

Offline Power Converter for High-Brightness LEDs Using the PIC16HV785 Microcontroller

*Author: Frank Hammerschmidt
Microchip Technology Inc.*

INTRODUCTION

High-brightness LEDs are finding their way into many popular applications and designers are now challenged to find new and more cost-effective ways of powering them. Many of these applications must be powered by an AC source requiring power factor correction, high efficiency, and possibly isolation for safety. Also, designers are looking for ways to add intelligence into their design. Often, it is not enough to turn the lights on and off. Applications need to sense internal and external changes and be able to respond intelligently to those events.

This application note demonstrates a simple flyback solution for an AC line-driven power converter capable of driving multiple high-brightness LEDs. Microchip's PIC16HV785 microcontroller manages the power section of the converter. The single chip controller solution integrates analog peripherals with an MCU and firmware, which, together, are used to create an intelligent lighting source.

The PIC16HV785 is a 20-pin microcontroller unit (MCU) with onboard peripherals useful for power converter design. These include Analog-to-Digital Converters (ADCs), op amps, comparators and

Pulse-Width Modulators (PWMs). Other peripherals include timers, general I/O ports, and a programmable, internal voltage reference. An internal shunt regulator is also included in this device, making it particularly useful for high voltage applications.

The solution described in this application note has the following specifications:

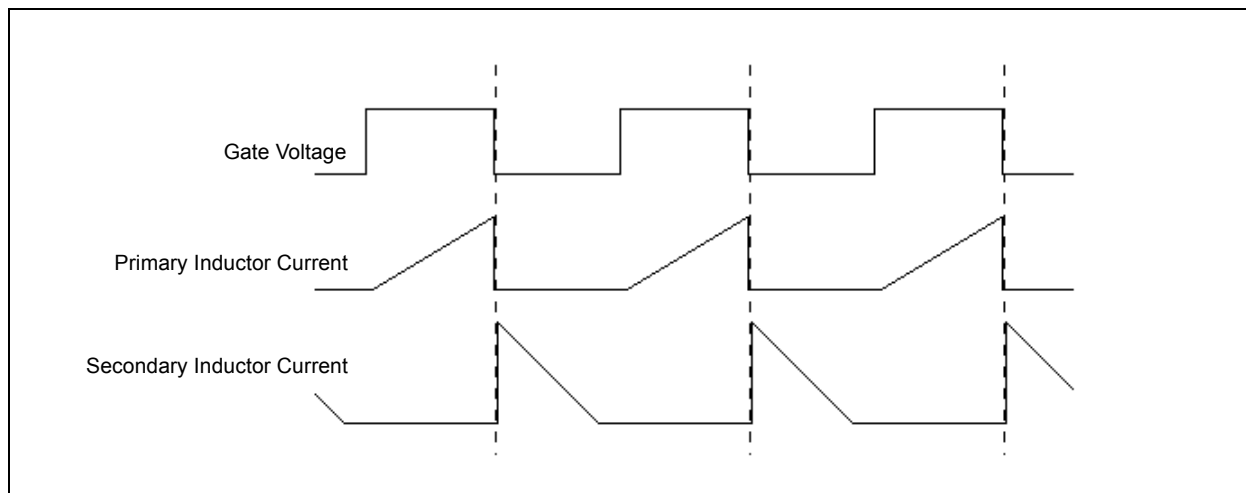
- Drives 5 power LEDs at up to 700 mA
- 85 – 265 VAC power input
- Efficiency > 80% and PFC > 0.95
- Dimming capability

THEORY

The design uses a flyback converter topology operating in Discontinuous Conduction Mode (DCM). Although the flyback is slightly less efficient than other topologies, it is still a very popular choice for AC line applications due to the electrical isolation provided by the transformer. Another useful characteristic of the flyback is its ability to produce multiple output voltages on separate windings of the same transformer.

DCM is characterized by the current flow through the storage inductor. In this mode, current begins to ramp on the primary winding during the 'on' stage of the switching cycle. After the switch is turned 'off', the current falls to zero and the energy stored in the primary is transferred to the secondary windings.

FIGURE 1: FLYBACK CONVERTER OPERATING IN DCM



AN1271

Because the current in the winding reaches zero each cycle, power can be determined by using the peak primary inductor current as shown in Equation 1.

EQUATION 1:

$$P = \frac{1}{2} LI_{Peak}^2 \times 1/T$$

Using the equation alone, the output power can be determined by information obtained from the primary side of the transformer. This eliminates the need for feedback across the isolation barrier and helps to reduce parts, cost and development time.

TRANSFORMER SELECTION

The number of LEDs that can be driven along with the maximum drive current is determined by the transformer characteristics. There are many resources for sizing the transformer. Coil manufacturers have considerable design experience and will make the decision easier. When contacting them, have some basic information ready. Typically, they will need to know the output power, including continuous current and operating voltages, along with the desired switching frequency.

The typical output current for the LEDs will be maintained at 350 mA, but the design will allow a maximum of 700 mA. In order to operate five LEDs at 700 mA, which requires approximately 4V drop across each LED, the master secondary winding will need to output 20V. At this current level, the transformer will need to transfer 14W to the output. At roughly 80% efficiency, the input power needs to be about 18W. This design operates at 150 kHz.

This design uses an additional bias winding to supply power back to the microcontroller once the converter is operating. The additional bias winding is designed to achieve a nominal 12V. This will be regulated to 5 volts using the PIC[®] MCUs internal shunt regulator. A lower voltage may be more efficient for the regulator, but the MOSFET gate voltage is the main reason for the higher voltage. Secondly, the bias voltage will change as the LEDs are dimmed.

With this information, the coil manufacturer suggested a transformer of 500 μ H primary inductance. This allows the transformer to store enough energy without saturating. Also, it allows the release of all the stored energy before the start of the next switching cycle, which keeps the converter operating in DCM.

SNUBBER CIRCUIT

All flyback transformers suffer from leakage inductance which can cause devastating effects to other components in the circuit. When the MOSFET is switched off, some of the energy stored in the

transformer core is released back into the primary coil. If there is no closed loop circuit available to dissipate the energy, then very high voltages will build up on the coil and cause damage to the other components. A snubber circuit will be required in order to minimize the peak voltages. This circuit consists of an avalanche diode, resistor, and series capacitor. When the voltage across the diode reaches 200V, it will begin to conduct and clamp the voltage to a level safe for the MOSFET.

MOSFET

Rearranging the power equation enables us to calculate the current through the MOSFET. At 18W, the peak current using a 500 μ H inductor switching at 150 kHz calculates to approximately 0.7A. An N-channel MOSFET was selected with a continuous current rating of 2 amps. This is playing it on the safe side, allowing for current surges and longer life with less heat stress.

Due to the start-up circuitry, it is essential that V_{GS} threshold for the MOSFET be at or below 5 volts. The MOSFET needs to begin switching before the transformer's auxiliary winding is operating, so there will not be 12V available at start-up. An MCP1402 gate driver is used to isolate the PIC[®] device from the higher drive voltage and to provide peak gate currents required to drive the MOSFET efficiently.

The maximum VAC input for this circuit may be as much as 265 Vrms. This equates to a peak voltage of 375V. This voltage, plus the additional 200V due to the leakage inductance spikes, requires the MOSFET voltage rating to handle up to 575V.

5V START-UP SUPPLY

In order to get operations started, a start-up, or bootstrap, supply will need to be created. A resistor is connected to the high voltage DC supply rail to supply a minimal amount of current to the 5V shunt regulator on the PIC device. The regulator requires a current supply of about 4 mA to begin operating. With a minimum input of 85 VAC, a 33K ohm resistor will be the maximum value that can be used. This allows just enough supply current to start the PIC device and keeps losses under 1/2W at higher input voltages. Initially, the power loss is greater, but after the converter is operating, power is supplied by the bias winding relieving some of the loss through the shunt resistor.

OUTPUT FILTER

As will be shown later, the instantaneous current will track the incoming rectified line voltage. Without filtering, this results in a 120 Hz component in the light output. The light flicker is visually not a concern because the human eye cannot detect it at this frequency. However, in order to maintain an average

output of 700 mA, the peak current may exceed the LEDs maximum rated current. For this reason, a small amount of capacitance is needed. One of the concerns with output filters is the cost of electrolytic components. Fairly large capacitances are needed to completely smooth out a 120 Hz rectified line voltage. It was determined for this design that a small 220 μF capacitor rated for 35V is needed to keep the current below the LEDs maximum current rating of 1A.

START-UP AND CONFIGURATION

The basic flyback topology is shown in Figure 2. It is a fairly simple circuit and requires little setup for proper operation. As power is initially applied to the circuit, the bus capacitor begins to charge through the shunt resistor, R_s . When the voltage reaches the minimum Power-on Reset of 1.8 volts, the power-on timer becomes active. After a short delay, the Reset is released and the firmware begins to run.

When the firmware initially starts up, there is not enough power stored in the bus capacitor, C_{BUS} , to begin running the flyback. A minimum of 5 volts is needed to fire the MOSFET, so the firmware places the PIC device into a Low-Power mode by leaving all peripherals turned off and setting the internal clock frequency to 32 kHz. In this state, the microcontroller draws very little current, allowing the bus capacitor to fully charge through resistor R_s . Without the delay in Low-Power mode, the 5V supply would quickly decay and a Brown-out Reset would occur.

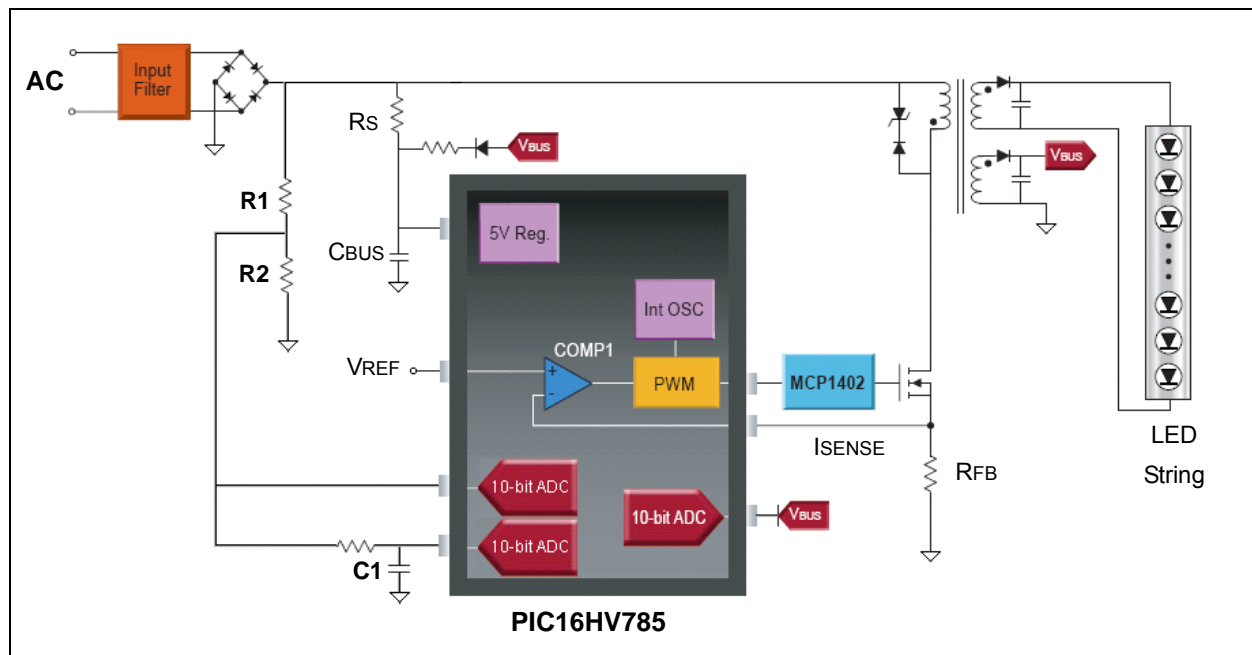
After the short code delay, the internal oscillator is set for 8 MHz and the PIC device will initiate configuration of the peripherals. There is very little configuration

needed to begin operation of the flyback converter. The two main peripherals required are the comparator and the PWM module. There is only a short time to get the peripherals configured. If the PIC MCU is running at full speed without the PWM output enabled, it will begin to drain the bus capacitor of its charge leaving little left to drive the MOSFET. Once this threshold is passed, the PIC device can do nothing until the voltage falls below the Brown-out Reset voltage and the PIC MCU restarts.

A high-speed comparator on the MCU is the central device for controlling the converter. It is used to compare the peak primary current to a given reference voltage and to drive the off-state of the PWM. This device can be configured to operate from several different inputs and provide various output options. In this arrangement, the CM2CON0 register sets the negative input to read the voltage from the current sense resistor applied to the C12IN2- pin on RC2. The positive input is configured to read a reference voltage from the C2IN+ pin on RC0. The output of the comparator drives the internal logic for the PWM duty cycle.

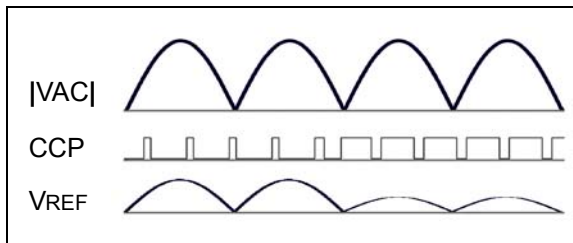
The PH1 channel of the two-phase PWM module is used to drive the gate of the MOSFET using the PH1 pin on RC1. The period is generated by the main system clock and the duty cycle is controlled by the C2 comparator. The PWM clock is configured using the PMWCLK register and is used to start the PWM cycle by driving the PH1 pin high. The comparator will detect the rising voltage from the current sense resistor and end the PWM cycle when the voltage reaches the reference. The PWM clock will continue counting until the period is completed and the cycle will restart.

FIGURE 2: SIMPLIFIED SCHEMATIC OF BASIC FLYBACK TOPOLOGY USING THE PIC16HV785



The second function of this circuit is to provide an adjustable, high resolution dimming solution. Figure 4 shows how the duty cycle of the transistor is used to adjust V_{REF} . As the PWM is increased, less of the incoming voltage is transferred to the RC filter, and thus a relatively lower reference voltage is generated. The PWM is a powerful way to adjust the power output and brightness levels of the LEDs. This solution offers dimming control by either a user interface, communications from an external source, or possibly automated dimming.

FIGURE 4: GENERATING V_{REF} FROM AC LINE VOLTAGE



OPERATION

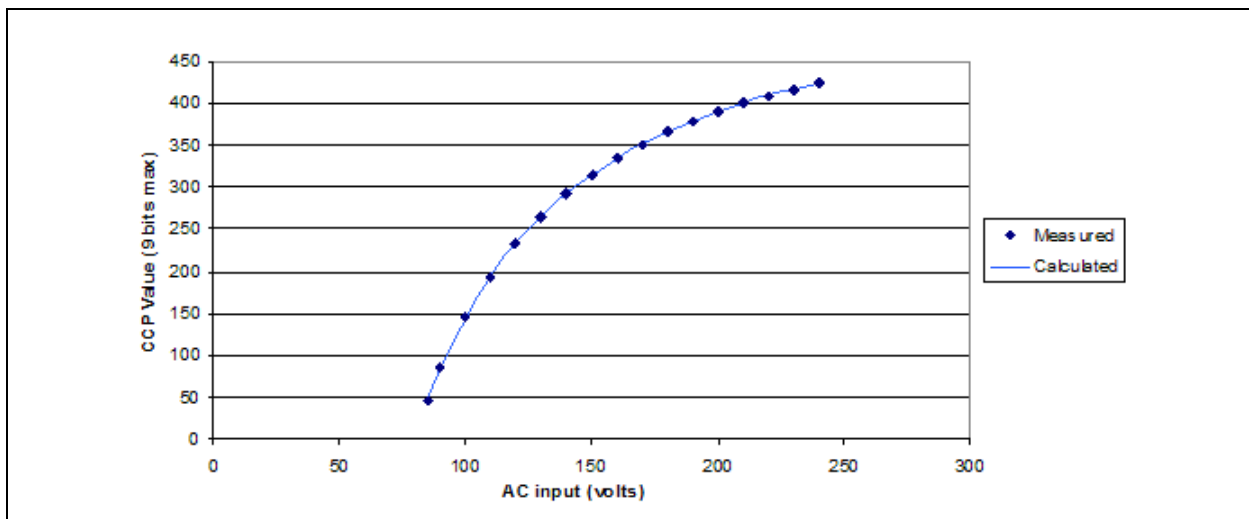
Before turning the flyback converter on, the firmware must determine the correct LED drive current at which the converter will begin operating. As previously stated, the brightness is a function of V_{REF} , which is based on the incoming line voltage and the PWM's duty cycle. Both of these inputs factor into generating the reference voltage and in maintaining the correct output power. A function will be necessary for calculating the CCP duty cycle based on these two inputs. The goal of this function is to determine the correct CCP values required to maintain a constant output power in the event of changing line voltages. Also, the function must adjust the CCP duty according to dimming requirements.

For start-up, the LED drive current and brightness is set to a minimum value. This reduces initial stress on the MOSFET and allows the firmware to smoothly transition up to the desired brightness level. Minimum brightness represents a maximum duty cycle of the dimming control's PWM. Of course, a 100% duty cycle would be completely on and no reference voltage would be generated, so the minimum brightness operating point is set to produce a 50 mA output. This was determined to be a point where all of the LEDs would stay equally lit with no flicker. In addition, the output drive cannot be reduced below a minimum level, because the bias supply for the PIC MCU and the gate driver is generated from an auxiliary winding on the transformer.

After the initial brightness is set, the incoming line voltage must be determined. The ADC is setup to measure the average voltage on AN9. Although there is a filter to smooth the incoming AC, there is still a small amount of ripple present. The ripple is enough so that the control algorithm would be continually chasing it to keep constant output. The firmware is used here to filter the remaining ripple. The voltage is read 64 times and the result is shifted to provide the average voltage. In practice, the average voltage does not change drastically, nor often enough to justify a more complex filtering algorithm.

Once the brightness is set and the incoming voltage is determined, the control's PWM duty cycle must be calculated and written to the CCP registers. For this, an experimental approach is used. The goal of the algorithm is to maintain an output of 350 mA based on changing line voltage. To determine the CCP values, a current meter was setup to measure the output current and a bench meter was used to measure the line voltage. A variac was used to sweep the line voltage from 85-265V.

FIGURE 5: MEASURED AND CALCULATED CCP VALUES FOR 350 mA OPERATION



AN1271

For each value of line voltage, the CCP value was determined to generate an output of 350 mA. The graphic in Figure 5 demonstrates the measured results of the sweep. The shape of this curve is due to the original power equation. The output power is a function of I_{PEAK}^2 , where the peak current is controlled by the CCP and the line voltage. In order to maintain the constant output power, the CCP value will need to vary by the square root of the line voltage.

With this data an equation can be formulated to solve for the CCP value. The data was entered into software designed for curve fitting and a suitable equation was found. Although the actual function should be a square root, it requires a significant amount of code to implement and there are other functions that fit very well. A less code intensive function was selected. The best equation was found to be the one in Equation 2,

EQUATION 2:

$$CCPvalue = \frac{k_1}{Vac} + k_2 \times Vac + k_3$$

where k_1 , k_2 , and k_3 are constants determined by the curve fitting software and Vac is the measured average line voltage. The calculated values along with the measured values are placed on the graphic in Figure 5.

Now that the required CCP value is determined, it is necessary to calculate the amount of dimming range available. For a Vac value of 85 volts with a CCP value of 46, it is clear that there is plenty of room left for dimming. In contrast, when the input is 240V and the CCP value required to produce 350 mA is already 425, there is significantly less duty cycle adjustment left to work with.

With this information, all that is necessary is to determine the maximum value of CCP to produce the dimmest setting. As stated earlier, 50 mA is the lowest current output capable of producing a steady, consistent light level. The duty cycle of the control's PWM produces a linear dimming range so, determining the CCP value at the min and max voltages is all that is required. The dimming range is simply the maximum dimming value minus the calculated CCP value for the given input voltage.

In firmware, the desired brightness level has been defined as a value from 0.0 to 1.0. This represents a percentage of the total possible brightness. After the dimming range has been determined, the final CCP value is computed by multiplying the dimming range by the brightness level and adding this back to the calculated CCP value. The following is an example with an input of 130V and 75% desired brightness.

At 130V the calculated CCP value for an output of 350 mA is 265. The maximum CCP values to produce 50 mA output were found to be 470 @ 85V and 511 @ 240. In

firmware, 85V represents an A2D value of d258 and 240V represents d615. This produces a slope of approximately 0.1 and a y-intercept of 440.

EQUATION 3:

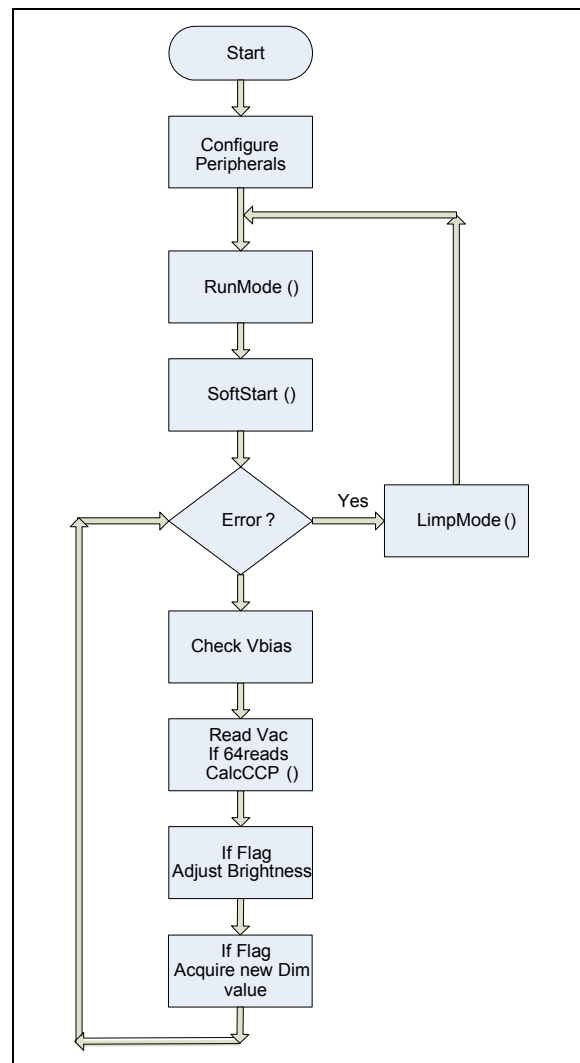
$$\begin{aligned} MaxDim &= 0.1(Vac) + 440 = 0.1(130) + 440 = 453 \\ DimRange &= 453 - 265 = 188 \\ DimValue &= 188 \times (1 - 0.75) = 47 \\ CCPValue &= CCPValue + DimValue = 265 + 47 = 312 \end{aligned}$$

Finally, the CCP value is written to the CCP registers and the desired brightness can be achieved.

FIRMWARE

The basic flow of firmware is shown in Figure 6. After the peripherals are setup and turned on, the Vac readings are performed and the initial CCP value is calculated. The brightness is set to a minimum and the PH1 PWM output is enabled.

FIGURE 6: FIRMWARE FLOW



The firmware now runs in a continual loop with only a few tasks to perform, since the current regulation function is performed by analog components. The first and most important task is to check the bias voltage for an overvoltage condition. As stated earlier, when the load is decreased or removed, the voltage on the secondary winding rises to an unsafe level. The maximum voltage that can be seen on the auxiliary winding is 18V due to the FET driver. When the overvoltage is detected, an error condition is flagged and the firmware enters the `LimpMode()` routine.

The `LimpMode()` routine shuts down all of the peripherals and places the PIC device into Low-Power mode. The PIC device must enter back into Low-Power mode, because the bias winding is no longer producing an output and the bus capacitor is quickly drained. During this time, the firmware delays for 150 ms before attempting to restart. A restart consists of turning the peripherals back on, recalculating the CCP value, and turning the PH1 PWM back on again.

Another task within the main loop is to take a reading of the AC voltage. A counter is setup to manage the 64 readings needed to get a steady average. Once the counter has reached 64, the `CalcCCP()` function described above is called and a new CCP value is calculated.

Two more tasks are performed when flags are set by timer interrupts. Events driven by the interrupts are used for dimming. To avoid abrupt changes in brightness levels, timers create a delay allowing smooth transitions to new power levels. The `AdjustDimFlag` signals the firmware to detect if the current brightness level is at the set point level. If they are not the same, the firmware enters a routine to increase or decrease the brightness level by 1%. This continues until the brightness level has reached the new set point. The `DelayFlag` is used for transition between the demonstration states. A small state machine is used to move between dimming levels.

SUMMARY

LEDs provide a unique light source that allows designers to utilize interesting circuit techniques to control them. In this article, it has been shown how the fairly constant forward voltage drop of a LED can be used to determine the output power using only the current on the primary side of the isolation transformer. This technique eliminates the need for isolation components to monitor the output current. The hardware peripherals found in the microcontroller are used to control the power with little firmware.

Another technique shown is the PFC. The AC line voltage is used to generate the reference voltage for the output control. This allows the output current to track the incoming line voltage which increases the overall power factor of the design.

The firmware is what allows the designer to really explore. In this reference design, the firmware was used to adjust the output power to maintain constant LED output regardless of line voltage. The firmware also demonstrates auto dimming.

APPENDIX B: BILL OF MATERIALS

TABLE 1: BILL OF MATERIALS

Ref. Des.	Description	Value	Manuf.	P/N	Digi-Key
R1, R3	Resistor, 1/8W Axial 1%	330k			
R2, R8, R13	Resistor, 0805 SMD 1%	4.7k			
R4	Resistor, 0805 SMD 1%	62k			
R5, R7, R20	Resistor, 0805 SMD 1%	10K			
R6, R10	Resistor, 1/4W Axial 1%	100K			
R9	Resistor, 0805 SMD 1%	1k			
R11	Resistor, 1206 SMD	200			
R12	Resistor, 1/2W Axial	47K			
R14	Resistor, 0805 SMD	10			
R15	Resistor, 0805 SMD	470			
R16	Resistor, 1/4W Metal Film 1%	1.5			PPC1.5W-1CT-ND
R17	Resistor, 1/4W Axial	22			
R18	Resistor, 1/4W Axial	100			
C1	Capacitor, 50V, 0805 SMD	6800p			
C2	Capacitor, 50V, 0805 SMD	1 μ			
C3, C4	Capacitor, 400V, 0.3" Spacing	0.047 μ	Panasonic	ECQ-E4473KF	EF4473-ND
C5	Capacitor, 600V, 0.3" Spacing	4700p	65N472MBKCA	478-4312-ND	
C6	Capacitor, 50V, 0805 SMD	1000p			
C7	Capacitor, 250VAC X1Y2 Radial	2200p			445-2424-ND
C8	Electrolytic cap, 35V, 0.1" Radial	47 μ			
C9, C12, C13	Capacitor, 50V, 0805 SMD	0.1 μ			
C10	Electrolytic cap, 35V, 0.2" Radial	220 μ			
C11	Electrolytic cap, 35V, 0.1" Radial	47 μ			
D1	Input rectifier bridge		Diodes Inc.	DF06MDI	DF06MDI-ND
D2	200V TVS		Littelfuse	P6KE200	P6KE200ALFCT-ND
D3, D4	Ultrafast diode, 600V			STTH2R06RL	497-4409-1-ND
D5, D6	Ultrafast diode, 50V			UF1001	UF1001DICT-ND
D7	Zener Diode,	27V		1N5254	
D8, D9	Shottky diode, 30V, SOD-323			BAT54WS-TP	BAT54WS-TPCT-ND
Q1	MOSFET, 600V, 2A		Toshiba	2SK3767Q	2SK3767Q-ND
Q2	NPN transistor 40V 600 mA SOT23			PMBT4401	568-1743-1-ND
L1	Common mode input choke		Panasonic	ELF15N005A	PLK1067-ND
T1	Coilcraft transformer		Coilcraft	GA3172-AL	
J1	Power inlet connector		Schurter	GSP1.9103.1	486-1130-ND
J2	ICSP™ programming header				
F1	Fuse 250V Slow 2AG 500 mA			0229.500H	F2466-ND
U1	PIC16HV785		Microchip		
U2	MCP1402		Microchip		

APPENDIX C: SOFTWARE

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.

```
#include <pic.h>

__CONFIG(INTIO & WDTDIS & PWRTEN);

#define V_OUT_MAX550// Max Vbias allowed - approx. 15 volts
#define V_OUT_MIN200// Min Vbias allowed - approx. 6 volts

#define OUTPUT_ENABLE() TRISC1=0
#define OUTPUT_DISABLE() TRISC1=1

#define MAX_BR0.0
#define MID_BR0.6
#define MIN_BR1.00

// Constants for calculating output power.
//          350mA    280mA
#define const1 -137400 // -123638
#define const2 -0.115 // -0.126
#define const3 718 // 721

#define A2D_ON          0x01
#define A2D_OFF         0xFE
#define VREN_ON         0x80
#define VREN_OFF        0x7F
#define CM1CON0_ON      0x80
#define CM1CON0_OFF     0x7F
#define CM2CON0_ON      0x80
#define CM2CON0_OFF     0x7F
#define TMR2_ON         0x04
#define TMR2_OFF        0xFB

#define Ch7 (unsigned char)0x7
#define Ch8 (unsigned char)0x8
#define Ch9 (unsigned char)0x9
```

```
// Globals
unsigned int    Vbias;
unsigned int    MaxDim;
float          SetDimValue;
float          Brightness;    // % Dimming 0 - 1.00

unsigned char DemoState;

//Flags
bit AdjustDimFlag = 0;
bit Error;
bit DelayFlag = 0;

// Function prototypes
unsigned int read_adc(void);
void WriteDutyCycle(unsigned int dutycycle);
void SetupCCP(void);
void SetupA2D(void);
void SelectA2DChannel(unsigned char Channel);
void LimpMode(void);
void RunMode(void);
void SoftStart(void);
void SetBrightness(void);
void DemoAdjust(void);
unsigned int ReadVac(void);
void CalcCCP(unsigned int);
void CalcMaxDim(unsigned int);

//-----
// void interrupt isr(void)
//
// The interrupt is used to set Flags for dimming.
//-----
void interrupt isr(void)
{
    unsigned int DelayCount;
    unsigned char TenMsecCount;

    if(TMR2IF)
    {
        TenMsecCount++;
        if(TenMsecCount == 40)
        {
            AdjustDimFlag = 1;    // set flag to adjust dimming
        }
    }
}
```

AN1271

```
TenMsecCount = 0;          // reset the 10ms counter
DelayCount++;

if(DelayCount == 400)
{
    DelayCount = 0;
    DelayFlag = 1;
}
}
TMR2IF = 0;
}

//-----
// void LimpMode(void)
//
// If an error is detected then shut everything down.  Wait in low power
// mode for bootstrap cap to fully charge.
//-----
void LimpMode()
{
    unsigned int i;

    OUTPUT_DISABLE();      // Disable the output

    GIE = 0;                // Disable interrupts
    TRISC5 = 1;            // turn off dimmer PWM

    ADCON0 &= A2D_OFF;     // Turn A2D OFF
    VRCON &= VREN_OFF;     // Turn Voltage Regulator OFF
    CM1CON0 &= CM1CON0_OFF; // Turn the comparators OFF
    CM2CON0 &= CM2CON0_OFF;
    T2CON  &= TMR2_OFF;    // Turn Timer2 OFF.
    VREN = 0;              // Turn Voltage Reference OFF
    CCP1CON = 0x0;
    PEIE = 0;

    OSCCON = 0;
    for(i=0; i<150; i++);
    OSCCON = 0x70; // switch to 8 MHz oscillator
}

//-----
// void RunMode(void)
```

```

//
// RunMode() turns all of the peripherals ON including the interrupts.
//-----
void RunMode()
{
    ADCON0 |= A2D_ON;          // Turn A2D ON
    VRCON  |= VREN_ON;        // Turn the Voltage reference ON.
    CM1CON0 |= CM1CON0_ON;    // Turn the comparators ON
    CM2CON0 |= CM2CON0_ON;
    T2CON  |= TMR2_ON;        // Turn Timer2 ON.
    VREN = 1;
    CCP1CON = 0x0C;
    TRISC5 = 0;
    PWMPH1 = 0x40;            // enable C2 for output reset
    PWMCLK = 0x39;            // 150KHz operation on PH1 PWM
    PWMCON0 = 0x91;           // auto-restart, blanking, PH1 enabled

    CVROE = 1;
    PEIE = 1;                 //turn on the peripheral interrupt
    GIE = 1;
}

//-----
// void SoftStart(void)
//
// SoftStart attempts to alleviate excess load put on the MOSFET by setting
// the Brightness to a minimum before turning on. This function measures
// the Vac to determine the minimum CCP value.
//-----
void SoftStart(void)
{
    unsigned int VacResult;
    unsigned int t;

    VacResult = 0;
    for(t=0; t<64; t++)
        VacResult = VacResult + ReadVac();
    VacResult >>= 6;

    Brightness = MIN_BR;     // Set output to a minimum before starting
    CalcCCP(VacResult);      // Setup the CCP register

    OUTPUT_ENABLE();         // Turn PH1 on.
}

```

AN1271

```
SelectA2DChannel (Ch8);
}

//-----
// void Main(void)
//
// Main() sets up the peripherals, but does not turn them on.
//-----
void main(void)
{
    unsigned int result;
    unsigned int Vac;

    unsigned int adc_channel = 0;
    unsigned char Delay;
    unsigned int i;
    unsigned char Count = 0;

    DemoState = 0;          // Initialize the state machine for demo mode

    OUTPUT_DISABLE();
    OSCCON = 0;
    for(i=0; i < 210; i++); // Wait in low power for bootstrap cap to fully charge //150
    OSCCON = 0x70; // switch to 8 MHz oscillator

    SetupA2D();           // Setup, but do not turn it on yet.

    VRCON = 0x21;         // C1VREN low range - turn on later
                          // VREF = Vdd / 24 * (b0001) = .208V
    CM1CON0 = 0x2F;       // Enable C1 on RC3 and its output for 60Hz ZC detect
    // positive input is internal VREF
    CM2CON0 = 0x3A;       // Turn these ON in RunMode()
    SetupCCP();

    while(1)
    {
        Error = 0;
        RunMode();
        SoftStart();
        SetDimValue = MAX_BR;
        while(!Error)
        {
            //**** Check VBias is good ****//

```

```
SelectA2DChannel(Ch8);
Vbias = read_adc();
if((Vbias > V_OUT_MAX))
{
    LimpMode();    // VBias is too high!
    Error = 1;
}

//**** Check VAC level      ****//
Count++;
result = ReadVac();
Vac = Vac + result;
if(Count == 64)
{
    Vac >>= 6;
    CalcCCP(Vac);
    Count = 0;
    Vac = 0;
}

//**** Adjust Brightness Level      ****//
if(AdjustDimFlag)
{
    if(Brightness != SetDimValue)
    {
        if(Brightness < SetDimValue)
            Brightness = Brightness + 0.01;
        else
            Brightness = Brightness - 0.01;

        AdjustDimFlag = 0;
    }
}

//**** Check new Dim Level      ****//
if(DelayFlag)
{
    SetBrightness();
    DelayFlag = 0;
}
}
}
```

AN1271

```
//-----  
// unsigned int ReadVac(void)  
//  
// Read the voltage on the AC line  
//-----  
unsigned int ReadVac(void)  
{  
    SelectA2DChannel(Ch9);  
    return (read_adc());  
}  
  
//-----  
// void CalcMaxDim(unsigned int)  
//  
// This function calculates the maximum CCPValue that can be used.  
// Near 50mA.  
//-----  
void CalcMaxDim(unsigned int VacVal)  
{  
    MaxDim = (unsigned int)((float)VacVal * 0.1 + 447);  
}  
  
//-----  
// void SetupA2D(void)  
//  
// Initializes the A2D to measure Vac, VBias.  
//-----  
void SetupA2D(void)  
{  
    TRISC0 = 1;           // scaled AC reference input  
    TRISC3 = 1;           // AC voltage feedback  
    TRISA2 = 0;           // C1 output  
    ANSEL0 = 0xD8;        // AN0, AN3, AN4, AN6, AN7 are analog  
    ANSEL1 = 0x0F;        // AN8 - AN11 are analog  
    ADCON1 = 0x50;        // Fosc/16  
    ADCON0 = 0xA0;        // Right Justified  
                           // Vdd reference  
                           // Select AN8 <- VBIAS  
                           // Turn A/D Off  
}  
  
//-----  
// void SelectA2DChannel(unsigned char)  
//
```

```
// Selects the channel of the A2D which will to be read.
//-----
void SelectA2DChannel(unsigned char Channel)
{
    CHS0 = Channel & 0x01;
    CHS1 = (Channel >> 1) & 0x01;
    CHS2 = (Channel >> 2) & 0x01;
    CHS3 = (Channel >> 3) & 0x01;
}

//-----
// unsigned int read_adc(void)
//
// Reads the selected A2D channel and returns the value.
//-----
unsigned int read_adc(void)
{
    unsigned int adconv;

    GODONE = 1;
    while(GODONE);
    adconv = (((unsigned int)ADRESH << 8) + ADRESL);
    return adconv;
}

//-----
// void SetupCCP(void)
//
// Setup the CCPregister and timer to be used for dimming
//-----
void SetupCCP(void)
{
    PR2 = 130;          // 15KHz PWM with Fosc = 8MHz
    TMR2 = 0;
    CCP1L = 0;
    CCP1CON = 0x0C;
    T2CON = 0x18;      // TMR2 with 1:16 interrupt postscale
                    // Interrupt rate will be 6250 Hz
    T2CON |= TMR2_ON;    // Turn Timer2 ON.

    TMR2IF = 0;
    while(!TMR2IF);
    TMR2IF = 0;
    TMR2IE = 1;
}
```

AN1271

```
TRISC5 = 0;
}

//-----
// void WriteDutyCycle(unsigned int)
//
// Write the passed value into the CCP register for setting the duty
// cycle.
//-----
void WriteDutyCycle(unsigned int dutycycle)
{
if(dutycycle & 0x0001)  DC1B0 = 1;
elseDC1B0 = 0;
dutycycle >>= 1;
if(dutycycle & 0x0001)  DC1B1 = 1;
elseDC1B1 = 0;
dutycycle >>= 1;
CCPR1L = (unsigned char)dutycycle;
}

//-----
// void CalcCCP(unsigned int)
//
// This function calculates the CCPValue based on the measured Vac and
// adjusts it with the current DimValue.
//-----
void CalcCCP(unsigned int VacValue)
{
    unsigned int CCPValue;
    unsigned int DimValue;
    unsigned int DimRange;
    int temp1 = 0;
    int temp2 = 0;

    // Calculate CCP Value
    // Although the function is really a square root, we can
    // approximate with a simpler equation.
    temp1 = (int)(const1 / VacValue);
    temp2 = (int)(const2 * (float)VacValue);
    CCPValue = (unsigned int)(temp1 + temp2 + const3);

    // Calculate Dimming and adjust the CCP Value.
    CalcMaxDim(VacValue);
    DimRange = MaxDim - CCPValue;
```

```
DimValue = (unsigned int)((float)DimRange * Brightness);
CCPValue = CCPValue + DimValue;

WriteDutyCycle(CCPValue);
}

//-----
// void SetBrightness(void)
//
// Set the Brightness of the LEDs. Place code here to read external
// sensors for dimming adjustment. In this case, the Demo code is
// called to stimulate dimming.
//-----
void SetBrightness(void)
{
    DemoAdjust();
}

//-----
// void DemoAdjust(void)
//
// Demo code to stimulate dimming
//-----
void DemoAdjust(void)
{

    switch(DemoState)
    {
        case 0:
        {
            SetDimValue = MAX_BR;
            DemoState++;
            break;
        }
        case 1:
        {
            SetDimValue = MIN_BR;
            DemoState++;
            break;
        }
        case 2:
        {
            SetDimValue = MAX_BR;
            DemoState++;
        }
    }
}
```

AN1271

```
        break;
    }
    case 3:
    {
        SetDimValue = MID_BR;
        DemoState++;
        break;
    }
    case 4:
    {
        SetDimValue = MIN_BR;
        DemoState++;
        break;
    }
    case 5:
    {
        SetDimValue = MID_BR;
        DemoState++;
        break;
    }
    case 6:
    {
        SetDimValue = MAX_BR;
        DemoState++;
        break;
    }
    case 7:
    {
        SetDimValue = MID_BR;
        DemoState++;
        break;
    }

    default:
    {
        break;
    }
}
}
```

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, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL 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, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, nanoWatt XLP, Omniclient Code Generation, PICC, PICC-18, PICkit, PICDEM, PICDEM.net, PICTail, PIC³² logo, REAL ICE, rFLAB, Select Mode, Total Endurance, TSHARC, 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.

© 2009, 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 design centers in California and India. 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

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

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

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

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

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

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 - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

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

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

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: 886-3-6578-300
Fax: 886-3-6578-370

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