A Digital Constant Current Power LED Driver

Author: *Stephen Bowling*
Microchip Technology Inc.

INTRODUCTION

This document describes a power LED driver solution using the PIC12HV615 microcontroller (MCU). The PIC12HV615 is an 8-pin MCU with many integrated analog features. The LED driver circuit is a buck (step-down) solution and the circuit presented here can operate from most any input voltage source as long as it exceeds the forward voltage of the LEDs to be driven.

A proportional-integral (PI) controller algorithm is used to regulate the LED current to a constant value. The PI controller is executed at a rate of 976 Hz, leaving plenty of CPU time available for other tasks. Although this sample rate would provide inadequate control response for most power supply applications, it works well for LED applications because the LED presents a constant load to the power stage. Therefore, the controller does not need to make frequent adjustments.

The LED current is sampled using a resistor in series with the source of the MOSFET in the buck circuit and amplified using a single op amp. The LED current is sampled using one of the available ADC inputs on the PIC12HV615. The Enhanced Capture Compare PWM (ECCP) module of the MCU is used in PWM mode to drive the buck circuit. Since the MCU has an internal voltage regulator and 8 MHz oscillator, very few external components are required to complete the circuit.

CIRCUIT DESCRIPTION

A detailed schematic of the circuit is provided in Figure B-1. A buck topology is used with the LED load referenced to the input voltage rail. The buck topology allows the LED to be in series with the inductor at all times and limit the current ripple. Therefore, an output capacitor is not required as it would be for a voltage regulator circuit. However, an output capacitor is shown in the schematic and can optionally be used to lower the required value of the inductor.

In order to select an inductor value for the buck circuit, we need to know some basic operating parameters. For calculation purposes, we will assume that the circuit will drive a single 1W power LED. The typical drive current for this type of LED would be 350 mA and a typical forward voltage would be 3.5V. Secondly, we will assume that the circuit input voltage is 12V.

The inductor value will be chosen to allow a maximum current ripple of +/- 20%. The eye will not be able to perceive this current ripple since the switching frequency is much faster than can be detected. Considering this fact, you might be tempted to further increase the allowable current ripple to reduce the inductor value. However, the +/- 20% limit is a rule of thumb to minimize efficiency losses in the LED. You will want to check the LED manufacturer's data sheet to verify the maximum peak current.

The switching frequency of the buck regulator is chosen to be 125 kHz. This frequency was chosen based on the selected operating frequency of the MCU and the resulting control resolution that can be obtained from the hardware PWM module. When operated from the internal 8 MHz RC oscillator, the ECCP module can provide 6 bits (64 steps) of duty cycle resolution at 125 kHz. If this control resolution is not sufficient, the PWM frequency can be lowered or the MCU can be operated with an external oscillator circuit to provide a higher operating frequency.

A summary of the operating parameters used for design calculations is provided in Table 1 below.

TABLE 1: BUCK REGULATOR OPERATING PARAMETERS

Parameter	Value
Input Voltage, V_{IN}	12V
LED Forward Voltage, V_f	3.5V
LED Current, I_f	0.35A
Current Ripple, ΔI	+/- 20%
Switching Frequency, f_{sw}	125 kHz

EQUATION 1: INDUCTOR VALUE

$$L = \frac{(V_{in} - V_f)}{\Delta I} \bullet t_{on} = \frac{V_f}{\Delta I} \bullet t_{off}$$

EQUATION 2: INDUCTOR CHARGE TIME

$$t_{on} = \frac{1}{f_{sw}} \bullet \frac{V_f}{V_{in}}$$

The inductor value, L, can be calculated using Equation 1 and Equation 2. The inductor value should be at least 128 μH to achieve the desired current maximum current ripple. A standard inductor value of 150 μH was chosen for the design based on this calculation.

A 0.56 ohm sensing resistor is used to measure the LED current. The sense resistor is placed in series with the source of the power MOSFET. The resistor value was chosen to provide low-power dissipation, but this value also provides a very low output voltage at typical operating currents.

A single op amp is used in a non-inverting amplifier configuration to increase the current feedback signal amplitude. A larger resistor could be used to eliminate the gain stage, but this would result in significant power dissipation. More important, the resistor power dissipation would reduce the overall efficiency of the LED driver.

The feedback components of the op amp are selected to provide a gain of 11. This gain value will provide a nominal feedback voltage of 2V at 350 mA drive current and 4V at 700 mA drive current.

Because of the location of the sense resistor in the circuit, the current feedback signal is present only when the MOSFET is turned on and charging the inductor. Since the feedback signal is discontinuous, a simple peak hold circuit using a Shottky diode is used at the output of the current sense amplifier. The component values of the peak hold circuit are chosen to provide a low pass filter with a very low cutoff frequency, which averages the current feedback signal. To ensure that the current can be properly regulated using the software PI controller, the cutoff frequency is chosen to be 1/10th the value of the current sample rate.

The Enhanced Capture Compare PWM (ECCP) module on the PIC12HV615 is used to control the buck regulator MOSFET. The output of the current sense amplifier and peak hold circuit is connected to input AN0 of the PIC12HV615 ADC.

A dropping resistor is used between the input voltage rail of the circuit delivered by J1 and the V_{DD} pin of the PIC12HV615. This resistor must be sized to provide the operating current for the internal shunt regulator, plus the expected operating current of the device. Refer to the PIC12HV615 (DS41302) data sheet to determine the value of this resistor.

Connector J2 allows the PIC12HV615 device to be programmed using a MPLAB® ICD 2 or PICkit™ 2 device programmer. Jumpers J3 and J4 must be removed during device programming. (See schematic diagram, Figure B-1.)

SOFTWARE DESCRIPTION

The CCS PICC™ C compiler was used to develop the source code for this application. The PICC C compiler has a variety of built-in library functions that allow the user to rapidly configure the I/O pins and peripherals of the MCU.

The Setup() function configures Timer2, the ECCP module, and the ADC. The Timer2 module is used along with the ECCP module to produce a PWM signal. Timer2 also provides software interrupts that schedule ADC conversions and calculation of the PI control algorithm.

The Timer2 output postscaler is used to reduce the frequency of interrupts to the CPU. With the 1:16 postscaler option the interrupt frequency is 7812 Hz. A count variable, `Count1msec`, is incremented each time a Timer2 interrupt occurs. An ADC conversion is performed and the PI algorithm is executed after every eight Timer2 interrupts. The PI calculation period is 1.025 msec, or 976 Hz. The ADC conversion and PI calculation is scheduled by setting a flag bit, `Event1msec`, in the ISR. The flag is responded to in the main software loop to avoid consuming time in the ISR.

A `typedef` is used to create a data type for the PI data structure. A pointer to the data structure is passed to the actual PI function. This implementation allows multiple control loops to be created and called in the application, if desired. The data structure contains all the variables associated with a particular control loop.

The PICC C compiler defines the long data type as a 16-bit integer and the int data type as an 8-bit integer. Most values in the PI data structure are declared as long values to provide adequate calculation resolution. The gain values and output limit can be declared as 8-bit values to conserve RAM space.

EXAMPLE 1: CODE EXAMPLE

```
typedef struct {
    unsigned char       Kp;
    unsigned char       Ki;
    unsigned char       OutMax;
    signed long         Setpoint;
    signed long         Feedback;
    signed long         Error;
    signed long         Integral;
    signed long         Output;
    int1                Saturated;
    } tPIParams;
```

© 2007 Microchip Technology Inc.

The equations required to execute a PI controller are shown in Equation 3. In this control system, the ADC conversion provides the amount of measured LED current. The PI controller calculates an error based on this measured feedback value and the chosen operating current. The PI controller output is then used to set the duty cycle for the buck converter. The operation and behavior of the PI controller is described in further detail below.

EQUATION 3: PI CONTROLLER

$$Error = Setpoint - Feedback$$

$$Integral = Integral + Error$$

$$Output = \frac{\overbrace{K_P \bullet Error}^{\text{PROPORTIONAL TERM}} + \overbrace{K_I \bullet Integral}^{\text{INTEGRAL TERM}}}{K}$$

First an error term is calculated. This is simply the difference between the desired system output value (Setpoint) and the actual output value (Feedback). In this current regulator application, the ADC will provide a sample value corresponding to the actual output current and a setpoint value will be initialized that represents the desired output current.

If the error term is small enough, then no further calculations are performed. This error windowing function saves a significant amount of CPU bandwidth. Often, little or no change of the PWM duty cycle is required once the power LED reaches a steady state operating point.

The proportional term of the controller multiplies the calculated error by a gain value. The proportional term provides a system response that is proportional to the amount of error. If the output current is very near the desired value, the proportional term of the controller will provide very little response to correct the error. If the output current is much greater or much less than the desired output value, then the proportional term of the controller will produce a very big response to correct the large error.

The goal is to reduce the error to zero and the PI control algorithm cannot do this with just a proportional term. With only proportional control, the output of the PI controller equation will be zero when the error is zero!

The integral part of the controller comes to the rescue and provides an output response that keeps the system at the desired setpoint under steady state conditions.

The integral part of the PI controller works by adding the calculated error to a running sum each time the PI controller is executed. This sum, or integrated error, is multiplied by a gain coefficient to produce the integral response of the controller. Dynamically, this causes a small system error to accumulate over time and produce a larger response. Therefore, the integral part of the controller will correct small steady state errors.

As the error converges to zero, the integral term will converge to a value that produces the correct control response to keep the system at the desired setpoint. As stated earlier, the proportional output of the controller will produce little or no response when the error is close to zero. So, it is the integral term that maintains the DC response that keeps the system operating at the desired setpoint.

The error is not added to the integral term under two conditions. First, the error will not be added if the PI controller output has reached its maximum or minimum allowable value. In this application, the minimum output duty cycle is 0 and the maximum is a duty cycle near 100%. If a maximum or minimum duty cycle limit has been reached, the output of the PI controller is restricted to that value and a flag is set to indicate that the controller is in a saturation condition. The saturation check avoids a condition known as “integral windup”. If the integral term is allowed to accumulate when the controller is saturated, then very erratic results will occur when the controller comes out of saturation.

Secondly, the error value is not added to the integral term when the integral value has reached a limit of +/- 32000. This limit check keeps the integral variable within the limits of a 16-bit signed integer and avoids the use of extra RAM locations.

As shown in the formulas provided in Equation 3, the output calculation for the controller divides the proportional and integral terms by a constant K. K is an integer value that sets the overall gain of the controller and also sets the resolution of the K_p and K_i gain values. This constant allows fractional gains to be utilized.

The PI controller can be tuned experimentally by finding the maximum K_p value that can be used without oscillations. The output of the ECCP module or the output of the current sensing circuit is observed as K_p is adjusted to confirm that no oscillations are present. The K_i gain value is set to 0 at this time, so the current output will not increase to the desired setpoint.

Once a suitable for K_p has been found, K_i can be slowly increased to find a value that will make the system error decrease to 0. Usually, the K_i value is set to a value much less than the value of K_p. When K_i is too small, the system will take longer to reach the desired setpoint. When K_i is too big, the system will over-correct and oscillations will occur.

GOING FURTHER...

This application note presents only the digital current control software required for the LED. At a minimum, the PIC12HV615 must monitor the LED current. In addition, other signals can be monitored depending on the application requirements. The schematic shown in Figure B-1 has extra MCU inputs available that could be used to monitor the input bus voltage, a temperature sensor, or a push-button. In particular, it might be desirable to monitor the temperature of the power LED with a temperature sensor. The digital control loop can then automatically reduce the current drive level to protect the lifetime of the power LED.

In particular, it might be desirable to monitor the temperature of the power LED with a temperature sensor. LEDs have very long lifetimes, but only if the temperature is kept within limits. It is not always possible to ensure that the mechanical installation of the LED will provide enough heat sinking to keep the LED temperature within specifications.

With electronic thermal management, the LED drive current can be reduced if the maximum LED temperature is exceeded. Therefore, the LED lighting system will always produce the maximum amount of light allowed by the mechanical installation and the lifetime of the LED will be maximized.

HARDWARE RESOURCES

This application uses 633 of the 1024 available instruction words in the Flash program memory. Dynamically, the application uses up to 44 bytes of the 64 bytes available.

CONCLUSION

This application note has shown how a PIC12HV615 MCU can be used to implement a digital current control loop for power LEDs. The integrated peripherals of the PIC12HV615 allow a low-cost implementation with room for additional features. These features include dimming of the LED, temperature regulation, and communication functions.

© 2007 Microchip Technology Inc.

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.

APPENDIX A: SOURCE CODE

```
#include <12F615.h>
#include "pi.h"

#fuses INTRC_IO,NOPROTECT
#fuses IOSC8,MCLR,BROWNOUT_NOSL,NOWDT
#fuses PUT,BROWNOUT

#device ADC = 10                   // ADC library will return 10-bit ADC result

// Variable declarations
tPIParams    Current;              // Data Structure for PI controller. See pi.h

int          Count1msec;           // Software count for scheduling PI calculation
int1         Event1msec;           // Flag to indicate time for PI calculation

//Function Prototypes
void Setup(void);

//------------------------------------------------------------------------
//  The Timer2 ISR will be invoked at a 7.8 kHz rate with Fosc = 8MHz.
//  A count variable is incremented in the ISR and a flag is set every
//  16 counts that will trigger the PI calculations.  The PI controller
//  is therefore executed at 7.8kHz/8 = 976 Hz.
//------------------------------------------------------------------------
#INT_TIMER2
void timer2_isr()
{
   Count++;
   if(Count == 8)
   {
   Count1msec = 0;
   Event1msec = 1;

   }
}
```

```
//---------------------------------------------------------------------
//  The device code begins here.  After the peripherals and variables
//  are initialized, the StartPI flag is polled in an endless loop.
//---------------------------------------------------------------------

void main()
{
       // Setup peripheral functions
       Setup();

       // Setup gains and initial values for the PI controller
       // Note:  The value of Current.Setpoint will depend on the
       // gain of the current feedback circuit.
       Current.Kp = 35;
       Current.Ki = 2;
       Current.OutMax = 50;
       Current.Setpoint = 100;
       Current.Feedback = 0;
       Current.Integral = 0;

       while(1)
       {
           // Poll for flag
           if(Event1msec)
           {
           // Read ADC to get current feedback value.  The present
           // sample is averaged with the previous sample.
           Current.Feedback += read_adc();
           Current.Feedback /= 2;
           // Calculate the PI controller using variables in the
           // 'Current' data structure.
           CalcPI(&Current);
           // Write the calculated PWM duty cycle to the CCP module.
           set_pwm1_duty(Current.Output);
           // Clear the flag
           Event1msec = 0;
           }
       }
}

//---------------------------------------------------------------------

void Setup(void)
{
       set_tris_a(0xfb);                    // GP2 is output for PWM
       setup_ccp1(CCP_PWM);                 // Enable CCP module for PWM
       set_pwm1_duty(0);                    // Set initial PWM duty cycle
       setup_timer_2(T2_DIV_BY_1,0x0f,16);// Enable TMR2 with /16 postcaler

       // Note: AN0, AN1, and AN3 are enabled as analog inputs.  Although
       // not required, the AN1 and AN3 inputs can be used to monitor
       // parameters such as temperature, bus voltage, or LED forward
       // voltage.

       setup_adc_ports(sAN0 | sAN1 | sAN3);// AN0,AN1,AN3 analog inputs
       setup_adc(ADC_CLOCK_INTERNAL);       // enable ADC - internal clk
       set_adc_channel(0);                  // input channel AN0

       enable_interrupts(INT_TIMER2);       // Enable timer interrupts
       enable_interrupts(GLOBAL);           // Enable CPU interrupts


}
```

© 2007 Microchip Technology Inc.

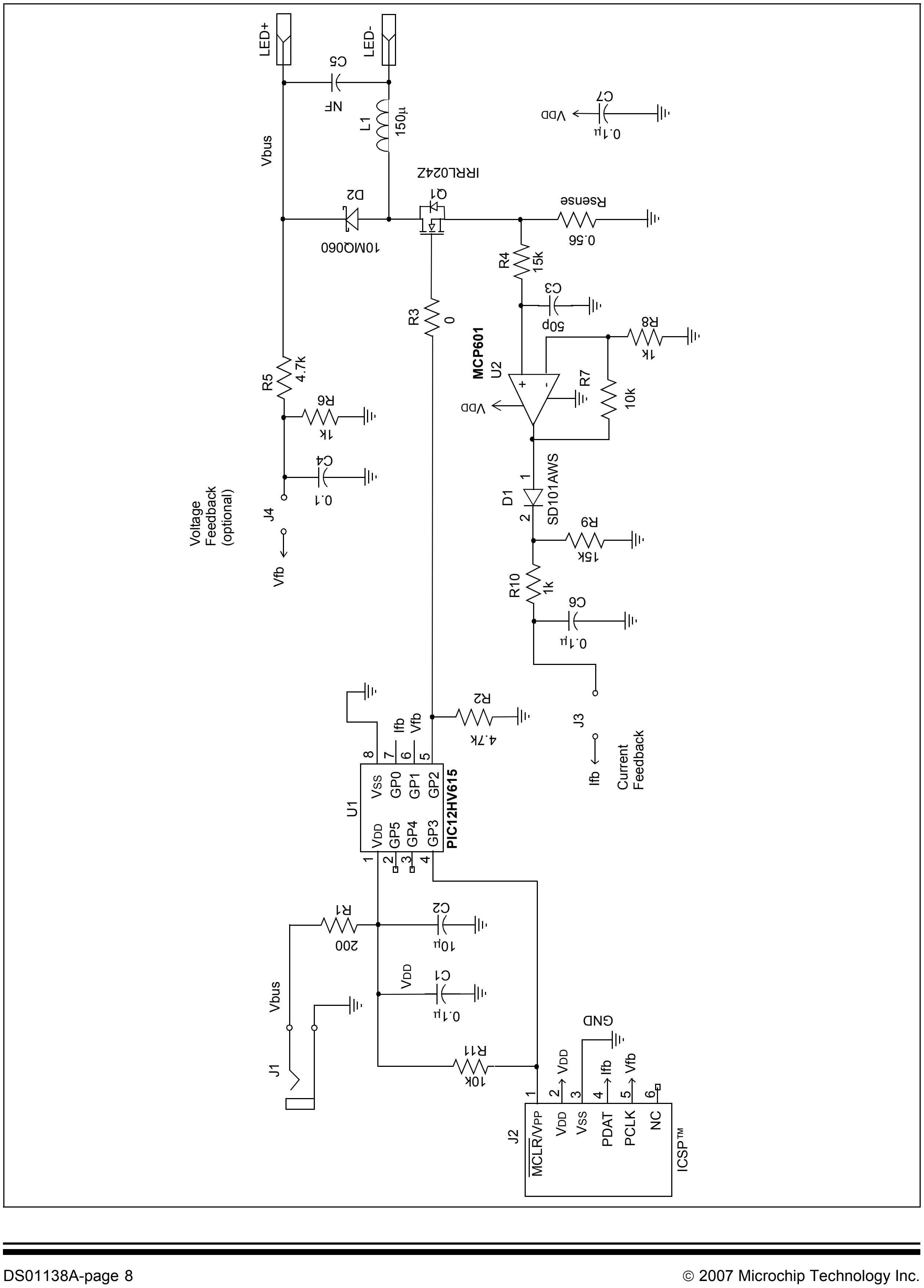
```
void CalcPI(tPIParams *PIdata)
{
     PIdata->Error = PIdata->Setpoint - PIdata->Feedback;
     // Deadband -- If the magnitude of the error is 2 or less,
     // then don't calculate the PI routine at all.  This saves
     // processor time and avoids oscillation problems.
     if((PIdata->Error > 2) || (PIdata->Error < -2))
     {
         // If the PI controller is saturated, then don't do any
         // accumulation of the integral.
         if(PIdata->Saturated == 0)
         {
             // Do some boundary checking on the integral value
             // to avoid overflow.  If the integral value is near
             // the limits, then we won't do the accumulation.
             if(PIData->Error > 0)
             {
                 if(PIData->Integral < 32000)
                 PIdata->Integral += PIdata->Error;
             }
             else
             {
                 if(PIData->Integral > -32000)
                 PIdata->Integral += PIdata->Error;
             }
         }
         // Now, calculate the actual PI function here.
         PIdata->Output = (PIdata->Error * PIData->Kp \
         + PIdata->Integral * PIdata->Ki)/256;

         // Perform boundary checks on the PI controller output.  If the
         // output limits are exceeded, then set output to the limit
         // and set flag.
         if(PIdata->Output > PIdata->OutMax)
         {
                 PIdata->Saturated = 1;
                 PIdata->Output = PIdata->OutMax;
         }

         else if(PIdata->Output < 0)
         {
                 PIdata->Saturated = 1;
                 PIdata->Output = 0;
         }
         else PIdata->Saturated = 0;
     }
}
```

APPENDIX B: LED DRIVER DEMO SCHEMATIC

FIGURE B-1: SCHEMATIC



© 2007 Microchip Technology Inc.

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, 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, 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, dsSPEAK, 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, 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 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

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 - 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 - 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: 86-29-8833-7252
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 - 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: 886-3-572-9526
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

10/05/07

© 2007 Microchip Technology Inc.