
Switching Power Supply Design with the PIC16F785

*Author: Keith Curtis
Microchip Technology Inc.*

INTRODUCTION

Microcontrollers are rapidly gaining ground as required components in the design of Switching Mode Power Supply designs (SMPS). Their flexibility and programmable nature gives manufacturers the ability to customize designs quickly in response to customer demands, and implement the wide variety of complex and custom features required by today's electronic systems. However, even with all of the microcontroller's strengths, adding a microcontroller to a SMPS design still adds one or more devices to an already crowded PCB.

That is, it used to add additional devices; the new PIC16F785 actually reduces the number of devices in a design by including not only the necessary interface peripherals for a SMPS design, but also two channels of analog PWM, two voltage comparators, and two op amps.

Now, all the parts needed to implement the analog control sections of up to two SMPS channels are included in the microcontroller. This means fewer parts to handle, a simpler layout, and even a lower material cost. In addition, the microcontroller control over the SMPS analog blocks allows control up through a Level 3 design (on/off control, output control, and topology/configuration control), something that is only rarely possible with a separate microcontroller/PWM controller solutions.

In this application note, we will examine a typical buck topology intelligent SMPS design using the PIC16F785. The design employs a Level 2 microcontroller integration, allowing the microcontroller to enable/disable and output voltage control. The microcontroller also has the ability to monitor the performance of the analog section of the SMPS design. Using both, control and monitoring, the microcontroller implements a variety of deterministic functions.

The following is a list of the electrical specifications for the design.

ELECTRICAL:

- 9-12 VDC input
- 2.0-3.9 VDC output at 10A
- <100 mV Ripple at the output
- Non-isolated buck-topology
- Current mode configuration
- Continuous inductor current operation
- Synchronous switching for increased efficiency
- Ambient temperature sensor

To keep the focus of the design, and simplify the firmware design, the communications system for the design will be limited to a simple parallel system. The following is a description of the interface:

COMMUNICATIONS:

- Active low SHUTDOWN input
- Active high POWERGOOD output
- Active high FAULT output
- Four preset voltage outputs, stored in on-chip EEPROM and selected by two voltage select jumpers.

The deterministic functions that are implemented in the design include the following:

DETERMINISTIC FEATURES:

- Delayed start-up for sequencing with other power supplies
- Soft start of the output voltage
- Under voltage lockout
- Slew rate limiting on output changes
- Hysteretic over temp error
- Shorted output Fault detection with limited restart
- Over current alarm

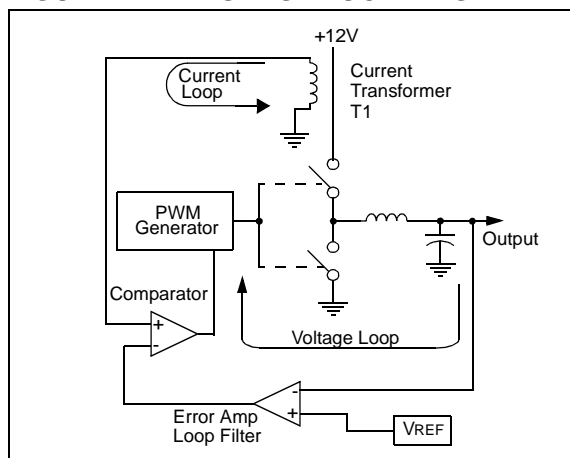
HARDWARE THEORY OF OPERATION

SMPS SECTION

To better understand how the on-chip peripherals in the microcontroller work in the traditional analog feedback control of a SMPS design, an examination of the operation of the SMPS power section is needed. Specifically, how the various peripherals are used, and what control the microcontroller can exert over their operation.

A block diagram of the switching regulator is shown in Figure 1. Note that there are two feedback paths, an inner current loop, and the outer voltage loop.

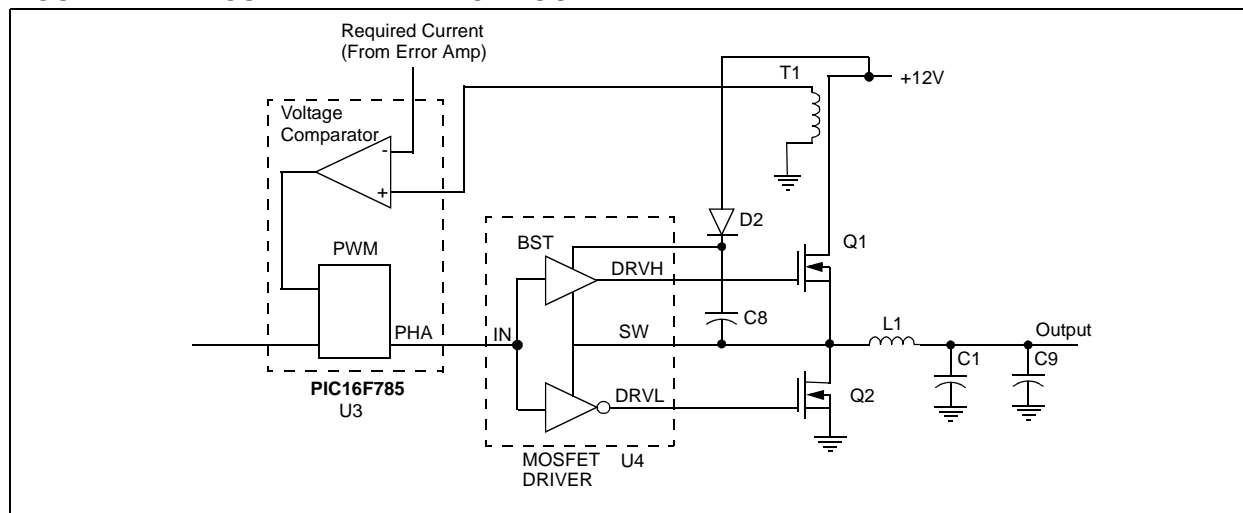
FIGURE 1: SMPS BLOCK DIAGRAM



CURRENT FEEDBACK LOOP

The inner current feedback path consists of a single channel of the on-chip analog PWM generator, the MOSFET driver U4, the two MOSFETs Q1 and Q2, the inductor L1, and the current transformer (T1) (see Figure 2).

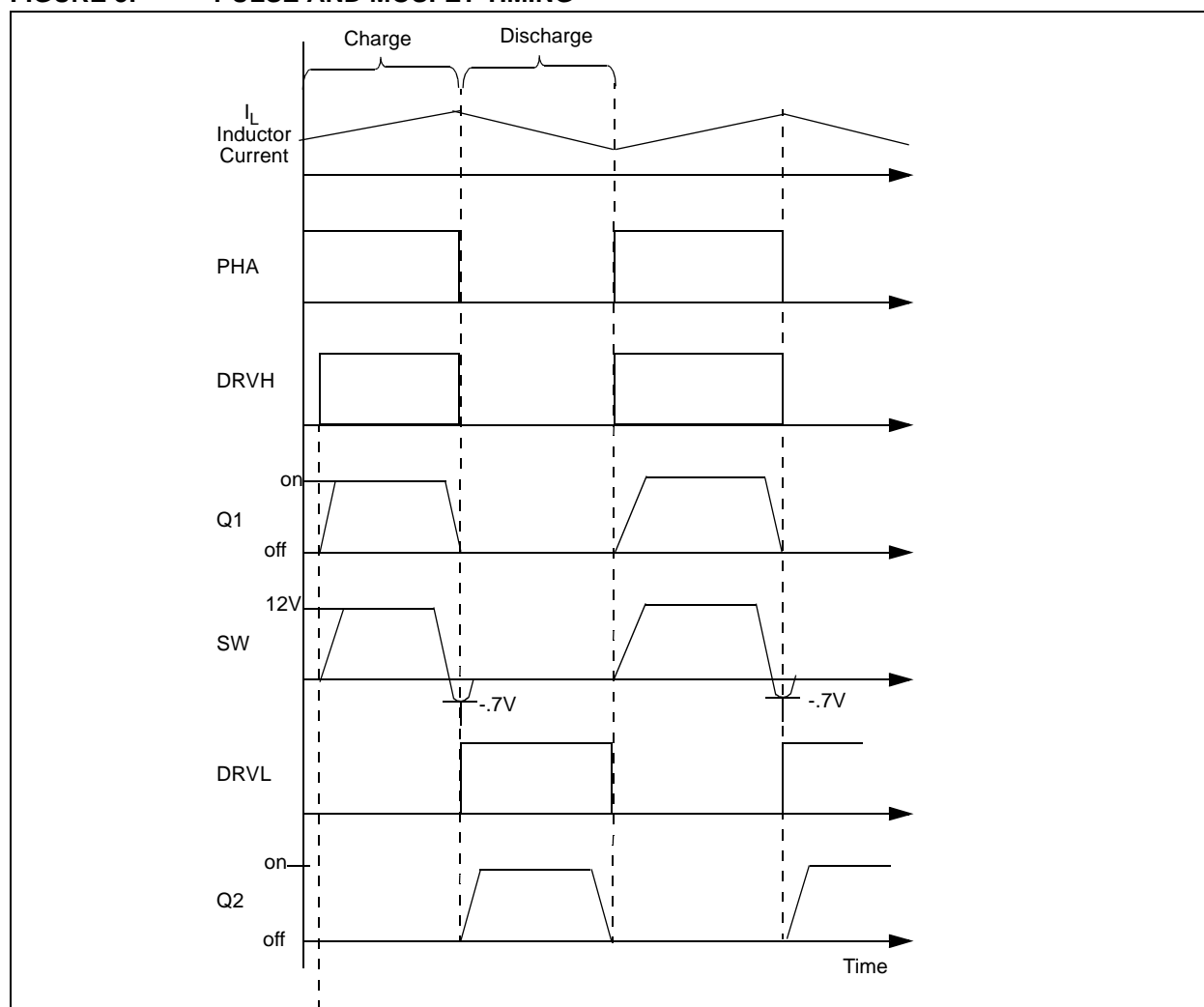
FIGURE 2: CURRENT FEEDBACK LOOP



The phase counter internal to the analog PWM module, initiates the PWM pulse by setting the PHA flip-flop. This sets the PHA output, which sets the DRVH output of the MOSFET driver U7, which turns on MOSFET Q1. While Q1 remains on, L1 is electrically connected between +12V and the output. The voltage difference between the 12V input and the SMPS output causes the current in L1 to ramp up, with the inductor current flowing from the +12V input, through the inductor L1, to

the output capacitors C1 and C9. The ramping inductor current continues until the feedback from the current transformer T1, reaches the desired level. When this happens, on-chip voltage comparator C2 resets the PHA flip-flop, which terminates the PWM pulse, and pulls the PHA output low. The DRVH output of U7 then goes low, and Q1 begins to turn off. See Figure 3 for a diagram of the waveforms.

FIGURE 3: PULSE AND MOSFET TIMING



Even though the gate of Q1 has been pulled low and Q1 is turning off, the current flowing through L1 does not stop. In fact, the current continues, driving the MOSFET side of L1 low in an attempt to keep the current flowing. This eventually drives the MOSFET side of L1 low enough to forward bias D1.

The sense input SW of U7 monitors the voltage on the MOSFET side of L1, waiting for the voltage at the inductor to drop from +12V to below -0.7V ground. When this occurs, the logic inside U7 turns on Q2 by pulling its DRV1 output high. This shorts out D1, causing the inductor current to flow from ground, through Q1, through L1, and finally to the output.

So, in general, the current feedback path of this design is very similar to the current feedback path of a discrete switching power supply design. However, there are a few notable differences:

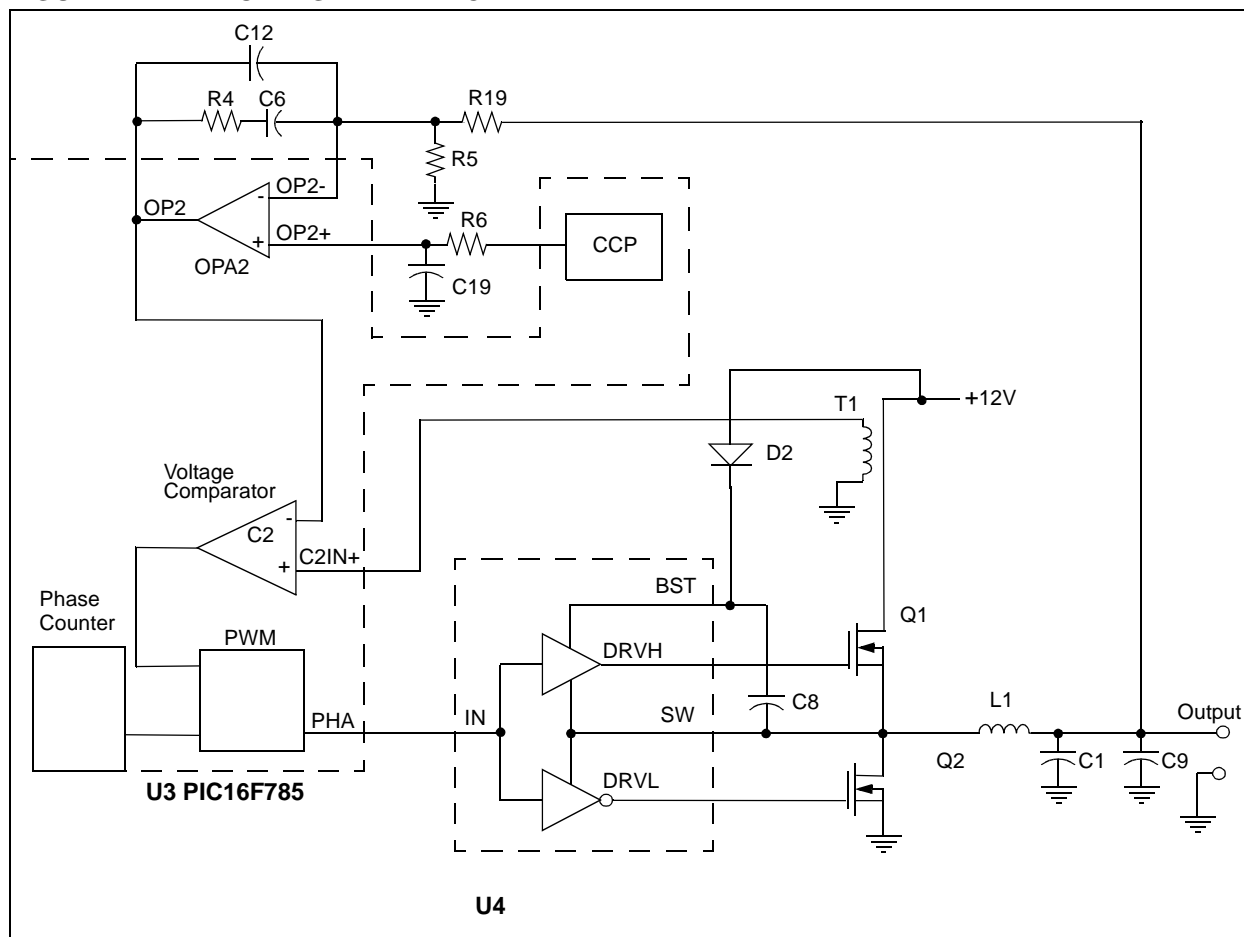
1. The frequency of the PWM generated by the PIC16F785 is programmable using the phase counter and prescaler within the analog PWM peripheral.
2. The phase delay between the PWM pulse outputs PHA and PHB is programmable in the PIC16F785.
3. The PWM module in the PIC16F785 can generate up to 2 channels with on-chip resources, and 3 channels using an external PWM generator such as the MCP1630.
4. The analog PWM module in the PIC16F785 can accept feedback from either single comparator, or both, for either PWM channel. This means;
 - a) Each comparator can be dedicated to a separate channel.
 - b) One comparator can drive both channels.
 - c) Both comparators can drive one, or both, channels.
5. The PIC16F785 can shutdown the PWM generator in software.

So, in addition to being a smaller solution, the PIC16F785 is also significantly more flexible.

VOLTAGE FEEDBACK

The desired current flow in L1, from the previous section, is set by the output of the error amplifier/loop filter in the voltage feedback loop. The voltage feedback loop consists of all the elements in the current feedback loop, plus the output capacitors C1 and C9, the error amplifier/loop filter (using the on-chip opa2 in the PIC16F785), and a voltage reference generated by the timer-based CCP PWM function in the microcontroller (see Figure 4). The current flow in L1 is designed to be continuous; it supplies all of the output current during the cycle, with the output capacitors storing the extra current from the high side of the charge cycle, for discharge during the low side. The challenge in a continuous current configuration is to make sure that the

inductor passes the right amount of current into the capacitors, regardless of the load. If the inductor current is too high, and the system is lightly loaded, the resulting output voltage could jump, or the output could have a large AC ripple. The task of regulating the amount of current in each charge/discharge cycle is the function of the voltage feedback loop.

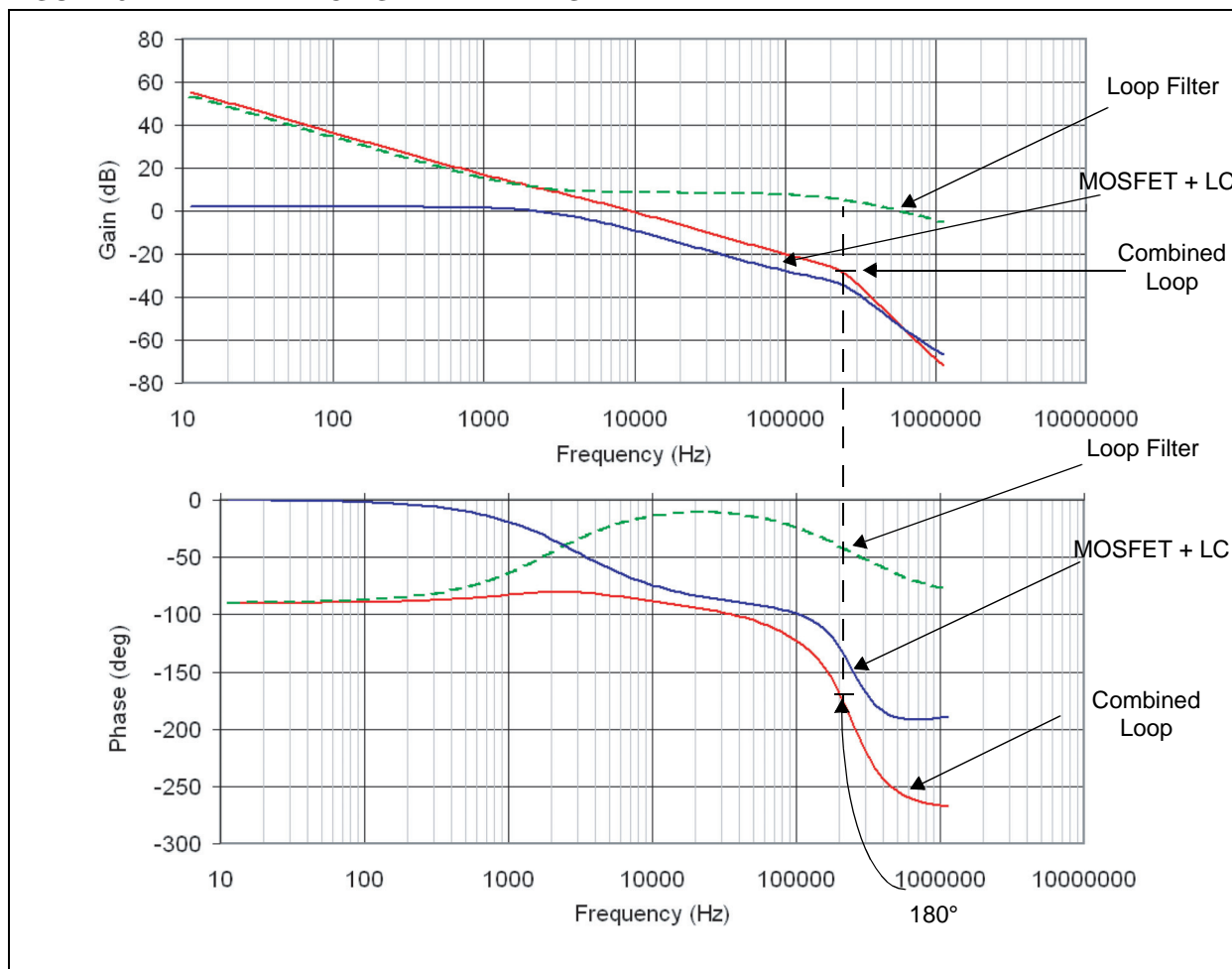
FIGURE 4: VOLTAGE FEEDBACK

The feedback process starts with the error amplifier and loop filter sampling the output voltage. The error amplifier portion of the circuit (OPA2) compares the output voltage to its reference voltage (C19), generating an error voltage. The error voltage is then passed to the voltage comparator (C2) in the current loop, which charges the inductor current up to the level of the error amplifier output. This keeps the current level in the inductor sufficient to maintain the charge in the output capacitors, and supply the necessary output current without overcharging the output capacitors and creating an output over voltage condition.

There are two hidden challenges in a continuous current design. One, the negative feedback of the voltage loop, and the phase delay of the various components, can create a condition in which the loop can go unstable and oscillate. Two, using a simple subtraction in the error amplifier will result in a constant error between the reference voltage and the feedback voltage.

It is the loop filter that acts to counteract the potential instability. Two poles and a zero in the loop filter's transfer function introduce both gain and phase changes in the feedback, such that the loop never has sufficient gain to oscillate when the phase delay is a multiple of 360 degrees (see Figure 5).

FIGURE 5: FEEDBACK GAIN AND PHASE



A secondary effect of the poles in the loop filter is their integration of any constant errors. This results in an offset of the error amplifier output such that any constant error is driven to zero.

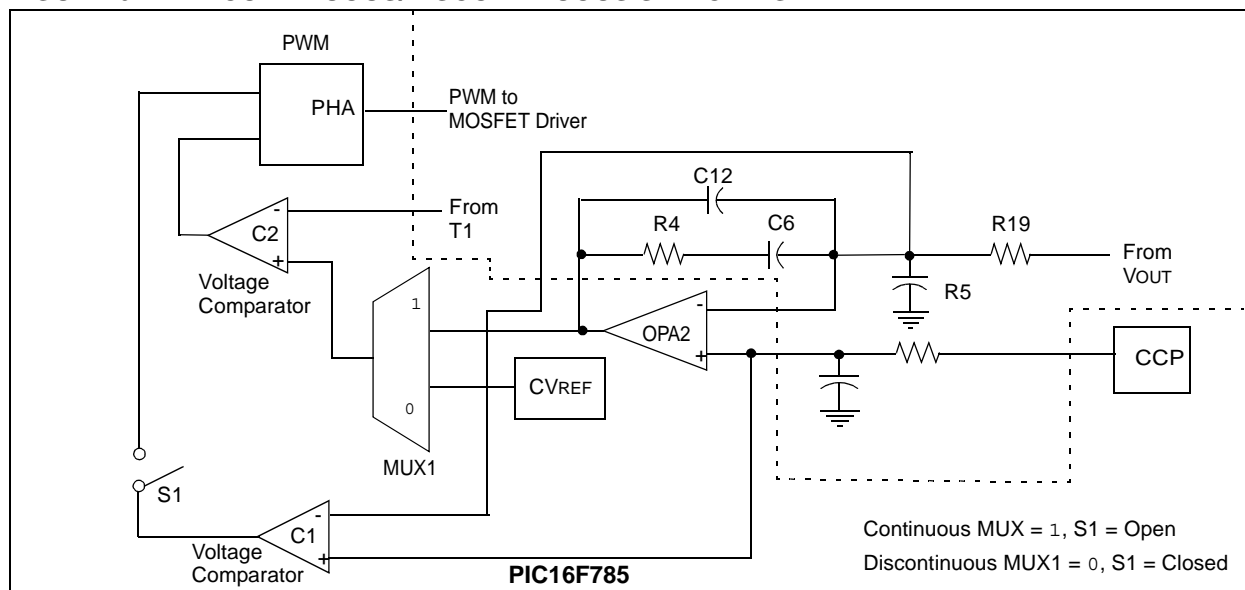
The location of the pole and zeros in the loop filter/error amplifier are determined by the resistors and capacitors in the feedback path of the op amp used as the error amplifier/loop filter.

As you might expect, the choice of the poles and zeros in the loop filter is critical to maintaining the stability of the feedback loop. The poles and zero in the loop filter of this design have been chosen to provide a gain margin (amplitude) of greater than 30dB, and a phase margin of over 90 degrees.

So, the operation of the voltage feedback system is nearly identical to any purely analog SMPS design. In fact, the use of the CCP base PWM to generate a reference voltage for the error amplifier is commonly used in intelligent SMPS designs. However, there are also some important features that make the single chip solution with the PIC16F785 significantly more flexible:

1. The analog multiplexers on the inputs to the comparators allow the software to switch between two or more loop filters, giving the system the ability to change its response characteristics.
2. The analog multiplexers on the inputs to the comparators, and the systems ability to reconfigure which comparators are used for the PWM feedback, allow the system to switch between a fully proportional feedback for continuous inductor current, and a hysteretic feedback for discontinuous inductor current on the fly (see Figure 6).

FIGURE 6: CONTINUOUS/DISCONTINUOUS SWITCHING



3. The two phase capability of the analog feedback PWM module allows the system to switch between a single, two, or three phase PWM load share system on the basis of load current, allowing the use of multiple smaller power chains in the place of one larger chain (see Figure 7 and 8).

FIGURE 7: MULTIPHASE CONVERTER

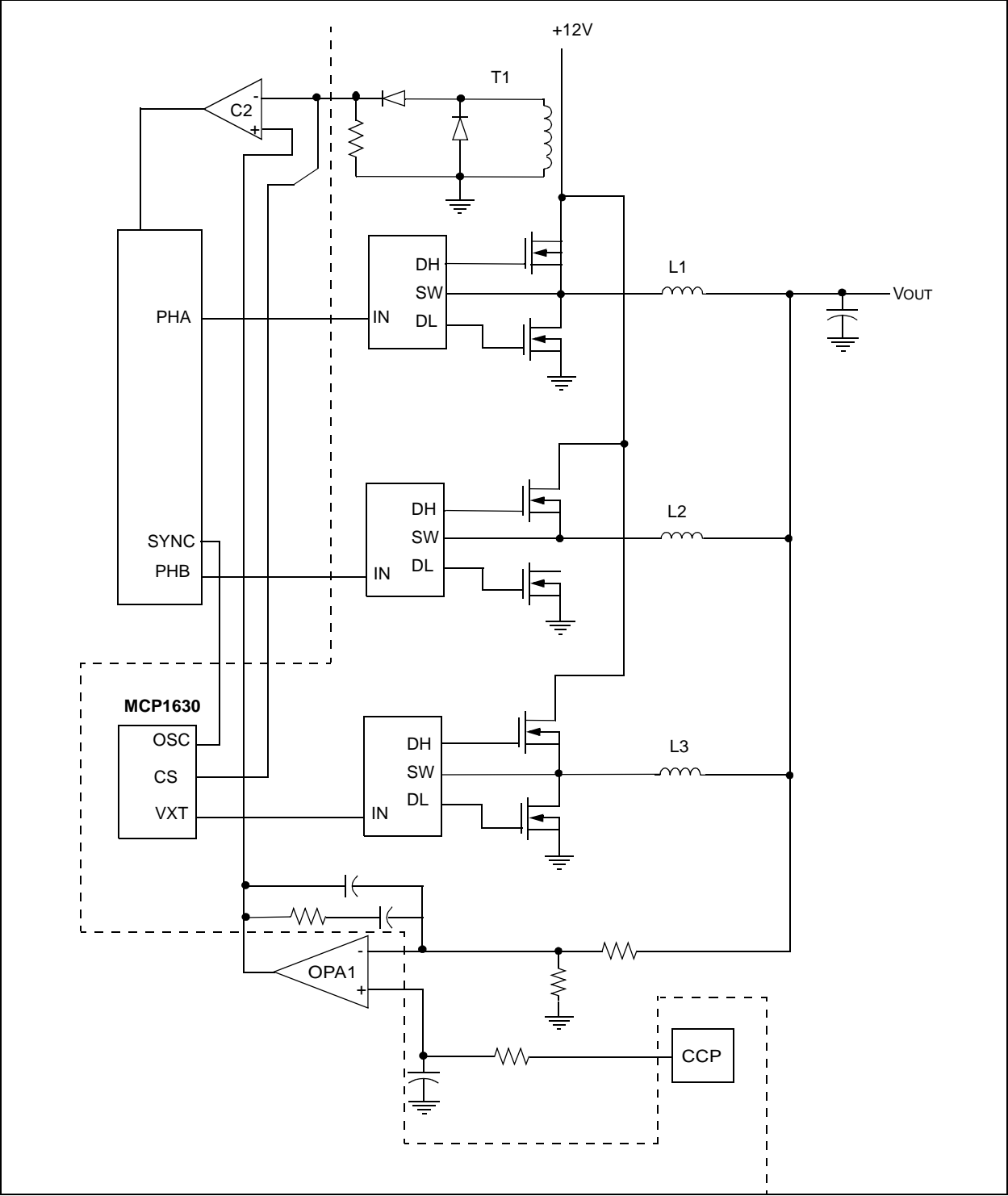
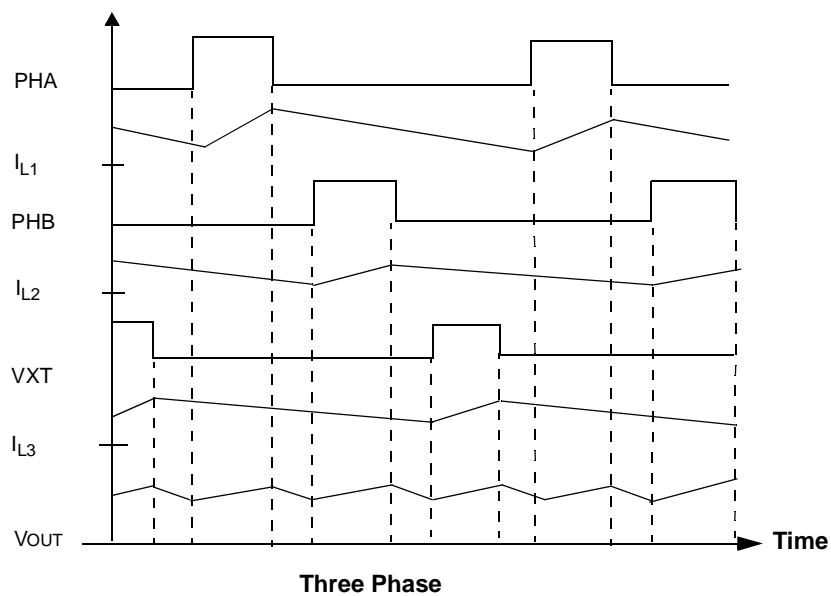
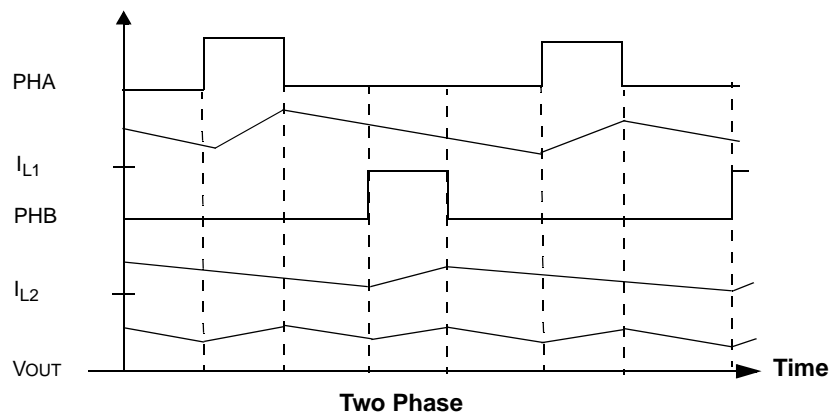
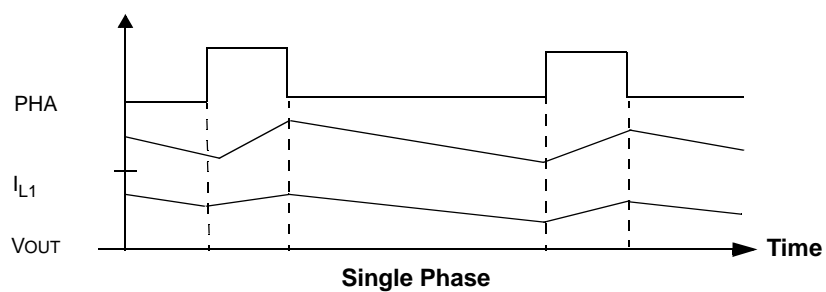


FIGURE 8: MULTIPHASE TIMING DIAGRAMS



4. The PIC16F785 is also available as the PIC16HV785 with an on-chip 5V shunt regulator for powering the microcontroller and associated circuitry.

So, the power conversion section of the power supply design is comprised of the two feedback loops, the inner loop regulating inductor current, and the outer loop regulating the voltage.

MICROCONTROLLER SECTION

In the beginning of this application note, it was stated that the purpose of the design was to provide a microcontroller control over the operation of the SMPS design, and a means to monitor its operation. So, what does the microcontroller control and what does it monitor? (see **Appendix A: “2.0V to 4.5V 10A Switcher Schematic”**).

Let's start with monitoring. Two obvious points to monitor are the input and output voltage. The input voltage will tell the microcontroller when the SMPS has sufficient voltage to operate properly, and the output voltage will tell the microcontroller when the SMPS is supplying the appropriate voltage.

So, the microcontroller will have to measure both voltages. Unfortunately, ADC peripherals do not have large common mode input voltage ranges, so if the voltage to be measured is greater than VDD, it will be necessary to scale the voltage down using a resistor divider.

This is in fact, the function of R17 and R18. They scale the input voltage down from 9-15 VDC, down to 1.17V-1.95V for conversion by the ADC. This will still provide the ADC with a 38 mV resolution in measuring the input voltage. The output of the SMPS is connected directly to RC3/AN7 for conversion by the ADC.

Note: The scaling value was chosen assuming a maximum supply voltage surge of 35V and an ADC reference of 5V.

Because the output voltage is limited by the 50% duty cycle (see the maximum duty cycle note), it is not necessary to scale the output voltage measurement input to the ADC.

U4 is a current mirror which generates an output voltage proportional to the output current. It is used to scale the output current for the ADC input on pin RA4.

The only other system value that needs to be monitored in the design is the ambient temperature of the system. This is accomplished by U1, a TC1047A analog linearized thermistor. Its output is monitored by the ADC through the analog input RA2/AN2.

Control of the SMPS design is via the reference voltage used by the error amplifier in the voltage feedback loop. By generating a voltage at the desired output voltage, the voltage feedback loop will drive the SMPS output to

the desired value. This means that the design will need some form of DAC to generate the digitally control reference voltage.

While DACs are available, their cost can be a problem in cost sensitive applications. Fortunately, there is an option; the CCP module within the microcontroller can be configured to produce a timer-based PWM signal on RC5/CCP1. This signal is then averaged out to a DC level using a low pass RC filter composed of R6 and C19. The resulting reference voltage is then related to the supply voltage of the microcontroller (5 VDC) by the duty cycle of the PWM. Also, given that the PWM has a 10-bit resolution, this means that the reference voltage can be controlled to within 0.1%, giving the microcontroller a more than adequate level of control.

Because the reference voltage PWM is based on the timers, internal to the microcontroller, its maximum frequency will be $1/1024^{\text{th}}$ of the microcontroller's instruction rate for 10 bits of resolution. As we will see later, the instruction rate of the microcontroller is 1 MIPS (million instructions per second). So, the reference voltage PWM would have a frequency of 1 kHz. To average out the PWM signal to less than 1 LSB, it will be necessary to place the corner frequency of the low pass filter at $1/1024^{\text{th}}$ of the PWM frequency, or roughly 1 Hz. This would mean that the maximum rate at which the power supply may ramp up its output will be roughly 1 second.

However, if the resolution of the reference voltage PWM is limited to 8 bits, resulting in control between 0.5% and 1%, then the frequency of the PWM signal is 4 times higher. The corner frequency of the low pass RC filter is correspondingly higher as well. So, dropping the resolution to 8 bits, moves the corner frequency to 16 Hz, giving the design the ability to change the output in 62.5 ms. For the purposes of our discussion here, this will be considered sufficient.

Note: If a 10-bit resolution is required, or the speed of the output must be increased, the use of a simple 2-pole op amp-based low pass filter in series with the existing RC low pass will speed up the system. A single pole RC low pass reduced the ripple voltage in its output by 10:1 for every decade of frequency between the pulse frequency and the filter corner frequency. Using a cascaded 3-pole filter reduces the ripple voltage at its output by 1000:1 for every decade of frequency. This means a 3-pole filter would allow the use of a 10-bit PWM, and still allow changes at a rate of 100 Hz or 10 ms.

Note: For systems that must switch faster than 10 ms, a DAC should be used. Several different varieties are available in various resolutions and interfaces. To connect an I²C™ DAC to the microcontroller, two unused I/O pins can be used to create a software-based I²C peripheral.

The microcontroller also supplies the clock signal for the PHA PWM, a 1 MHz clock which sets the frequency and maximum duty cycle of the PWM pulses in the SMPS. This clock signal is generated by the internal oscillator in the microcontroller, and routed internally to the phase counter in the analog PWM module.

One final connection to the SMPS portion of the design is required, the \overline{OD} control pin on U4. When this input is pulled low, the outputs of the MOSFET driver are disabled, and both MOSFETs are turned off. This has the effect of isolating the output of the SMPS from +12V and ground. For this design, the \overline{OD} control will be used to turn off the output, without creating a short circuit for the charge in the output capacitors to ground through the L1 and Q2.

There are three other connections to the microcontroller that will be required for the design; an In-Circuit Serial Programming™ port (ICSP™) through pins RA0, RA1, and RA3. The control inputs $\overline{SHUTDOWN}$, VSEL0, and VSEL1 through the RA3, RB6 and RB7 pins. And, the Status output POWERGOOD, on RA5. For the design presented here, the following convention will be applied to these interface connections;

- ICSP is only used for initial programming and the pins are left open or have alternate uses under normal operation.
- The $\overline{SHUTDOWN}$ pin will disable the output of the SMPS when pulled low.
- The POWERGOOD pin will indicate the correct output voltage is present when high, at all other times the output will be low indicating a fault condition or a command in the process of executing.
- The VSEL0 and VSEL1 pins form a 2-bit binary number selecting the desired output voltage. (VSEL0 is the LSB)

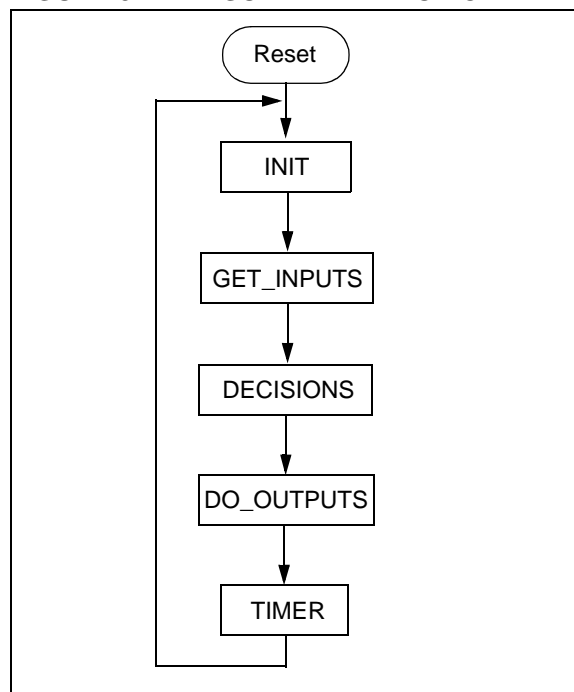
With the last of the connections to the microcontroller, the design of the hardware for the system is complete. It should be noted, that the additional circuitry for the microcontroller is limited to just the microcontroller itself and a few interface connectors and resistors, while the basic design of the SMPS section of the design is a traditional analog current mode design. The only microcontroller connections are the various points being monitored by the microcontroller, the PWM signal provided by the microcontroller, and the reference voltage.

Now that the microcontroller has access to the internal signals of the power supply, and a means of control over the output, the only remaining section to design is the software.

SOFTWARE THEORY OF OPERATION

The basic software construct that will be used is a simple infinite loop as shown in Figure 9.

FIGURE 9: SOFTWARE FLOWCHART



First in the flowchart is the INIT section, which contains all of the initial setting for the variables and peripherals that will be used by the system. The next section, located within the main loop, is the GET_INPUTS block, which is charged with measuring the various signals monitored by the microcontroller, and gathering the various communications and control inputs. The third section, DECISIONS, makes decisions based on the inputs to the system. The results from the DECISIONS block are then passed to the DO_OUTPUTS section which makes the appropriate adjustments to the controls, whether they are controls over the SMPS operation or communication outputs. The final section is the TIMER section which regulates the timing of the infinite loop. Here the rate at which inputs are gathered, decisions are made, and controls are adjusted, is regulated to provide predictable sampling and control rate changes.

Note: For simplicity and ease of reading, the various sections of the software presented here are written in C. The routines can, of course, be written in assembly, with potentially faster execution, however, this exercise is left to the reader, as assembly language versions of the routines would not add clarity to the discussion.

INIT ROUTINE

The first section to explore is the INIT section. As mentioned above, this section configures the various peripherals for the system and presets any system variables.

The first peripherals to be configured are the TRIS registers which control the direction of the Input/Output pins for the microcontroller. The ports are also preset to their initial states. Next, all analog inputs are configured by setting the ANSEL0 and ANSEL1 registers. This turns off the digital input to prevent shoot-through current in the input buffers when analog voltages are present.

The next peripherals to configure are Timer2 and the CCP module, which generate the reference voltage PWM signal. While the CCP module controls the duty cycle of the PWM output, and must be configured for PWM mode, Timer2 is the time base of the PWM and must be running for the CCP to generate and output pulse.

Timer2 is enabled, and both the prescaler and postscaler are disabled by loading the T2CON register. The period of the PWM pulse is then set by loading a hex 3F into the PR2 period register. This forces Timer2 to roll over to 0 from 3F. When the resulting 6-bit timer is combined with the internal 2-bit FOSC/4 counter, the PWM output frequency will be 500 kHz/256 or 1950 Hz.

The CCP module is configured for a single PWM output by loading the CCP1CON register.

Note 1: Loading the CCP1CON register also sets the 2 Least Significant bits of the PWM duty cycle. The 6 MSBs of the duty cycle are then set by loading the CCPR1L register. The final two PWM mode control registers, PWM1CON and ECCPAS, are left with their default settings. This disables both the automatic shutdown feature, and the H-bridge output dead-band control.

2: One possible addition to the system would be a fast shutdown capability in the event of an output over voltage condition. To implement this feature, one of the two voltage comparators would have to be configured to monitor the output voltage. When it exceeded its limit, the change in the comparator output would trigger the automatic shutdown and disable the reference voltage PWM.

The next peripheral to configure is the ADC. Loading the ADCON0 and ADCON1 registers sets five aspects of the ADC's operation.

1. Turns on the ADC.
2. Sets the ADC clock frequency.
3. Selects the initial input channel (VIN).
4. Selects VDD as the ADC reference voltage.
5. Configures the ADC for a right justified output (bit 0-7 in ADRESL, bits 8 and 9 in ADRESH).

Once configured, the ADC is now ready to perform its first conversion on the input voltage VIN. For convenience, this first conversion is started in the INIT routine. The value will be read and stored in the first call to the GET_INPUTS routine.

After the ADC, the op amps and comparators are enabled and configured. For this design op amp 2 and comparator 2 are used, with op amp 1 and comparator 1 disabled.

Note: The resources available on-chip are sufficient to build two SMPS voltage and current feedback loops, however, for this design, only one SMPS design is implemented.

Once the CCP is configured and generating a reference voltage, and the op amp and comparator used for the loop filter and current feedback are enabled, the analog PWM can be configured. This is done using the PWMCON, PWMCLK, and PWMPH registers.

For this design, a single output channel is enabled on PHA, using comparator C2 for feedback, with a phase offset of 0, and a pulse frequency of 500 kHz.

Note: The phase offset is relative to the SYNC output from the peripheral. The SYNC output/input is used to synchronize the pulse generation of multiple analog PWM modules, or to synchronize an external PWM such as the MCP1630. The offset capability is used to offset the start of a given phase's pulse relative to other pulses being generated by the on-chip PWM, or external PWM generators synchronized to the internal PWM peripheral.

The final peripheral to configure is Timer0. It is used as the time base for the TIMER section of the software. For the purposes of this design, the basic timing tick was chosen to be 1 ms. So, a prescaler value of 4:1, combined with the 8-bit Timer, and a 1 MIPS instruction rate will result in approximately a 1 ms roll over of Timer0.

The last activity in the INIT routine is the preset of any variables used by the system. Because the routines that use these variables have yet to be discussed, their preset values will be skipped over for now, and identified latter when the routine that uses the variables is discussed.

GET_INPUTS

The GET_INPUTS routine is charged with routinely sampling the various analog voltages in the system. To do this efficiently, a data indexed statemachine is used.

There are two standard forms of a statemachine, the execution indexed and the data indexed. In the more recognizable execution indexed form, a state variable is used to determine which, of several possible, blocks of code is to be executed when the statemachine is called (see Figure 10). A data indexed statemachine, on the other hand, executes the same block of code each time it is called. It is the data acted upon by the statemachine that is indexed by the state variable (see Figure 11).

This has several advantages for a routine such as the GET_INPUTS routine:

1. The basic routine to control the ADC is the same for all of the analog inputs; only the channel select bits are changed. Using a data indexed statemachine saves redundant coding for each channel.
2. If the results from the ADC are stored in an array, the state variable can be used to automatically access the correct location for storing the data, retrieving the next input configuration value, and accessing alarm limits for the individual analog inputs.

3. If the state variable is incremented each time the statemachine is called, the routine will automatically poll through all of the system inputs.
4. By manipulating the state variable, the order in which the inputs are sampled, can be altered and some analog inputs can be skipped as well.

The resulting routine is displayed in Example 1.

Each time the routine is called within the infinite loop, this statemachine will retrieve the result of the last conversion, configure the ADC for the next channel, and then start the conversion. The VSEL and SHUTDOWN inputs are also polled each time the statemachine is called.

EXAMPLE 1: DATA INDEXED STATE MACHINE

```
void GetADC()
{
    char channelSelect[4] = {0x89, 0x9D,
                             0x8D, 0xA5};
    ADCON0 = channelSelect[state1];
    DelayUs(20);
    GODONE = 1;
    while (GODONE == 1);
    monitor[state1] = ADRESL + (ADRESH*256);
    state1++;
    if (state1 > 3) state1 = 0;
}
```

Recall that the INIT routine begins this process by starting a conversion on the first input to be measured. That means that the first time the statemachine is called, there will already be a value waiting for the statemachine to retrieve. This pre-conversion, and the clearing of the data array, presets all the variables to safe values so the DECISIONS routine will function properly. This prevents random data in the array from causing undesirable decisions and erroneous outputs.

Note: The INIT routine will also preset the state variable for the GET_INPUTS routine, so it will know where to place the data from the first conversion.

FIGURE 10: EXECUTED INDEXED STATE MACHINE

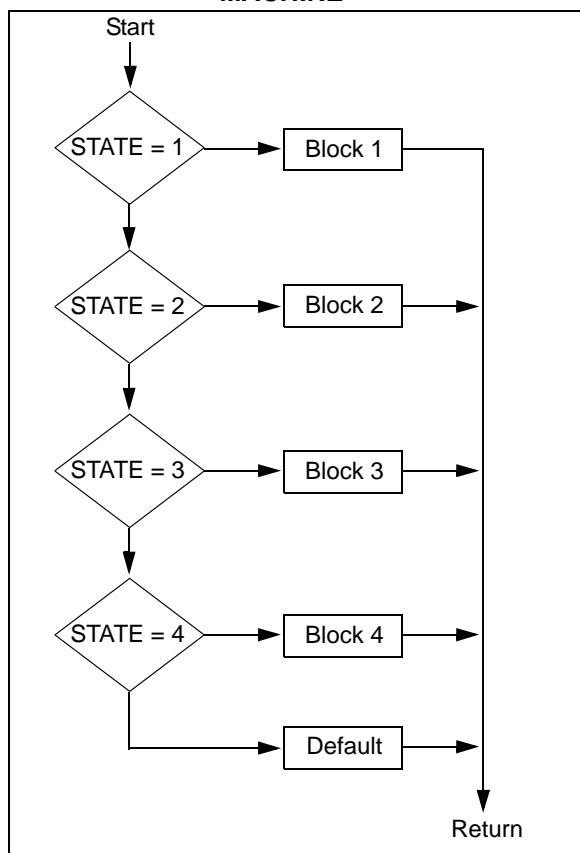
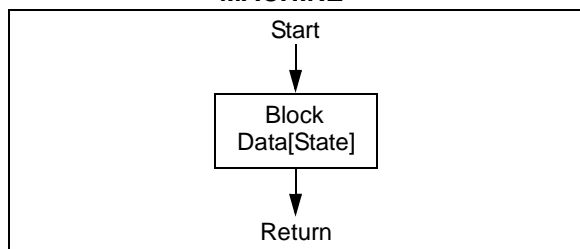


FIGURE 11: DATA INDEXED STATE MACHINE



DECISIONS

The next section of the firmware design is the DECISIONS section. In this section of the firmware, the data gathered in the GET_INPUTS section is analyzed and the appropriate actions are taken. To perform this function, an execution indexed statemachine is used.

To design the required statemachine, it is first necessary to determine the various states in which the system can exist, and the reasons for changes in these states. Two obvious states are: SHUTDN, in which the system is powered but is not producing an output, and ACTIVE, in which the system is powered and produc-

ing an output. For the other states needed, we start by examining the deterministic function requirements for the design.

DETERMINISTIC FUNCTIONS:

- Delayed start-up for sequencing
- Soft start
- Under voltage lockout
- Slew rate limiting on all VOUT changes
- Hysteretic over temp error
- Shorted output fault detection with limited restart
- Over current alarm
- Four programmable VOUT presets

The Delayed start-up indicates that there is a DELAY state needed between the SHUTDN and ACTIVE states.

Soft start function indicates that there will also be a RAMP state in which the output will be ramp up from 0 to the final output voltage.

Under voltage lockout does not seem to indicate an additional state, rather it is just a condition that would force the transition from ACTIVE to SHUTDN, or SHUTDN to DELAY.

Slew rate limiting on all VOUT changes indicates that are additional states for ramping the output up and down is required. However, if a single RAMP state is created to ramp up or down, a separate state will not be needed.

The hysteretic over temp fault requires that the system declare a error at one temperature, but then clear the error at lower temperature. This will require an additional active state in which there is a error, but the output is still active. Let's call this the ERROR state.

A shorted output condition is an immediate fault, and a simple jump to the SHUTDN state, followed by a restart, should be sufficient to handle the fault. However, if the fault persists, it may be necessary to have a "Sticky" fault condition that requires intervention by the communications function to clear. So, for this design, it seems advisable to create a Fault state which can be entered in response to a persistent fault and requires user intervention to move to the SHUTDN state.

The over current alarm is similar to the over temperature alarm, however, it doesn't require hysteresis, so it can be handled as simple comparison. If an over current condition exists, and if the power supply has completed the start-up delay and soft start, then turn on the alarm LED. When and if the condition clears, then turn off the LED. So, because the condition does not require an additional historical information to operate, a separate state is not required.

The four programmable VOUT presets simply require the system to recognize the new output voltage request, and ramp up or down to the new voltage. So, like the under voltage lockout function, this is just a condition forcing a move to the RAMP state.

So, if we gather up the various states for the system, we have:

SHUTDN	System is inactive
DELAY	System is delaying to start
RAMP	Output is ramping up or down
ACTIVE	Output is stable and active
ERROR	Output is active with an error
FAULT	Output is inactive pending release by the user

The next step is to identify the reasons for changing from one state to another. Again, from the deterministic functions list, and our understanding of the designs operation, we get the list of state transitions in Table 1.

TABLE 1: STATES

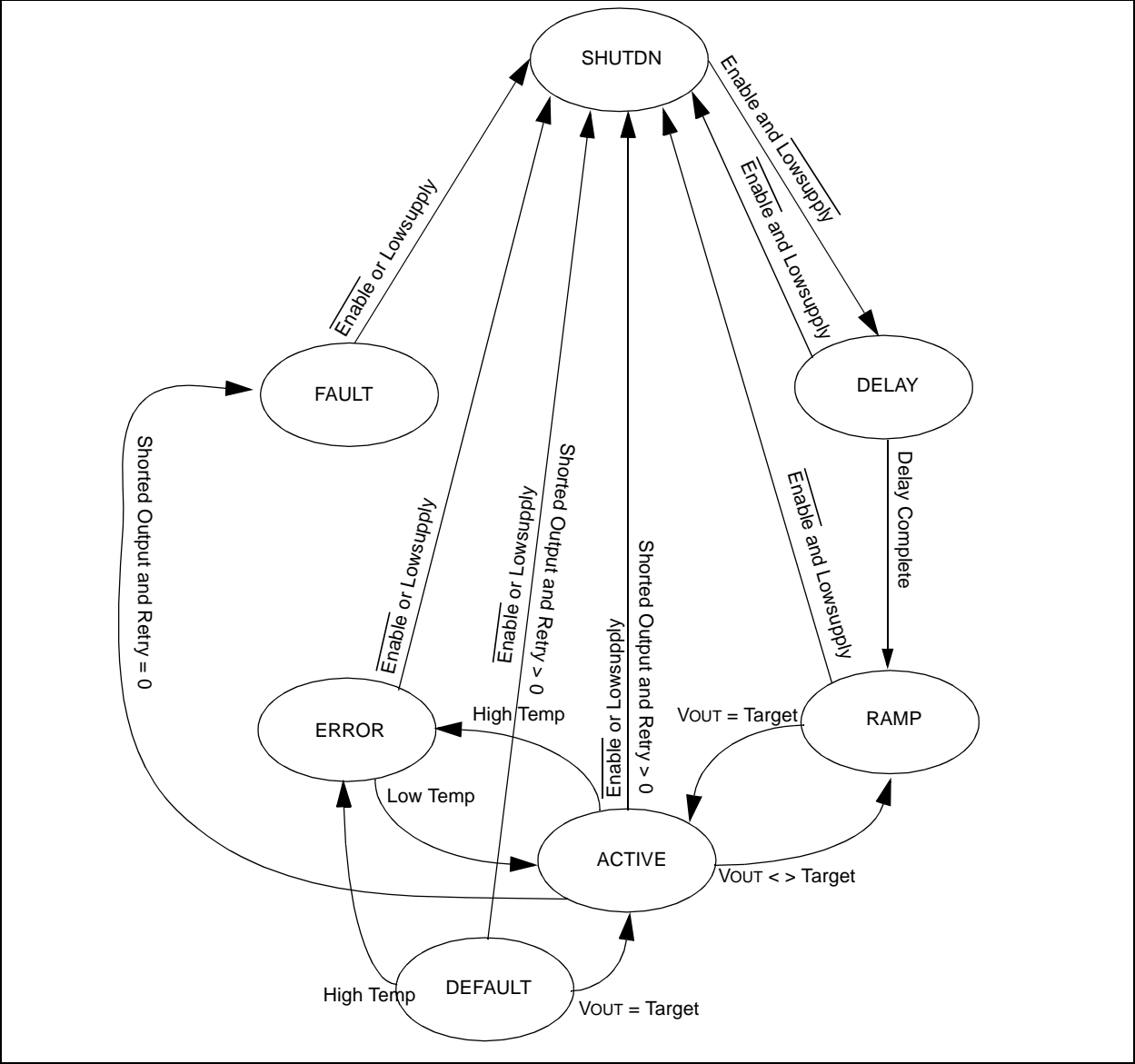
Current State	Next State	Reason for Change
SHUTDN	DELAY	SHUTDOWN = 1 and VIN > lockout
DELAY	RAMP	Delay complete
RAMP	ACTIVE	VOUT = Target
ACTIVE	RAMP	VOUT! = Target
(any)	SHUTDN	SHUTDOWN = 0 Or VIN < lockout
ACTIVE	ERROR	Temperature > highTempLimit
ACTIVE	FAULT	VOUT < ½ Target and Retry_count = 0
ACTIVE	SHUTDN	VOUT < ½ Target and Retry_count > 0
ERROR	ACTIVE	Temperature < lowTempLimit
ERROR	FAULT	VOUT < ½ Target and Retry_count = 0
ERROR	SHUTDN	VOUT < ½ Target and Retry_count > 0

Often, using a table for understanding the various states and their transition conditions can be confusing. Another method is to graphically plot out the state transitions as a series of circles and arrows. The circles represent the various states, and the arrows represent the state transitions. The reasons for the transitions are then written along the arrows. Figure 12 shows an example of such a plot based on the states and state transitions of the DECISIONS state machine.

Now that the statemachine is sufficiently defined, we can create the SWITCH statement and build in the state change conditional statements. Example 2 shows the resulting statemachine routine, plus the specific actions to be taken in each state. We can also go back to the INIT routine and add the necessary statement to preset the state machine to the SHUTDN state at start-up.

One final step in the design of the DECISIONS state-machine is to add a DEFAULT state to the SWITCH statement of the statemachine. This state will catch any state variable value which does not correspond to a legitimate state. While this is unlikely to occur during normal operation, it is possible that the state variable could become corrupted due to noise for EMI or RFI. Therefore, including the catch-all state is a prudent safety measure.

FIGURE 12: STATE TRANSITION DIAGRAM



EXAMPLE 2: MAIN EXECUTION INDEXED STATEMACHINE

```

void main()
{
    Init();
    while(1) {
        GetADC();
        switch (state2) {
            case SHUTDN:    POWERGOOD = 0;
                           retry = maxRetry;
                           FAULTLED = 1;
                           DutyCycleOut(0);
                           mDisableDrive
                           dutyCycle = 0;
                           waitCount = 100;
                           if ((ENABLE) && (monitor[3] > lowSupply))
                               state2 = DELAY;
                           break;

            case DELAY:     FAULT = 0;
                           waitCount--;
                           if (waitCount == 0)
                               state2 = RAMP;
                           break;

            case RAMP:      dutyCycle = dutyCycle + 1;
                           DutyCycleOut(dutyCycle);
                           mEnableDrive
                           if (dutyCycle >= GetVoltageSelect())
                               state2 = ACTIVE;
                           break;

            case ACTIVE:    POWERGOOD = 1;
                           if ((!ENABLE) || (monitor[3] < lowSupply))
                               state2 = IDLE;
                           if (monitor[2] > hiCurrent)
                               FAULTLED = 1;
                           else
                               FAULTLED = 0;
                           if (monitor[0] > hiTempLimit
                               state2 = ERROR;
                           if ((monitor[1] < (dutyCycle >> 1)) && (retry == 0))
                               state2 = FAULT;
                           if ((monitor[1] < (dutyCycle >> 1)) && (retry-- > 0))
                               state2 = SHUTDN;
                           break;

            case FAULT:     POWERGOOD = 0;
                           FAULTLED = 1;
                           if ((!ENABLE) & (monitor[3] < lowSupply))
                               state2 = SHUTDN;
                           break;

            case ERROR:     POWERGOOD = 1;
                           FAULTLED = 1;
                           if (monitor[0] < lowTempLimit)
                               state2 = ACTIVE;
                           if ((!ENABLE) & (monitor[3] < lowSupply))
                               state2 = SHUTDN;
                           break;

            default:        state2 = SHUTDN;
                           break;
        }
        DelayMs(10);
    }
}

```

The actions taken in the DEFAULT state include verifying the output voltage, and checking for other potential errors in the system operation. If no errors are found, and the system is generating the requested output voltage, the statemachine can return to the active state.

If not, then the power supply should go to the Fault state and wait for instructions from the user.

There are a couple of interesting points to note about the DECISIONS routine:

1. The state transition that checks for a low on the ENABLE input, or low input voltage is repeated in all states but SHUTDN. To save space, this statement could be moved before the SWITCH statement, requiring only one instance of the test.

Note: The test would also override any other condition that could change STATE.

2. The Fault state is "Sticky", it requires the user to pull the ENABLE input low to exit. This is the user intervention mentioned above.
3. The test for a shorted output in the ACTIVE and ERROR states directs the statemachine to either the SHUTDN state for restart, or the Fault state for hold based on the value in Retry_count. This implements the limited number of retry on faults.
4. The Retry_count variable is only reloaded in the SHUTDN state when the ENABLE input is low. This allows a user shutdown to reset the counter, but prevents a restart from resetting the counter.
5. The DEFAULT state does check the status of the system and jumps to the appropriate state to restart.
6. Target[Vselect] is an array of preset output voltages, indexed by the two voltage select inputs. If a select pin changes, the value returned is the new preset output.
7. The order in which state transitions is tested is important, the earlier tests have lower priority than the latter tests. So, the last condition tested has the ability to override the first condition tested. This simplifies the test statements by removing some of the conditions that might override a change.

The final result is a relatively simple system which allows for the complex interaction of multiple conditions and controls.

DO_OUTPUTS

This section of the code consists of functions to set the reference voltage duty cycle.

The function to set the reference voltage duty cycle has one complication, in that it has to set the two LSBs of the duty cycle in the CCP1CON register, without

affecting the other bits. The remaining 6 MSBs can then be loaded into the CCPR1L register. Example 3 shows how this is accomplished in the DO_OUTPUTS routine.

EXAMPLE 3:

```
void DutyCycleOut(unsigned int dc)
{
    CCP1CON &= 0xCF;
    CCP1CON |= (dc & 0x03) * 16;
    CCPR1L = dc / 4;
}
```

TIMER

The TIMER function is designed to synchronize the execution of the infinite loop to a hardware counter within the microcontroller. Doing this locks the start of the loop to a fixed time increment, and allows the loop to manage the timing of its samples and control outputs. If the TIMER function was not present, the rate at which the output changed would be a function of what other activities the microcontroller was performing at the time. This would make any ramping changes unpredictable and nonlinear. So, locking the start time of the loop to a fixed standard is the only way to maintain proper timing for the system.

The TIMER routine uses Timer0 as its time base. Recall that in the INIT routine, Timer0 was configured to roll over every 1 ms, so the system timing (tick) will be in increments of 1 ms. The TIMER routine is just a function that holds up execution until the next time Timer0 rolls over. It then clears the interrupt flag indicating the roll over and returns to the infinite loop so the next pass can commence.

Note: Because the system tick is 1 ms, the maximum start-up time for a 0 to maximum output voltage ramp is 256 ms or approximately 1/4 of a second. If the ramp needs to be faster, then either the Timer0 configuration should be changed to configure the prescaler for a different prescaler ratio, or the ramp state in the DECISIONS statemachine should be modified to increment the reference voltage duty cycle by an increment greater than 1.

CONCLUSIONS

The design presented here shows an alternative single-chip approach to adding intelligence to SMPS designs. The basic design is really unchanged. There are current and voltage feedback loops, a counter-based PWM is used to generate the reference voltage to the voltage loop, and the microcontroller uses the reference voltage to modify the operation of the system in response to conditions sensed through the ADC.

While the design presented here is rudimentary, it does show some of the advantages of having the analog components in the SMPS feedback paths within the microcontroller, and more importantly, under the microcontroller's control:

1. Giving the microcontroller access to the multiplexers on the comparator inputs gives the controlling firmware the ability to reconfigure the feedback pathways on the fly.
2. Control over the PWM generator allows the firmware to reconfigure pulse frequency, phase, and number of phases on the fly.
3. The on-chip peripheral reduces the parts count for the design, reducing stocking and sourcing problems.
4. The reprogrammability of the microcontroller, and its control over the various peripherals, allows the firmware to configure the product through firmware. Thus allowing production to customize the system at the end of the line.

All in all, using a microcontroller to implement a SMPS design allows for flexibility and greater capability. Using a microcontroller with the necessary analog functions on-chip, extends the flexibility and capability, while simplifying the layout of the design.

MEMORY USAGE

The memory usage for the control program is as follows:

Program memory	466 words out of 2K words
Data memory	30 bytes out of 128 bytes
EEPROM	8 bytes out of 256 bytes

Note: All code compiled with version 9.50 of HI-TECH's PICC.

Note: No compiler optimization was used.

APPENDIX A: 2.0V TO 4.5V 10A SWITCHER SCHEMATIC



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 devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. 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, KEELoQ logo, 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, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, PS logo, 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, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA 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.

© 2007, 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 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, 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

Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

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

Santa Clara

Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto

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

ASIA/PACIFIC

Asia Pacific Office

Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

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: 86-28-8665-5511
Fax: 86-28-8665-7889

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 - Qingdao

Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

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 - Wuhan

Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian

Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

ASIA/PACIFIC

India - Bangalore

Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi

Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune

Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama

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

Korea - Gumi

Tel: 82-54-473-4301
Fax: 82-54-473-4302

Korea - Seoul

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

Malaysia - Penang

Tel: 60-4-646-8870
Fax: 60-4-646-5086

Philippines - Manila

Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore

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

Taiwan - Hsin Chu

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

Taiwan - Kaohsiung

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

Taiwan - Taipei

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

Thailand - Bangkok

Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels

Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen

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

France - Paris

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

Germany - Munich

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

Spain - Madrid

Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham

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

12/08/06