INTRODUCTION

The PIC12CXXX family of devices adds a new twist to the 8-bit microcontroller market by introducing for the first time fully functional microcontrollers in an eight pin package. These parts are not stripped down versions of their larger brethren, they add features in a package smaller than available ever before for microcontrollers. Using the familiar 12-bit opcode width of the PIC16C5X family with the same TMR0 module, Device Reset Timer, and WatchDog Timer (WDT), the PIC12C5XX family adds an internal 4MHz oscillator main clock, serial programming, wake-up on change, user selectable weak pullups, and multiplexing of the MCLR, T0CKI, OSC1, and OSC2 pins.

This combination of familiar and new features in a compact package gives the designer unprecedented flexibility to produce designs which are much cheaper and smaller than ever before possible, and allows the replacement of even mundane devices like timers and discrete components economically. This reference note describes an application where the use of a microcontroller was not previously economically feasible for any but the highest end products: lamp dimming.

Information contained in this publication is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Inc. with respect to the accuracy or use of such information, or infringement of patents arising from such use or otherwise. It is the responsibility of each user to ensure that each UPS is adequately designed, safe, and compatible with all conditions encountered during its use. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals", must be validated for each customer application by the customer’s technical experts. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

© 1997 Microchip Technology Inc.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware overview</td>
<td>3</td>
</tr>
<tr>
<td>Software overview</td>
<td>5</td>
</tr>
<tr>
<td>Design Modifications</td>
<td>5</td>
</tr>
<tr>
<td>APPENDIX A: SYSTEM SPECIFICATIONS</td>
<td>9</td>
</tr>
<tr>
<td>APPENDIX B: BILL OF MATERIALS</td>
<td>9</td>
</tr>
<tr>
<td>APPENDIX C: SOFTWARE PROGRAM</td>
<td>11</td>
</tr>
<tr>
<td>APPENDIX D: DIM508.LST FILE</td>
<td>17</td>
</tr>
</tbody>
</table>

## ACKNOWLEDGMENTS

**Project Lead Engineer:**
Scott Fink

**System and Code Development:**
Scott Fink
HARDWARE OVERVIEW

Lamp dimming using a TRIAC

Logic level TRIACS are a relatively new introduction. They allow a microcontroller to directly drive (through a current limiting resistor) the gate of a TRIAC.

TRIACs can be used to control the brightness of a lamp by switching the AC power on part-way through each half wave (Figure 2 and Figure 3). By controlling where the TRIAC is "fired" during the power-line cycle, the microcontroller can control the average voltage across the filament of the lamp, and thus the brightness.

The TRIAC used for this application is able to handle lamps up to a maximum of 100W.

R9 is connected to the "hot" lead of the AC power line and to pin GP4. The ESD protection diodes of the input structure of the GPIO allows this connection without damage (see Figure 1). When the voltage on the AC power line is positive, the protection diode from the input to VDD is forward biased, and the input buffer will see approximately VDD+0.7 volts and the software will read the pin as high. When the voltage on the line is negative, the protection diode from VSS to the input pin is forward biased, and the input buffer sees approximately VSS-0.7 volts and the software will read the pin as low. By polling GP4 for a change in state, the software can detect a zero crossing.

Since there is no transformer for power-line isolation, the user must be very careful and assess the risks from line-transients in his application location. The varistor (RV1) will add some protection.

The Power Supply

The power supply used for this design uses only discrete components and has no transformer or voltage regulator making it extremely low cost. It has been designed to handle either 60Hz or 50Hz input power, 120V nominal line voltage.

The caveat to this low cost power supply is that it can not provide large currents, and the user must take care not to overload it.

FIGURE 1: ZERO CROSSING DETECTION
FIGURE 2: WAVEFORMS

Line Voltage

Voltage at Hot output lead, near full bright

Voltage at PIC12C508, GP2

FIGURE 3: OUTPUT VOLTAGE OF FULL-WAVE PHASE CONTROL

Output Voltage vs. Conduction Angle φ
SOFTWARE OVERVIEW

The software is written in ‘C’ using MPLABC, V1.21. There is only a main function and one function called Buttoncheck.

Main Function
Initialization

The main function begins by initializing all of the RAM registers used, and setting the TRIS register so that the zero crossing sense, dim button, and bright button pins are set as inputs, and so that the TRIAC drive pin is set to be an input. The OPTION register is set to assign the prescaler to the timer with a ratio of 1:64, timer to increment on internal clock, and enable the weak pull-up resistors on GP0, GP1, and GP3.

The next statement sets the output latch of GP2 (the output to the TRIAC) high. Note that this statement only sets the output latch high. Since it is set to be an input at this point, the pin will be at high-impedance.

Because the internal RC oscillator of the PIC12C508 can vary with temperature and supply voltage (the Vdd supply should be fairly constant at 5V), the program constantly keeps track of the total Timer0 count of each half cycle of the AC line. If this were not done and the count was too long for maximum dimming, the TRIAC would be fired shortly after the next half-cycle had begun and actually cause the lamp to be on full bright instead of full dim. The rest of the code before entering the main program loop synchronizes the Timer0 count with the line voltage so that the line frequency/Timer0 count is known.

Main Program Loop

The main program loop counts the line cycles and calls Buttoncheck after DelayCnt cycles. If it is not time to call Buttoncheck, two short routines are run, one for the positive and one for the negative half-cycle of the AC line. The routines are identical except for the line polarity checking, so only one will be described.

The line phase is checked to see if the next half-cycle has already begun. If it has, Maxdim is incremented and a wait state is initiated to re-synch with the line voltage. If it hasn’t, the program waits for the line voltage to cross zero and when it does, resets Maxdim to match the half-cycle time. If the selected on-percentage is selected to be greater than full dim, it is reset to give full dim.

The timer is set to time out when the TRIAC should be fired for the desired brightness. The program then goes into a loop to wait for either the timer to roll over to zero, or for the AC line half cycle to expire.

The TRIAC is then fired by setting the pin connected to it’s gate to be an output (the output latch was already set high) to supply current into the gate. A short delay is initiated to widen the firing pulse before again setting the pin to a high-impedance. The TRIAC will shut off when the AC line voltage next crosses zero.

Buttoncheck Subroutine

This subroutine checks for presses of the BRT and DIM buttons and increments or decrements Percenton based on their states.

If both buttons are pressed and the lamp is not off, it is turned off. If it is already off, it is turned on full bright.

In addition to taking commands from the buttons, a test function is built in to this routine. The test mode is entered by holding both buttons, and then releasing and pressing DIM again. The test will run for 255 cycles or until the DIM button is pressed. The test runs in a cycle of brightening to full bright, dimming to full dim and then flashing full bright twice.

After the section of Buttoncheck where the test cycling is done if the program is in test mode, the program checks the buttons for the sequence to enter test mode, and looks for a both pressed for instant on or off. Following this code is the single button up and down commands with checking for more than full bright and less than full dim.

DESIGN MODIFICATIONS

This reference design will work for many applications without modification. It is anticipated that customers may want to customize its functionality, however, and this section offers suggestions for modification:

• The software was written for a 60Hz line frequency and might work on a 50HZ line, but has not been tested at anything but 60Hz.
• Modify the circuit to use a single button. For this modification, pressing the button would turn the lamp on and off, and if held, would gradually brighten the lamp to full bright, then gradually dim to full dim. The brightness would stay at whatever level it was at when the button was released.
• Add a light level sensor such that if full darkness was sensed when the button was pressed, the lamp would gradually brighten to avoid shocking eyes adjusted for darkness.
• Add a sensor to automatically switch the lamp on and off based on the room occupancy.
• Use the two available pins to add a serial bus for control from remote computer.
• Add a “Halloween” mode that would flash the lamp at random times for a short period to simulate spooky lightning and such.
• Add a photo sensor to maintain a given brightness level in a room depending on ambient light.
FIGURE 4: SOFTWARE FLOWCHART, MAIN PROGRAM LOOP

Start
- Initialize Variables
- Sync to AC Powerline
  - Time to check buttons?
    - Yes: Call ButtonPress
    - No
      - Line Already High?
        - Yes: Increment Maxdim and resync
        - No
          - Wait for Zero Crossing
            - Compensate Maxdim
  - First Pass?
    - Yes: Set for fulldim
    - No
      - Initialize TMR0
        - Timer Rollover or Zero Cross?
          - Yes: Fire TRIAC
          - No
            - Line Already Low?
              - Yes: Increment Maxdim and resync
              - No
                - Wait for Zero Crossing
                  - Compensate Maxdim
                    - Initialize TMR0
                      - Timer Rollover or Zero Cross?
                        - Yes
                        - No
                          - Fire TRIAC
FIGURE 5: SOFTWARE FLOWCHART, FUNCTION BUTTONPRESS

Buttonpress

In Test Mode? Yes

No

Both buttons Pressed? Yes

No

PercentOn = Maxdim? Yes

PercentOn = Maxbrt

PercentOn = Maxdim

PercentOn = Maxbrt

Dim Pressed? Yes

No

Modify Percent On

Has Cycle run 255 times? Yes

No

Cancel Test Mode Return

Return

Only DIM Pressed? Yes

No

Decrement PercentOn

Increment PercentOn

Only BRT Pressed? Yes

No

PercentOn >Maxbrt? Yes

No

PercentOn = Maxbrt Return

PercentOn >Maxdim? Yes

No

PercentOn = Maxdim Return

Return

PercentOn = Maxdim

Return

PercentOn = Maxbrt

Return

Modify Percent On

Cancel Test Mode Return

Return
APPENDIX A: SYSTEM SPECIFICATIONS

The following is a list of specifications for the Lamp dimmer:

AC Input: 120 VAC ± 10%, 60Hz ± 3Hz
Output: 100W, resistive load only!

APPENDIX B: BILL OF MATERIALS

<table>
<thead>
<tr>
<th>Description</th>
<th>Qty</th>
<th>Designators</th>
<th>Part #, Manufacturer, Contact #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistor, 1/4 Watt, 47ohm, Axial Lead</td>
<td>1</td>
<td>R1</td>
<td>Generic</td>
</tr>
<tr>
<td>Resistor, 1/4 Watt, 475ohm, Axial Lead</td>
<td>3</td>
<td>R4, R5, R6</td>
<td>Generic</td>
</tr>
<tr>
<td>Resistor, 1/4 Watt, 1Mohm, Axial Lead</td>
<td>1</td>
<td>R2</td>
<td>Generic</td>
</tr>
<tr>
<td>Resistor, 1/4 Watt, 20Mohm, Axial Lead</td>
<td>1</td>
<td>R3</td>
<td>Generic</td>
</tr>
<tr>
<td>8 Pin, 8-Bit, CMOS, Microcontroller</td>
<td>1</td>
<td>U1</td>
<td>12C508, Microchip Technology, Inc. (602) 786-7200</td>
</tr>
<tr>
<td>Logic Triac, TO-202AB, 400V</td>
<td>1</td>
<td>Q1</td>
<td>L4004F51, Teccor Electronics Inc. (214) 580-1515</td>
</tr>
<tr>
<td>Zener Diode, 5.1V, DO-35</td>
<td>1</td>
<td>D3</td>
<td>1N5231BCT, Diodes Incorporated/Digi-Key (800) 344-4539</td>
</tr>
<tr>
<td>Diode</td>
<td>2</td>
<td>D1, D2</td>
<td>1N4001, Generic</td>
</tr>
<tr>
<td>Keyswitch, Momentary PCB Mount</td>
<td>2</td>
<td>S1, S2</td>
<td>BF3-1000, Omron (847) 843-7900</td>
</tr>
<tr>
<td>ZNR Transient/Surge Absorbers, 1250A Surge, 300VDC, 230VAC</td>
<td>1</td>
<td>RV1</td>
<td>ERZ-D07D361, Panasonic (206) 395-7343</td>
</tr>
<tr>
<td>Aluminum Electrolytic Capacitor, 220uF, 35V</td>
<td>1</td>
<td>C1</td>
<td>ECE-A1VU221, Panasonic (206) 395-7343</td>
</tr>
<tr>
<td>Axial Ceramic Capacitor, 0.01uF, 50V</td>
<td>1</td>
<td>C2</td>
<td>A103Z15Z5UFVWVA, Philips (620) 820-2225</td>
</tr>
<tr>
<td>Polyester &amp; Foil Capacitor, 0.1uF, 200V</td>
<td>1</td>
<td>C3</td>
<td>ECQ-M2104KZ, Panasonic (206) 395-7343</td>
</tr>
</tbody>
</table>

TABLE 1: BUTTON FUNCTIONS

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRT</td>
<td>Brighten</td>
</tr>
<tr>
<td>DIM</td>
<td>Dim</td>
</tr>
<tr>
<td>Hold DIM, Press BRT</td>
<td>If off: turn full on, if on: turn off</td>
</tr>
<tr>
<td>Hold BRT, Press, release, and press DIM again. To exit test mode, press DIM.</td>
<td>Enter test/demo mode</td>
</tr>
</tbody>
</table>
FIGURE 6: CIRCUIT DIAGRAM

DANGER
Electrocution Hazard
APPENDIX C: SOFTWARE PROGRAM

```c
#pragma option v;
#include <12C508.h>
/***************/
/* DIMMER.C */
/* Lamp dimmer for the 12C508.
/* This program uses the internal 4MHz oscillator
/* To drive TRIAC, the output is taken high
/* or put in high-impedance(open drain) to release it
/* NOTE: This program is designed to work with a 60Hz
/* line frequency, it must be modified if used
/* on a 50Hz AC line.
/*
/* GPIO<0> = Dim button
/* GPIO<1> = No Connect
/* GPIO<2> = Output to TRIAC
/* GPIO<3> = Bright button
/* GPIO<4> = Zero Crossing sense input
/* GPIO<5> = No Connect
/***************/
#define Brtbut GPIO.0 //Brighten button
#define Output GPIO.2 //Output to TRIAC
#define Dimbut GPIO.3 //Dim button
#define LineInput GPIO.4 //AC line zero crossing sense

void Buttoncheck(void); //Button check routine

unsigned int PercentOn, Maxdim; //Global variables
unsigned int TestCheck, Outcount, TestCount;
unsigned int DelayCnt;
unsigned int LastBoth, FirstPass;
const Maxbrt = 0xFD, NotInTest = 3;

void main()
{
  PercentOn = 0xD0; //On Period
  Maxdim = 0x70; //Value of Maximum dimming
  TestCheck = 0; //Test mode check counter
  Outcount = 0; //Counter for test mode exit
  TestCount = 0; //Test mode counter
  DelayCnt = NotInTest; //Delay count
  LastBoth = 0; //Both buttons pressed last time flag
  FirstPass = 1; //Indicate power-up
  Count = 0; //General counter
  for(Count = 0; Count < 60; Count++) //Allow power supply to stabilize
    
      while(LineInput == 1);
    while(LineInput == 0);

  WREG = 0x85;
  #asm ( OPTION); //1:64 tnr0 prescaler, pullups enabled
  WREG = 0x1D;
  #asm ( TRIS GPIO); //Set up I/O

  GPIO = 0x04; //Set TRIAC output latch high
  while(LineInput == 1) //Synch to line phase
    CLRWDFT;
  TMRO = PercentOn; //Get Delay time
  while(TMRO >= 3 && LineInput == 0) //Delay to enter main at proper point
    CLRWDFT;
  while[1] //Stay in this loop
```

---
© 1997 Microchip Technology Inc.
Count = 0;
while (Count++ < DelayCnt)  //Check for button press every
    DelayCnt zero crossings
{  
    if(LineInput == 1)  //Check for AC line already high
    {  
        Maxdim += 5;  //If so, increment Maxdim
        while(LineInput == 1);  // and re-sync with line
        while(LineInput == 0)
        
            CLRWDT;
    }  
    else
    {  
        while(LineInput == 0)  //Wait for zero crossing
            CLRWDT;
        Maxdim = PercentOn - TMR0 + 2;  //Compensate full dim value for line
        // frequency vs osc. speed
    }
    if(FirstPass == 1)  //If first pass, go to full dim
    {  
        FirstPass = 0;
        PercentOn = Maxdim;
    }
    if(PercentOn < Maxdim)  //If maxdim moved, fix brightness
    {  
        PercentOn = Maxdim;
        TMR0 = PercentOn;
        while(TMR0 >= 3 && LineInput == 1)  //Delay TRIAC turn on (wait for Counter rollover)
            CLRWDT;
        GPIO = 0x04;  //Set TRIAC output latch high
        WREG = 0x19;
        #asm ( TRIS GPIO);  //Fire TRIAC
        NOP;  //Delay for TRIAC fire pulse
        NOP;
        NOP;
        NOP;
        NOP;
        NOP;
        NOP;
        WREG = 0x1D;
        #asm ( TRIS GPIO);  //Release TRIAC fire Signal
        CLRWDT;
    }
    if(LineInput == 0)  //Check for AC line already low
    {  
        Maxdim += 5;  //If so, increment Maxdim
        while(LineInput == 0);  // and re-sync with line
        while(LineInput == 1)
            CLRWDT;
    }  
    else
    {  
        while(LineInput==1)  //Wait for zero crossing
            CLRWDT;
        Maxdim = PercentOn - TMR0 + 2;  //Compensate full dim value for line
        // frequency vs osc. speed
    }
    if(PercentOn < Maxdim)  //If maxdim moved, fix brightness
    {  
        PercentOn = Maxdim;
        TMR0 = PercentOn;
        while(TMR0 >= 3 && LineInput == 0)  //Delay TRIAC turn on
            CLRWDT;
        GPIO = 0x04;  //Set TRIAC output latch high
        WREG = 0x19;
        #asm ( TRIS GPIO);  //Fire TRIAC
NOP;  //Delay for TRIAC fire pulse
NOP;
NOP;
NOP;
NOP;
NOP;
NOP;
WREG = 0x1D;
asm { TRIS GPIO};  //Release TRIAC fire signal
CLRWDT;
}
Buttoncheck();  //Check for button press
}

/********************************************************
/* ButtonCheck */
/* */
/* This subroutine checks for presses on the BRT and DIM */
/* buttons and increments or decrements PercentOn. */
/* */
/* If both buttons are pressed and the lamp */
/* is not off, it is turned off, if off, it is set to */
/* to max bright. */
/* */
/* In addition, a test function is built in. If both */
/* buttons are pressed, the dim let go and then pressed */
/* again, test mode is entered. If dim is pressed */
/* (alone), the program goes to normal operation at max */
/* dim. The test mode brightens to full bright, dims to */
/* full dim, flashes full bright twice, and repeats. */
/********************************************************

void Buttoncheck()
{
  NOP;  /*Bugfix for MPLABC V1.10*/
  if(TestCheck == 3)  /*Check test mode flag*/
  {
    DelayCnt = 2;  /*Reset the delay count*/
    if(Brtbut && !Dimbut)  /*If Dimbutton pressed, exit test mode*/
    {
      TestCheck = 0;  /*Clear Test mode flag*/
      DelayCnt = 5;
      return;
    }
    if(TestCount == 0)  /*Ramp up to full dim*/
    {
      if(++PercentOn > Maxbrt)  /*Check for full bright*/
      {
        PercentOn = Maxbrt;
        +TestCount;
        return;
      }
      else
        return;
    }
    if(TestCount == 1)  /*Ramp down to full dim*/
    {
      if(--PercentOn <= Maxdim)  /*Check for full dim*/
      {
        PercentOn = Maxbrt;
        +TestCount;
        return;
      }
    }
    else
      return;
  }
  while(TestCount++ < 5)  /*Delay*/
return;
while(TestCount++ < 10) //Turn off for a short period
{
    PercentOn = Maxdim;
    return;
}
while(TestCount++ < 15) //Turn On for a short period
{
    PercentOn = Maxbrt;
    return;
}
while(TestCount++ < 20) //Turn off for a short period
{
    PercentOn = Maxdim;
    return;
}
while(TestCount++ < 25) //Turn on for a short period
{
    PercentOn = Maxbrt;
    return;
}
while(TestCount++ < 30) //Turn off for a short period
{
    PercentOn = Maxdim;
    return;
}
PercentOn = Maxdim;
TestCount = 0; //Reset to beggining of test sequence
if(++Outcount == 255) //Run 255 cycles of test mode
{
    TestCheck = 0; //Clear Test mode flag
    DelayCnt = NotInTest;
    Outcount = 0;
    return;
}
if(TestCheck) //If Test mode not entered quickly,
    if(++Outcount == 0x60) // quit checking
    {
        DelayCnt = NotInTest;
        Outcount = 0;
        TestCheck = 0;
    }
if(!TestCheck && !Brtbut && !Dimbut) //Check bright & dim at same time
    TestCheck = 1; //If both pressed, set to look for next combo
if(TestCheck == 1 && !Brtbut && Dimbut) //Check for only bright button pressed
    TestCheck = 2; //If pressed, set to look for next combo
if(TestCheck == 2 && !Brtbut && !Dimbut) //Check for both pressed again
    {
        TestCheck = 3; //Enable test mode
        TestCount = 0;
        PercentOn = Maxdim;
        Outcount = 0;
    }
if(!Dimbut && !Brtbut) //If both pressed
    {
        if(LastBoth == 0) //Don't flash if held
            {
                LastBoth = 1;
                if(PercentOn == Maxdim) //If full off...
                    PercentOn = Maxbrt; // turn full on...
                else
                    PercentOn = Maxdim; // otherwise turn off
            }
    }
else
    LastBoth = 0;
if(!Brtbut && Dimbut) //Check for brighten cmd
    PercentOn ++;
if(Brtbut && !Dimbut) //Check for dim cmd
    PercentOn --;
if(PercentOn > Maxbrt) //If greater than full bright
    PercentOn = Maxbrt;
if(PercentOn < Maxdim) //If less than full dim
    PercentOn = Maxdim;
}
#pragma option v;
#include <12C508.h>
#ifndef _12C508_H
/*
PIC12C508 Standard Header File, Version 1.02
(c) Copyright 1996 Microchip Technology, Inc., Byte Craft Limited
RAM locations reserved for temporary variables: 8x07
*/
#pragma option +l;
#endif

/********************************************************/
/* DIMMER.C               */
/*                 */
/* Lamp dimmer for the 12C508.          */
/* This program uses the internal 4MHz oscillator   */
/* To drive TRIAC, the output is taken high     */
/* or put in high-impedance(open drain) to release it*/
/* */
/* NOTE: This program is designed to work with a 60Hz * */
/* line frequency, it must be modified if used */
/* on a 50Hz AC line.   */
/* */
/* */
/* GPIO<0> = Dim button */
/* GPIO<1> = No Connect */
/* GPIO<2> = Output to TRIAC */
/* GPIO<3> = Bright Button */
/* GPIO<4> = Zero Crossing sense input */
/* GPIO<5> = No Connect */
/********************************************************/

#define Brtbut GPIO.0            //Brighten button
#define Output GPIO.2            //Output to TRIAC
#define Dimbut GPIO.3            //Dim button
#define LineInput GPIO.4          //AC line zero crossing sense

void Buttoncheck(void);        //Button check routine

unsigned int PercentOn, Maxdim;  //Global variables
unsigned int TestCheck, Outcount, TestCount;
unsigned int DelayCnt;
unsigned int LastBoth, FirstPass;
unsigned int Count;
const Maxbrt = 0xFD, NotInTest = 3;

void main()
{
    MOVLW D0h            PercentOn = 0xD0;       //On Period
    MOVWF 08
    MOVLW 70h            Maxdim = 0x70;         //Value of Maximum dimming
    MOVWF 09
    CLRF 0A TestCheck = 0;                //Test mode check counter
    CLRF 0B Outcount = 0;                 //Counter for test mode exit
    CLRF 0C TestCount = 0;                //Test mode counter
    MOVLW 03h            DelayCnt = NotInTest;   //Delay count
    MOVWF 10
    MOVLW 0Eh            LastBoth = 0;            //Both buttons pressed last time flag
    MOVLW 01h            FirstPass = 1;           //Indicate power-up
    MOVLW 0Fh
    CLRF 0E Count = 0;                    //General counter
    CLRF 10 for(Count = 0; Count < 60; Count++ ) //Allow power supply
                    to stabilize
    GOTO 007
}
0013 0686    BTFS  06,4                      while(LineInput == 1);
0014 0A13    GOTO   0013h
0015 0786    BTFS  06,4                      while(LineInput == 0);
0016 0A15    GOTO   0015h
0017 0004    CLRWD T;                      CLRWD T;
              
0018 0280    INCF  10
0019 0A0F    GOTO   000Fh
              WREG = 0x85;
001A 0C85    MOVLM  85h                    #asm ( OPTION);
001B 0022    OPTION
              WREG = 0x1D;
001C 0C1D    MOVLM  1Dh                    #asm ( TRIS GPIO);
001D 0006    TRIS  PORTB
// __OPTION(0x85);        //1:64 tmr0 prescaler, pullups enabled
// __TRIS(0x1D,GPIO);      //Set up I/O
001E 0C04    MOVLM  04h
001F 0026    MOVWF  06                      GPIO = 0x04;         //Set TRIAC output latch high
while(LineInput == 1)          //Synch to line phase
0020 0786    BTFS  06,4
0021 0A24    GOTO   0024h
0022 0004    CLRWD T                      CLRWD T;
0023 0A20    GOTO   0020h
0024 0208    MOVF   08,W                  TMR0 = PercentOn;          //Get Delay time
0025 0021    MOVWF  01
0026 0C03    MOVLW  03h             while(TMR0 >= 3 && LineInput == 0)     //Delay to enter main
0027 0081    SUBWF  01,W
0028 0703    BTFS  03,0
0029 0A2E    GOTO   003Fh
002A 0686    BTFS  06,4
002B 0A2E    GOTO   003Fh
002C 0004    CLRWD T;
002D 0A26    GOTO   0026h
002E 0070    CLRF   10
002F 0210    MOVF   10,W                 
while (Count++ < DelayCnt)    //Check for button press every
                          DelayCnt zero crossings
{                           
0030 02B0    INCF  10
0031 008D    SUBWF  0D,W
0032 0743    BTFS  03,2
0033 0703    BTFS  03,0
0034 0A05    GOTO   00A5h
              if(LineInput == 1)       //Check for AC line already high
0035 0A00    GOTO   0040h
0036 0786    BTFS  06,4
0037 0C05    MOVLM  05h                  Maxdim += 5;         //If so, increment Maxdim
0038 01E9    ADDWF  09
0039 0686    BTFS  06,4                  while(LineInput == 1);     // and re-sync with line
003A 0A39    GOTO   0039h
003B 0686    BTFS  06,4                  while(LineInput == 0)
003C 0A3F    GOTO   003Fh
003D 030D    CLRWD T;                  CLRWD T;
003E 0A3B    GOTO   003Bh
003F 0A4A    GOTO   004Ah
else
{                           
0040 0686    BTFS  06,4
0041 0A44    GOTO   0044h
0042 0004    CLRWD T;
0043 0A40  GOTO  0040h
0044 0201  MOVF  01,W
           Maxim = PercentOn - TMR0 + 2;  //Compensate full dim
           value for line
0045 0088  SUBWF  08,W
0046 0227  MOVWF  07
0047 0C02  MOVLW  02h
0048 01C7  ADDWF  07,W
0049 0029  MOVWF  09
           // frequency vs osc. speed
004A 0C01  MOVLW  01h
           if(FirstPass == 1)     //If first pass, go to full dim
004B 008F  SUBWF  0F,W
004C 0743  BTFS  03,2
004D 0A51  GOTO  0051h
004E 006F  CLRIF  0F
           FirstPass = 0;
004F 0209  MOVF  09,W
           PercentOn = Maxim;
0050 0028  MOVWF  08
0051 0209  MOVF  09,W
           if(PercentOn < Maxim)   //If maxdim moved, fix brightness
0052 0088  SUBWF  08,W
0053 0743  BTFS  03,2
0054 0603  BTSSC  03,0
0055 0A58  GOTO  0058h
0056 0209  MOVF  09,W
           PercentOn = Maxim;
0057 0028  MOVWF  08
0058 0208  MOVF  08,W
           TMR0 = PercentOn;       //Get delay time
0059 0221  MOVWF  01
005A 0C03  MOVLW  03h           while(TMR0 >= 3 && LineInput == 1) //Delay TRIAC turn on
           (wait for Counter rollover)
005B 0081  SUBWF  01,W
005C 0703  BTSS  03,0
005D 0A62  GOTO  005Ah
005E 0786  BTFSS  06,4
005F 0A62  GOTO  0062h
0060 0004  CLRWDI
0061 0A5A  GOTO  005Ah
0062 0C04  MOVLW  04h
           GPIO = 0x04;        //Set TRIAC output latch high
0063 0026  MOVWF  06
0064 0C19  MOVLW  19h
           WREG = 0x19;
0065 0006  TRIS  PORTB
           // ___TRIS{0x19,GPIO};   //Fire Triac
0066 0000  NOP
0067 0000  NOP
0068 0000  NOP
0069 0000  NOP
006A 0000  NOP
006B 0000  NOP
006C 0000  NOP
           WREG = 0x1D;
006D 0C1D  MOVLW  10h
006E 0006  TRIS  PORTB
           // ___TRIS{0x1D,GPIO};   //Release TRIAC fire signal
006F 0004  CLRWDI
0070 0686  BTSS  06,4
           if(LineInput == 0)     //Check for AC line already low
0071 0A7B  GOTO  007Bh
0072 0000  NOP
0072 0C05  MOVLW  05h
0073 01E9  ADDWF  09
0074 0786  BTFSS  06,4
0075 0A74  GOTO  0074h
           while(LineInput == 0);  // and re-sync with line
           while(LineInput == 1)
while(LineInput==1) //Wait for zero crossing

PercentOn = Maxdim;

if(PercentOn < Maxdim) //If maxim moved, fix brightness

// frequency vs osc. speed

GPIO = 0x04; //Fire TRIAC

//       __TRIS(0x19,GPIO);     //Release TRIAC fire signal

*/

/***************************************************************************/
/* ButtonCheck             */
/*                 */
/* This subroutine checks for presses on the BRT and DIM*/
/* buttons and increments or decrements PercentOn.      */
/*                 */
/* If both buttons are pressed and the lamp */
/* is not off, it is turned off, if off, it is set to */
/* to max bright. */
/*                 */
/* In addition, a test function is built in. If both */
/* buttons are pressed, the dim let go and then pressed */
/* again, test mode is entered. If dim is pressed */
/* (alone), the program goes to normal operation at max */
/* dim. The test mode brightens to full bright, dims to*/
/* full dim, flashes full bright twice, and repeats. */
/********************************************************/

void Buttoncheck()
{
  00A8 0000    NOP                      NOP;             //Bugfix for MPLAB V1.10
  00A9 0C03    MOVLW  03h               if(TestCheck == 3)        //Check test mode flag
  00AA 008A    SUBWF  0A,W
  00AB 0743    BTFSS  03,2
  00AC 0B1B    GOTO   011Bh
  00AD                                  {
  00AD 0C02    MOVLW  02h                  DelayCnt = 2;        //Reset the delay count
  00AE 002D    MOVWF  0D
  00AF 0706    BTFS 06,0           if(Brtbut && !Dimbut)     //If Dimbutton pressed, exit test mode
  00B0 0AB7    GOTO 00B7h
  00B1 0666    BTFS 06,3
  00B2 0AB7    GOTO 00B7h
  00B3
  00B3 006A    CLRF   0A                     TestCheck = 0;     //Clear Test mode flag
  00B4 0C05    MOVLW  05h                    DelayCnt = 5;
  00B5 002D    MOVWF  0D
  00B6 0800    RETLW  00h                    return;
  00B7 022C    MOVF   0C                   if(TestCount == 0)       //Ramp up to full dim
  00B8 0743    BTFS 03,2
  00B9 0AC5    GOTO 00C5h
  00BA
  00BA 02A8    INCF   08                     if(++PercentOn > Maxbrt)    //Check for full bright
  00BB 0CFD    MOVLW 08h
  00BC 0088    SUBWF 08,W
  00BD 0743    BTFS 03,2
  00BE 0703    BTFS 03,0
  00BF 0AC4    GOTO 00C4h
  00C0
  00C0 0CFD    MOVLW 0dh
  00C1 0228    MOVF 08h
  00C2 02AC    INCF 0C
  00C3 0800    RETLW 00h
  00C4 0800    RETLW 00h
  00C5 0C01    MOVLW 01h
  00C6 008C    SUBWF 0C,W
  00C7 0743    BTFS 03,2
  00C8 0AD5    GOTO 00D5h
  00C9                              {
  00C9 0DE8    DECW 08
  00CA 0208    MOVF 08,W
  00CB 0089    SUBWF 09,W
  00CC 0643    BTFS 03,2
  00CD 0AD0    GOTO 00D0h
  00CE 0703    BTFS 03,0
  00CF 0AD4    GOTO 00D4h
  00D0
  00D0 0CFD    MOVLW 0dh
  00D1 0228    MOVF 08

  © 1997 Microchip Technology Inc.  DS40171A-page 21
while(TestCount++ < 5) //Delay

while(TestCount++ < 10) //Turn off for a short period

while(TestCount++ < 15) //Turn On for a short period

while(TestCount++ < 20) //Turn off for a short period

while(TestCount++ < 25) //Turn on for a short period
while(TestCount++ < 30)     //Turn off for a short period
{
    0104 020C    MOVF   0C,W                 
    0105 02AC    INCF   0C
    0106 0027    MOVWF  07
    0107 0C1E    MOVLW  1Eh
    0108 0087    SUBWF  07,W
    0109 0003    BTSC  03,0
    010A 0003    BTFSC  03,0
    010B 0B0E    GOTO   010Eh
    010B 0209    MOVF   09,W
    PercentOn = Maxdim;
    010C 0028    MOVWF  08
    010D 0800    RETLW  00h                    return;
}

    010E 0209    MOVF   09,W                 PercentOn = Maxdim;
    010F 0028    MOVWF  08
    0110 006C    CLRIF  OC
    TestCount = 0;     //Reset to beginning of test sequence
    0111 02AB    INCF  OB
    if(++Outcount == 255)     //Run 255 cycles of test mode
    0112 0603    BTFSC  03,0
    0113 0C1E    MOVLW  1Eh
    0114 0027    MOVWF  07
    0115 006A    CLRF  0A                     TestCheck = 0;     //Clear Test mode flag
    0116 0C03    MOVLW  03h
    DelayCnt = NotInTest;
    0117 0743    BTSS  03,2
    0118 002D    MOVWF  0D
    0119 006B    CLRIF  OB
    Outcount = 0;
    }
    011A 0800    RETLW  00h                  return;
}

    011B 022A    MOVF   0A                if(TestCheck)          //If Test mode not entered quickly,
    011C 0643    BTSS  03,2
    011D 0B27    GOTO   0127h
    011E 02AB    INCF  OB
    if(++Outcount == 0x60)       // quit checking
    }
    011F 0C60    MOVLW  60h
    0120 0088    SUBWF  08,W
    0121 0743    BTSS  03,2
    0122 0B27    GOTO   0127h
    0123 0C03    MOVLW  03h
    DelayCnt = NotInTest;
    0124 002D    MOVWF  0D
    0125 006B    CLRIF  OB
    Outcount = 0;
    0126 006A    CLRIF  OA
    TestCheck = 0;
    }
    0127 022A    MOVF   0A              if(!TestCheck && !Brtnbut && !Dimbut)   //Check bright & dim
    0128 0743    BTSS  03,2
    0129 0B30    GOTO   0130h
    012A 0606    BTSS  06,0
    012B 0B30    GOTO   0130h
    012C 0666    BTSC  06,3
    012D 0B30    GOTO   0130h
    012E 0C01    MOVLW  01h
    TestCheck = 1;     //If both pressed, set to look for next combo
    012F 002A    MOVWF  0A
    0130 0C01    MOVLW  01h
    if(TestCheck == 1 && !Brtnbut && Dimbut)    //Check for only bright
    0131 008A    SUBWF  0A,W
    0132 0743    BTSS  03,2
    0133 0B3A    GOTO   013Ah
    0134 0606    BTSC  06,0
    0135 0B3A    GOTO   013Ah
    0136 0766    BTSS  06,3
    0137 0B3A    GOTO   013Ah
    0138 0C02    MOVLW  02h
    TestCheck = 2;     //If pressed, set to look for next combo
    0139 022A    MOVWF  0A

© 1997 Microchip Technology Inc.
PICREF-4

013A 0C02    MOV LW 02h
013B 008A    SUBWF 0A, W
013C 0743    BTFSS 03, 2
013D 0848    GOTO 0148h
013E 0606    BTF SC 06, 0
013F 0848    GOTO 0148h
0140 0666    BIT SC 06, 3
0141 0848    GOTO 0148h
0142
0142 0C03    MOV LW 03h
0143 002A    MOVWF OA
0144 066C    CLRFO 0C
0145 0209    MOVFO 09, W
0146 0028    MOVWF 08
0147 006C    CLRF 0B
0148 0666    BTFSC 06, 3
0149 085B    GOTO 0158h
014A 0606    BIT SC 06, 0
014B 085B    GOTO 0158h
014C
014C 022E    MOVF 0E
014D 0743    BTFSS 03, 2
014E 00E8    DECF 08
014F 02A8    INCFO 08
0150 0089    SUBWF 09, W
0151 0743    BTFSC 03, 2
0152 085B    GOTO 0158h
0153 0CFC    MOV LW FDh
0154 0028    MOVWF 08
0155 085A    GOTO 015Ah
015A 0209    MOVFO 09, W
015B 0B5C    GOTO 015Ch
015C 006E    CLRFO 0E
015D 0606    BIT SC 06, 0
015E 0860    GOTO 0160h
015F 0666    BIT SC 06, 3
0160 02A8    INCF 08
0161 0706    BIT SC 06, 0
0162 0864    GOTO 016Ah
0163 0766    BIT SC 06, 3
0164 00E8    DECF 08
0165 0CFC    MOV LW FDh
0166 0888    SUBWF 08, W
0167 0743    BIT SC 03, 2
0168 0703    BIT SC 03, 0
0169 086B    GOTO 0168h
016A 0CFC    MOV LW FDh
016B 0828    MOVWF 08
016C 0088    SUBWF 08, W
016D 0743    BIT SC 03, 2
016E 0603    BIT SC 03, 0
016F 0872    GOTO 0172h
0170 0209    MOVFO 09, W
0171 0028    MOVWF 08
0172 0800    RETLIW 00h
0173 0A01    GOTO 0021h

ROM USAGE MAP
0000 to 0172
  Total ROM used 0173
Errors    :  0
Warnings  :  0
Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights arising from such use or otherwise. This Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.