              ; Right shift by 2 bits to get the
      mov   W4,Frequency          ; modulation frequency.

      sl    W1,#5,W4              ; Left shift AN7 and AN12 values to get
      sl    W0,#5,W5              ; 1.15 fractional data.
      mpy   W4*W5,A              ; multiply frequency by V/Hz gain to get
      sac   A,W0                 ; mod. amplitude. Store result in W0
      mov   #28000,W1            ; Limit modulation amplitude to avoid
      cp    W1,W0                ; dead-time induced distortion in PWM
      bra   GE,NoLimit           ; modulation.
      mov   W1,W0

NoLimit:
      mov   W0,Amplitude

      pop.d W4
      pop.d W0

      return

;-----
; PWM sine wave modulation subroutine
;-----
Modulation:
      push.d W0                  ; Save off working registers
      push.d W2
      push.d W4
      push.d W6
      push.d W8
      push.d W10

      ; The next three instructions initialize the TBLPAG and pointer register
      ; for access to the sinewave data in program memory using table reads.

      mov   #tblpage(SineTable),W0
      mov   W0,TBLPAG

```

AN984

```
mov    #tbloffset(SineTable),W0

; The next block of instructions loads various constants and variables
; used in the sinewave modulation routine.

mov    Phase,W1      ; Load the sinewave table pointer
mov    #Offset_120,W4 ; This is the value for a 120 degree offset
mov    Amplitude,W6   ; Load the Amplitude scaling factor
mov    #PWM_Scaling,W7 ; Load the PWM scaling value
mov    Frequency,W8   ; Load the Frequency constant that will
                    ; be added to the table pointer at each
                    ; interrupt.

; This is the pointer adjustment code. The Frequency value is added
; to the sine pointer to move through the sine table. Then, offsets
; are added to this pointer to get the phase 2 and phase 2 pointers.
; Note: If different phase offsets are desired, other constant values
; can be used here. Add 0x4000 to get a 90 degree offset, 0x8000 will
; provide a 180 degree offset. Here, 0x5555 has been loaded to W4
; to provide 120 degrees.

add    W8,W1,W1      ; Add the Frequency value to the sine pointer
add    W1,W4,W2      ; Add 120 degree offset value for phase 2
add    W2,W4,W3      ; Add another 120 degree offset for phase 3

; The sine table has 64 entries, so the pointers are right shifted
; to get a 6-bit pointer value.

lsr    W1,#10,W9     ; Shift the phase 1 pointer right to get the upper 6 bits
sl     W9,#1,W9      ; Left shift by one to convert to byte address
lsr    W2,#10,W10    ; Shift the phase 2 pointer right to get the upper 6 bits
sl     W10,#1,W10    ; Left shift by one to convert to byte address
lsr    W3,#10,W11    ; Shift the phase 3 pointer right to get the upper 6 bits
sl     W11,#1,W11    ; Left shift by one to convert to byte address

; Now, the pointer for each phase is added to the base table pointer
; to get the absolute table address for the lookup value. The lookup
; value is then scaled for the correct amplitude and for the range
; of valid duty cycles. The next block of instructions calculates
; the duty cycle for phase 1. The phase 2 and phase 3 code is the same.

add    W0,W9,W9      ; Form the table address for phase 1
tblrdl [W9],W5       ; Read the lookup value for phase 1
mpy    W5*W6,A       ; Multiply by the amplitude scaling
sac    A,W5          ; Store the scaled result
mpy    W5*W7,A       ; Multiply by the PWM scaling factor
sac    A,W8          ; Store the scaled result
add    W7,W8,W8      ; Add the PWM scaling factor to produce 50% offset
mov    W8,PDC1       ; Write the PWM duty cycle

; The next block of code calculates the duty cycle for phase 2.

add    W0,W10,W10    ; Form the table address for phase 2
tblrdl [W10],W5      ; Read the lookup value for phase 2
mpy    W5*W6,A       ; Multiply by the amplitude scaling
sac    A,W5          ; Store the scaled result
mpy    W5*W7,A       ; Multiply by the PWM scaling factor
sac    A,W8          ; Store the scaled result
add    W7,W8,W8      ; Add the PWM scaling factor to produce 50% offset
mov    W8,PDC2       ; Write the PWM duty cycle

; The next block of code calculates the duty cycle for phase 3.

add    W0,W11,W11    ; Form the table address for phase 3
tblrdl [W11],W5      ; Read the lookup value for phase 3
```



```

mpy    W5*W6,A      ; Multiply by the amplitude scaling
sac    A,W5         ; Store the scaled result
mpy    W5*W7,A      ; Multiply by the PWM scaling factor
sac    A,W8         ; Store the scaled result
add    W7,W8,W8     ; Add the PWM scaling factor to produce 50% offset
mov    W8,PDC3      ; Write the PWM duty cycle

```

```

; Now, save off the adjusted sinewave table pointer so it can be
; used during the next iteration of this code.

```

```

mov    W1,Phase

```

```

pop.d  W10          ; restore working registers
pop.d  W8
pop.d  W6
pop.d  W4
pop.d  W2
pop.d  W0

```

```

return          ; return from the subroutine

```

```

;-----
; PWM and ADC setup code
;-----

```

Setup:

```

; The first thing we need to do before enabling the PWM is to
; configure the I/O and reset the power module. The control board
; has a driver IC that buffers the PWM control lines. The active
; low output enable for this buffer is on port RD11.
; The power module has an active high reset line which is connected
; to port RE9.

```

```

clr    PORTD
clr    PORTE
mov    #0xF7FF,W0    ; Make RD11 an output to drive PWM buffer
mov    W0,TRISD      ; output enable.
mov    #0xFDFF,W0;
mov    W0,TRISE      ; Make RE9 an output for power module reset

```

```

; Now, ensure the power module is reset by driving the reset line for
; a few usec.

```

```

bset   PORTE,#9
repeat #39
nop
bclr   PORTE,#9

```

```

; Setup the ADC

```

```

mov    #0x0404,W0    ; scan inputs
mov    W0,ADCON2     ; 2 sample/converts per interrupt
mov    #0x0003,W0;
mov    W0,ADCON3     ; Tad is 2*Tcy
clr    ADCHS         ;
clr    ADPCFG        ; all A/D pins Analog mode
clr    ADCSSL        ;
bset   ADCSSL,#7     ; enable scan of AN7
bset   ADCSSL,#12    ; enable scan of AN12
mov    #0x8066,W0    ; enable A/D, PWM trigger, auto sample
mov    W0,ADCON1     ;
bclr   IFS0,#ADIF    ; clear A/D interrupt flag

```

AN984

```
    ; Now, setup the PWM registers

    mov    #0x0077,W0      ; complementary mode, #1, #2, and #3
    mov    W0,PWMCON1     ; pairs are enabled
    mov    #0x000F,W0     ; 2usec deadtime at 7.38 MIPS
    mov    W0,DTCON1
    mov    #PWM_Scaling, W0 ; set period for 16KHz PWM at 7.38 MIPS
    mov    W0,PTPER
    mov    #0x0001,W0     ;
    mov    W0,SEVTCMP    ; setup the special event trigger for the ADC
    mov    #0x0F00,W0     ; set the special event postscaler to 1:16
    mov    W0,PWMCON2    ;
    mov    #0x8002,W0     ; PWM timebase enabled, center aligned mode
    mov    W0,PTCON

    return                ; return from the Setup routine

;.....
;Subroutine: Initialization of W registers to 0x0000
;.....

_wreg_init:
    CLR W0
    MOV W0, W14
    REPEAT #12
    MOV W0, [++W14]
    CLR W14
    RETURN

;-----End of All Code Sections -----

.end                        ;End of program code in this file
```

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELoQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, MPASM, MPLIB, MPLINK, MPSIM, PICKit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance and WiperLock are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2005, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

**QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==**

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELoQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://support.microchip.com>
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

03/01/05