PICREF-4

PICDIM Lamp Dimmer for the PIC12C508

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.

TABLE OF CONTENTS

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 V_{DD} is forward biased, and the input buffer will see approximately V_{DD}+0.7 volts and the software will read the pin as high. When the voltage on the line is negative, the protection diode from V_{SS} to the input pin is forward biased, and the input buffer sees approximately V_{SS}-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

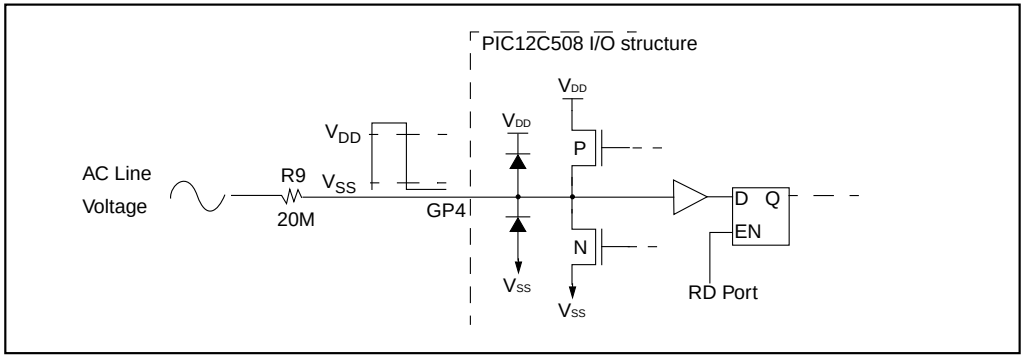


FIGURE 2: WAVEFORMS

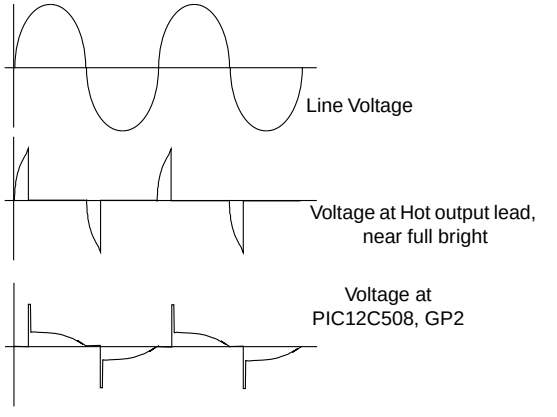
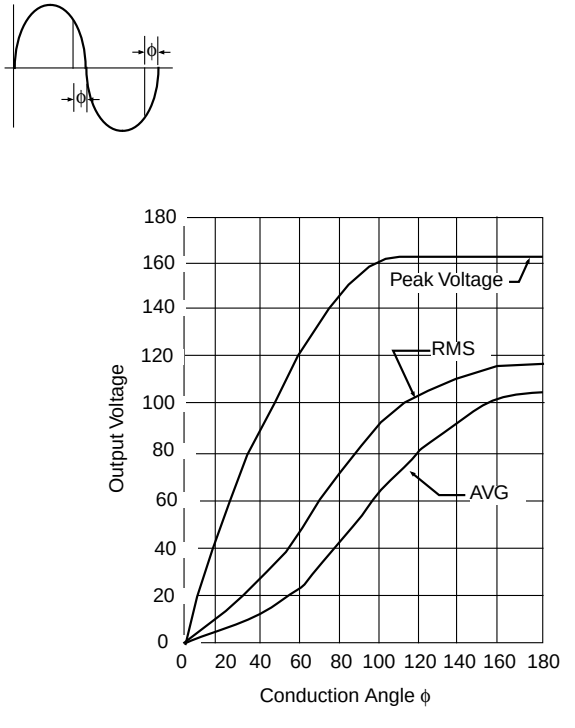


FIGURE 3: OUTPUT VOLTAGE OF FULL-WAVE PHASE CONTROL



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

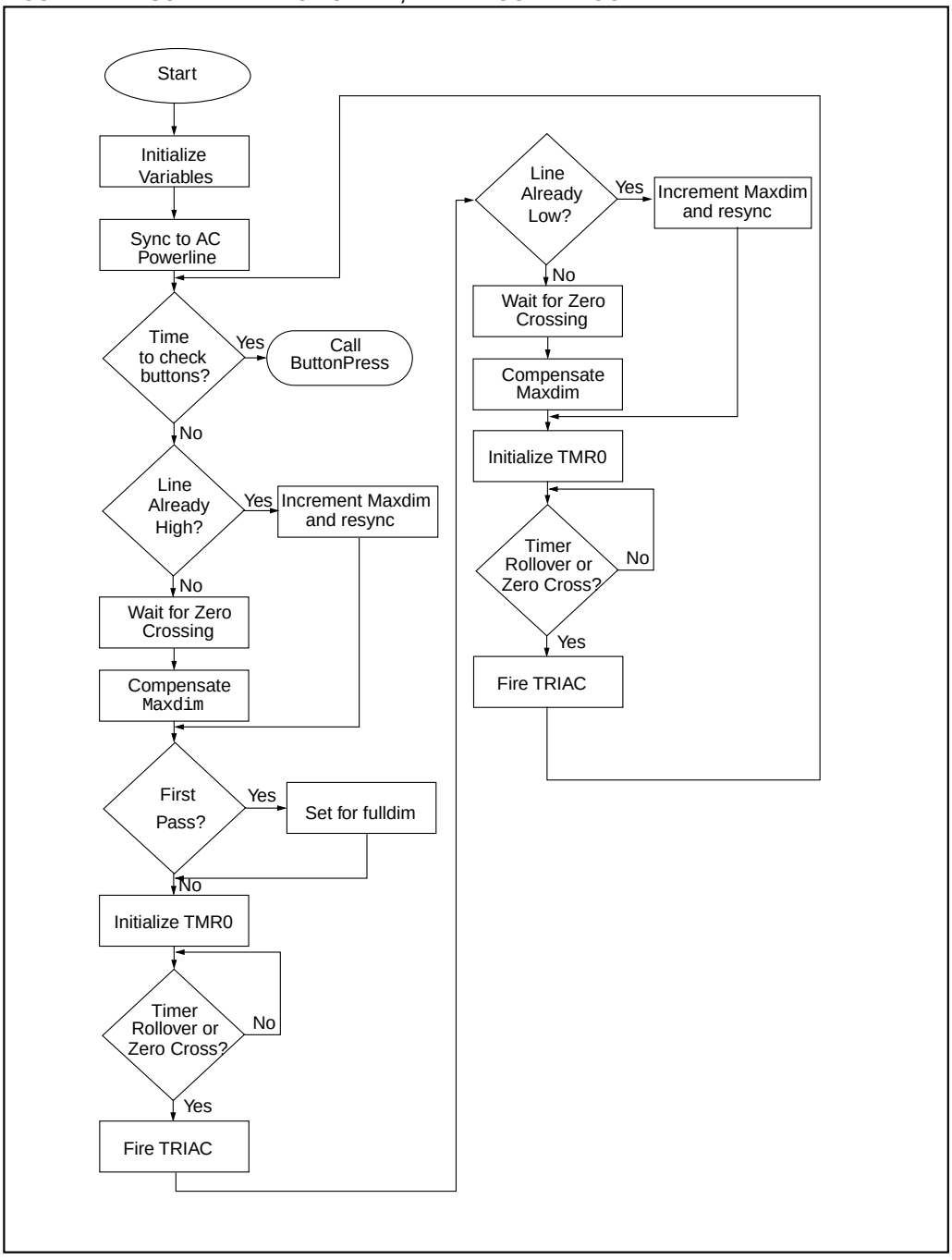
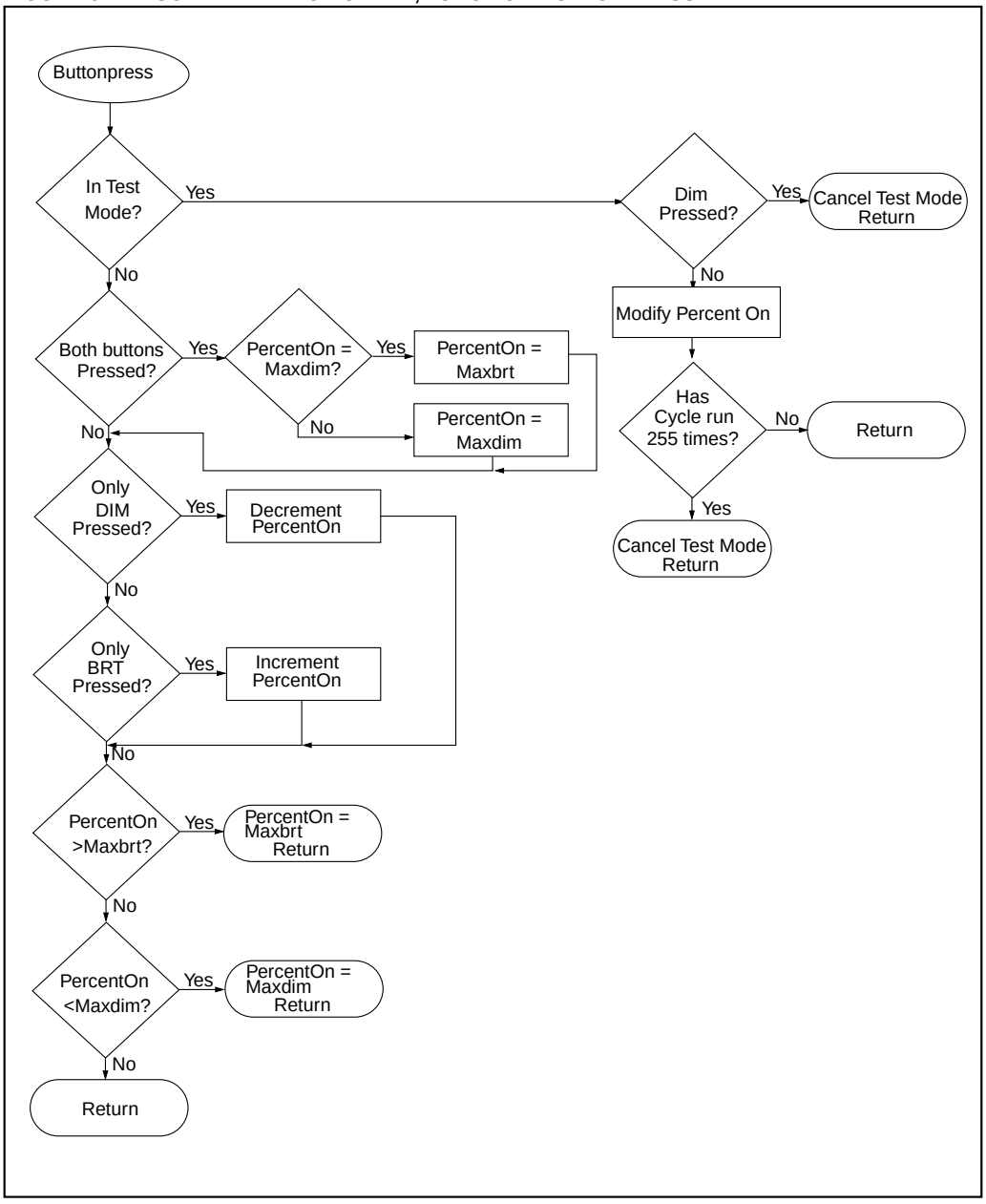


FIGURE 5: SOFTWARE FLOWCHART, FUNCTION BUTTONPRESS



NOTES:

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!

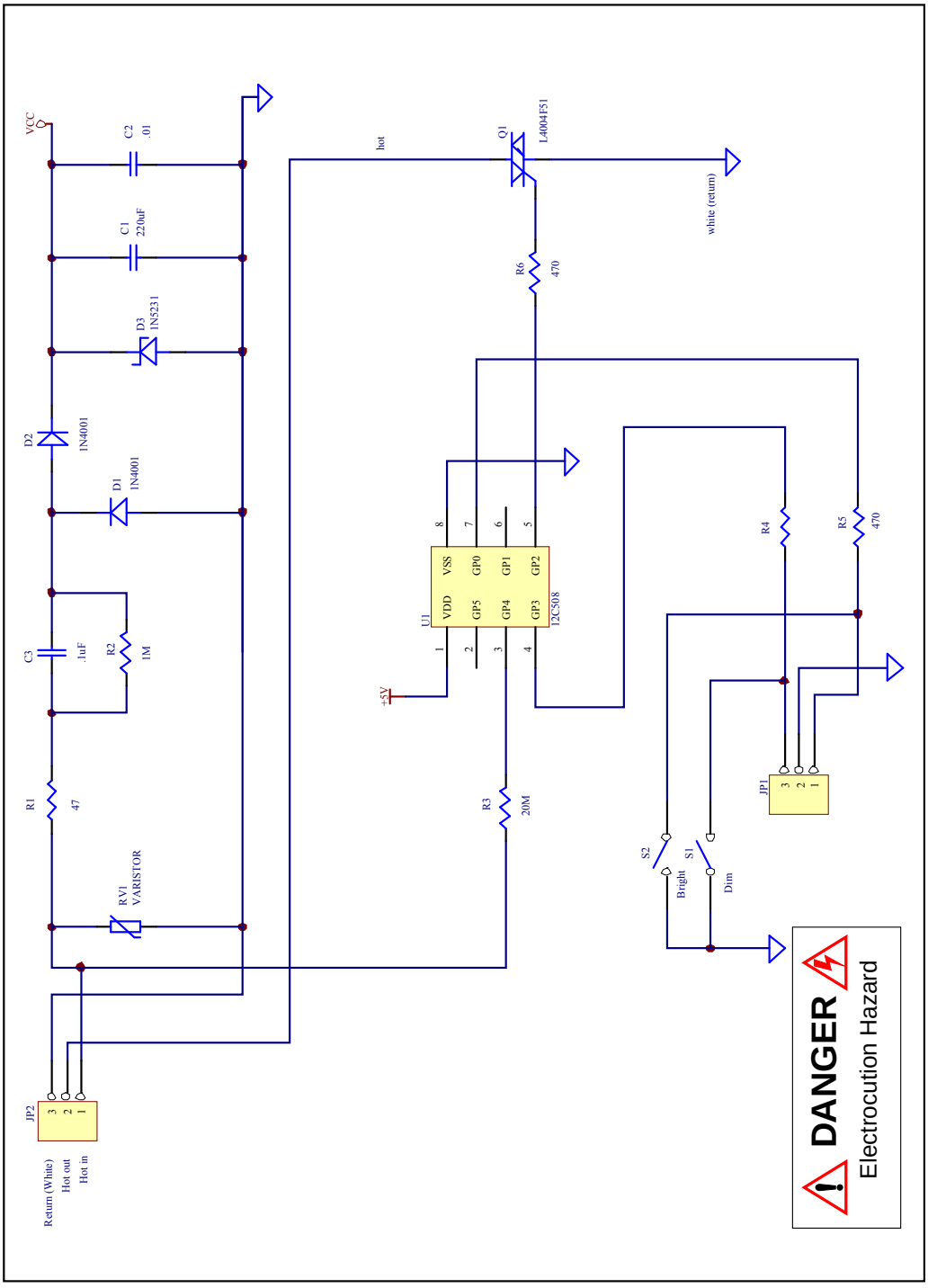
APPENDIX B: BILL OF MATERIALS

Description	Qty	Designators	Part #, Manufacturer, Contact #
Resistor, 1/4 Watt, 47ohm, Axial Lead	1	R1	Generic
Resistor, 1/4 Watt, 475ohm, Axial Lead	3	R4, R5, R6	Generic
Resistor, 1/4 Watt, 1Mohm, Axial Lead	1	R2	Generic
Resistor, 1/4 Watt, 20Mohm, Axial Lead	1	R3	Generic
8 Pin, 8-Bit, CMOS, Microcontroller	1	U1	12C508, Microchip Technology, Inc. (602) 786-7200
Logic Triac, TO-202AB, 400V	1	Q1	L4004F51, Teccor Electronics Inc. (214) 580-1515
Zener Diode, 5.1V, DO-35	1	D3	1N5231BCT, Diodes Incorporated/Digi-Key (800) 344-4539
Diode	2	D1, D2	1N4001, Generic
Keyswitch, Momentary PCB Mount	2	S1, S2	BF3-1000, Omron (847) 843-7900
ZNR Transient/Surge Absorbers, 1250A Surge, 300VDC, 230VAC	1	RV1	ERZ-V07D361, Panasonic (206) 395-7343
Aluminum Electrolytic Capacitor, 220uF, 35V	1	C1	ECE-A1VU221, Panasonic (206) 395-7343
Axial Ceramic Capacitor, 0.01uF, 50V	1	C2	A103Z15Z5UFVVWA, Philips (602) 820-2225
Polyester & Foil Capacitor, 0.1uF, 200V	1	C3	ECQ-M2104KZ, Panasonic (206) 395-7343

TABLE 1: BUTTON FUNCTIONS

Button	Function
BRT	Brighten
DIM	Dim
Hold DIM, Press BRT	If off: turn full on, if on: turn off
Hold BRT, Press, release, and press DIM again. To exit test mode, press DIM.	Enter test/demo mode

FIGURE 6: CIRCUIT DIAGRAM



APPENDIX C: SOFTWARE PROGRAM

```
#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-impeadance(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
/*****************************************************/
#defineBrtbut 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()
{
    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);
        CLRWDT;
    }
    WREG = 0x85;
    #asm ( OPTION);                                 //1:64 tmr0 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
        CLRWDT;
    TMR0 = PercentOn;                               //Get Delay time
    while(TMR0 >= 3 && LineInput == 0)              //Delay to enter main at proper point
        CLRWDT;
    while(1)                                        //Stay in this loop
```



```
{
    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;                           //Get delay time
        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;                       //Get Delay time
        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;
}
```


NOTES:

APPENDIX D: DIM508.LST FILE

```
MPLAB-C "C" COMPILER V1.21 Released                                    PAGE  1
                              #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: 0x07
                              */
                              #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-impeadance(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                                */
                              /*****************************************************/
0007                          #define Brtbut GPIO.0            //Brighten button
0008                          #define Output GPIO.2            //Output to TRIAC
0009                          #define Dimbut GPIO.3            //Dim button
000A                            #define LineInput GPIO.4          //AC line zero crossing sense
                              void Buttoncheck(void);        //Button check routine
0008 0009                     unsigned int PercentOn, Maxdim  ;         //Global variables
000A 000B 000C                unsigned int TestCheck, Outcount, TestCount;
000D                          unsigned int DelayCnt;
000E 000F                     unsigned int LastBoth, FirstPass;
0010                          unsigned int Count;
007E 0001                     const Maxbrt = 0xFD, NotInTest = 3;
                              void main()
                              {
0001 0CD0    MOVLW  D0h          PercentOn = 0xD0;       //On Period
0002 0028    MOVWF  08
0003 0C70    MOVLW  70h          Maxdim = 0x70;         //Value of Maximum dimming
0004 0029    MOVWF  09
0005 006A    CLRF   0A           TestCheck = 0;         //Test mode check counter
0006 006B    CLRF   0B           Outcount = 0;          //Counter for test mode exit
0007 006C    CLRF   0C           TestCount = 0;         //Test mode counter
0008 0C03    MOVLW  03h          DelayCnt = NotInTest;          //Delay count
0009 002D    MOVWF  0D
000A 006E     CLRF   0E            LastBoth = 0;           //Both buttons pressed last time flag
000B 0C01    MOVLW  01h          FirstPass = 1;         //Indicate power-up
000C 002F    MOVWF  0F
000D 0070    CLRF   10           Count = 0;              //General counter
000E 0070     CLRF   10           for(Count = 0; Count < 60; Count++)       //Allow power supply
                                                                            to stabilize
000F 0C3C    MOVLW  3Ch
0010 0090    SUBWF  10,W
0011 0603    BTFSC  03,0
0012 0A1A    GOTO   001Ah
                                  {
```

```
0013 0686    BTFSC  06,4                while(LineInput == 1);
0014 0A13    GOTO   0013h
0015 0786    BTFSS  06,4                while(LineInput == 0);
0016 0A15    GOTO   0015h
0017 0004    CLRWDT                     CLRWDT;
                                       }
0018 02B0    INCF   10
0019 0A0F    GOTO   000Fh
                                       WREG = 0x85;
001A 0C85    MOVLW  85h                 #asm ( OPTION);
001B 0002    OPTION
                                       WREG = 0x1D;
001C 0C1D    MOVLW  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    MOVLW  04h               GPIO = 0x04;         //Set TRIAC output latch high
001F 0026    MOVWF  06
                                       while(LineInput == 1)          //Synch to line phase
0020 0786    BTFSS  06,4
0021 0A24    GOTO   0024h
0022 0004    CLRWDT                     CLRWDT;
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
                                                                              at proper point
0027 0081    SUBWF  01,W
0028 0703    BTFSS  03,0
0029 0A2E    GOTO   002Eh
002A 0686    BTFSC  06,4
002B 0A2E    GOTO   002Eh
002C                                   CLRWDT;
002C 0004    CLRWDT
002D 0A26    GOTO   0026h
                                       while(1)             //Stay in this loop
                                       {
002E 0070    CLRF   10
002F                                   Count = 0;
                                      while (Count++ < DelayCnt)    //Check for button press every
                                                                      DelayCnt zero crossings
002F 0210    MOVF   10,W                {
0030 02B0    INCF   10
0031 008D    SUBWF  0D,W
0032 0743    BTFSS  03,2
0033 0703    BTFSS  03,0
0034 0AA5    GOTO   00A5h
0035                                     if(LineInput == 1)     //Check for AC line already high
0035 0786    BTFSS  06,4                 {
0036 0A40    GOTO   0040h
0037 0C05    MOVLW  05h                   Maxdim += 5;       //If so, increment Maxdim
0038 01E9    ADDWF  09
0039 0686    BTFSC  06,4                  while(LineInput == 1);  // and re-sync with line
003A 0A39    GOTO   0039h
003B 0686    BTFSC  06,4                  while(LineInput == 0)
003C 0A3F    GOTO   003Fh
003D                                          CLRWDT;
003D 0004    CLRWDT
003E 0A3B    GOTO   003Bh
                                         }
003F 0A4A    GOTO   004Ah                 else
                                         {
0040 0686    BTFSC  06,4                  while(LineInput == 0)   //Wait for zero crossing
0041 0A44    GOTO   0044h
0042                                          CLRWDT;
0042 0004    CLRWDT
```

```
0043 0A40    GOTO   0040h
0044 0201    MOVF   01,W                    Maxdim = PercentOn - TMR0 + 2;  //Compensate full dim
                                                                              value for line
0045 0088    SUBWF  08,W
0046 0027    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    BTFSS  03,2
004D 0A51    GOTO   0051h
004E                                           {
004E 006F    CLRF   0F                          FirstPass = 0;
004F 0209    MOVF   09,W                        PercentOn = Maxdim;
0050 0028    MOVWF  08
                                               }
0051 0209    MOVF   09,W                     if(PercentOn < Maxdim)   //If maxdim moved, fix brightness
0052 0088    SUBWF  08,W
0053 0743    BTFSS  03,2
0054 0603    BTFSC  03,0
0055 0A58    GOTO   0058h
0056 0209    MOVF   09,W                        PercentOn = Maxdim;
0057 0028    MOVWF  08
0058 0208    MOVF   08,W                       TMR0 = PercentOn;       //Get delay time
0059 0021    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    BTFSS  03,0
005D 0A62    GOTO   0062h
005E 0786    BTFSS  06,4
005F 0A62    GOTO   0062h
0060                                           CLRWDT;
0060 0004    CLRWDT
0061 0A5A    GOTO   005Ah

0062 0C04    MOVLW  04h                        GPIO = 0x04;        //Set TRIAC output latch high
0063 0026    MOVWF  06
                                           WREG = 0x19;
0064 0C19    MOVLW  19h                    #asm ( TRIS GPIO);
0065 0006    TRIS   PORTB
                                     //       __TRIS(0x19,GPIO);     //Fire Triac
0066 0000    NOP                               NOP;          //Delay for TRIAC fire pulse
0067 0000    NOP                               NOP;
0068 0000    NOP                               NOP;
0069 0000    NOP                               NOP;
006A 0000    NOP                               NOP;
006B 0000    NOP                               NOP;
006C 0000    NOP                               NOP;
                                           WREG = 0x1D;
006D 0C1D    MOVLW  1Dh                    #asm ( TRIS GPIO);
006E 0006    TRIS   PORTB
                                     //       __TRIS(0x1D,GPIO);     //Release TRIAC fire signal
006F 0004    CLRWDT                            CLRWDT;

0070 0686    BTFSC  06,4                       if(LineInput == 0)     //Check for AC line already low
0071 0A7B    GOTO   007Bh
0072                                           {
0072 0C05    MOVLW  05h                         Maxdim += 5;       //If so, increment Maxdim
0073 01E9    ADDWF  09
0074 0786    BTFSS  06,4                        while(LineInput == 0);  // and re-sync with line
0075 0A74    GOTO   0074h
                                                while(LineInput == 1)
```

```
0076 0786    BTFSS  06,4
0077 0A7A    GOTO   007Ah
0078 0004    CLRWDT                              CLRWDT;
0079 0A76    GOTO   0076h
                                               }
007A 0A85    GOTO   0085h                      else
                                               {
                                                while(LineInput==1) //Wait for zero crossing
007B 0786    BTFSS  06,4
007C 0A7F    GOTO   007Fh
007D 0004    CLRWDT                              CLRWDT;
007E 0A7B    GOTO   007Bh
007F 0201    MOVF   01,W                 Maxdim = PercentOn - TMR0 + 2;  //Compensate full dim value for
                                                                             line
0080 0088    SUBWF  08,W
0081 0027    MOVWF  07
0082 0C02    MOVLW  02h
0083 01C7    ADDWF  07,W
0084 0029    MOVWF  09
                                                        // frequency vs osc. speed
                                               }
0085 0209    MOVF   09,W                 if(PercentOn < Maxdim)     //If maxdim moved, fix brightness
0086 0088    SUBWF  08,W
0087 0743    BTFSS  03,2
0088 0603    BTFSC  03,0
0089 0A8C    GOTO   008Ch
008A 0209    MOVF   09,W                        PercentOn = Maxdim;
008B 0028    MOVWF  08
008C 0208    MOVF   08,W                       TMR0 = PercentOn;       //Get Delay time
008D 0021    MOVWF  01
008E 0C03    MOVLW  03h                       while(TMR0 >= 3 && LineInput == 0) //Delay TRIAC turn on
008F 0081    SUBWF  01,W
0090 0703    BTFSS  03,0
0091 0A96    GOTO   0096h
0092 0686    BTFSC  06,4
0093 0A96    GOTO   0096h
0094                                           CLRWDT;
0094 0004    CLRWDT
0095 0A8E    GOTO   008Eh
0096 0C04    MOVLW  04h                        GPIO = 0x04;        //Set TRIAC output latch high
0097 0026    MOVWF  06
                                           WREG = 0x19;
0098 0C19    MOVLW  19h                     #asm ( TRIS GPIO);
0099 0006    TRIS   PORTB
                                     //       __TRIS(0x19,GPIO);     //Fire TRIAC
009A 0000    NOP                               NOP;          //Delay for TRIAC fire pulse
009B 0000    NOP                               NOP;
009C 0000    NOP                               NOP;
009D 0000    NOP                               NOP;
009E 0000    NOP                               NOP;
009F 0000    NOP                               NOP;
00A0 0000    NOP                               NOP;
                                           WREG = 0x1D;
00A1 0C1D    MOVLW  1Dh                     #asm ( TRIS GPIO);
00A2 0006    TRIS   PORTB
                                     //       __TRIS(0x1D,GPIO);     //Release TRIAC fire signal
00A3 0004    CLRWDT                            CLRWDT;
00A4 0A2F    GOTO   002Fh                    }
00A5 09A8    CALL   00A8h                    Buttoncheck();          //Check for button press
00A6 0A2E    GOTO   002Eh                  }
00A7 0800    RETLW  00h                  }
                                         /*****************************************************/
                                         /* 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 MPLABC 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    BTFSS  06,0          if(Brtbut && !Dimbut)     //If Dimbutton pressed, exit test mode
00B0 0AB7    GOTO   00B7h
00B1 0666    BTFSC  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    BTFSS  03,2
00B9 0AC5    GOTO   00C5h
00BA                                   {
00BA 02A8    INCF   08                   if(++PercentOn > Maxbrt)    //Check for full bright
00BB 0CFD    MOVLW  FDh
00BC 0088    SUBWF  08,W
00BD 0743    BTFSS  03,2
00BE 0703    BTFSS  03,0
00BF 0AC4    GOTO   00C4h
00C0                                     {
00C0 0CFD    MOVLW  FDh                   PercentOn = Maxbrt;
00C1 0028    MOVWF  08
00C2 02AC    INCF   0C                    ++TestCount;
00C3 0800    RETLW  00h                   return;
                                         }
                                         else
00C4 0800    RETLW  00h                   return;
                                       }
00C5 0C01    MOVLW  01h                 if(TestCount == 1)       //Ramp down to full dim
00C6 008C    SUBWF  0C,W
00C7 0743    BTFSS  03,2
00C8 0AD5    GOTO   00D5h
00C9                                   {
00C9 00E8    DECF   08                   if(--PercentOn <= Maxdim) //Check for full dim
00CA 0208    MOVF   08,W
00CB 0089    SUBWF  09,W
00CC 0643    BTFSC  03,2
00CD 0AD0    GOTO   00D0h
00CE 0703    BTFSS  03,0
00CF 0AD4    GOTO   00D4h
00D0                                     {
00D0 0CFD    MOVLW  FDh                   PercentOn = Maxbrt;
00D1 0028    MOVWF  08
```

```
00D2 02AC    INCF   0C                    ++TestCount;
00D3 0800    RETLW  00h                   return;
                                         }
                                         else
00D4 0800    RETLW  00h                   return;
                                        }
00D5 020C    MOVF   0C,W                 while(TestCount++ < 5)       //Delay
00D6 02AC    INCF   0C
00D7 0027    MOVWF  07
00D8 0C05    MOVLW  05h
00D9 0087    SUBWF  07,W
00DA 0703    BTFSS  03,0
00DB                                       return;
00DB 0800    RETLW  00h
                                        while(TestCount++ < 10)     //Turn off for a short period
00DC 020C    MOVF   0C,W                {
00DD 02AC    INCF   0C
00DE 0027    MOVWF  07
00DF 0C0A    MOVLW  0Ah
00E0 0087    SUBWF  07,W
00E1 0603    BTFSC  03,0
00E2 0AE6    GOTO   00E6h
00E3                                       PercentOn = Maxdim;
00E3 0209    MOVF   09,W
00E4 0028    MOVWF  08
00E5 0800    RETLW  00h                    return;
                                        }
                                        while(TestCount++ < 15)     //Turn On for a short period
00E6 020C    MOVF   0C,W                {
00E7 02AC    INCF   0C
00E8 0027    MOVWF  07
00E9 0C0F    MOVLW  0Fh
00EA 0087    SUBWF  07,W
00EB 0603    BTFSC  03,0
00EC 0AF0    GOTO   00F0h
00ED                                       PercentOn = Maxbrt;
00ED 0CFD    MOVLW  FDh
00EE 0028    MOVWF  08
00EF 0800    RETLW  00h                    return;
                                        }
                                        while(TestCount++ < 20)     //Turn off for a short period
00F0 020C    MOVF   0C,W                {
00F1 02AC    INCF   0C
00F2 0027    MOVWF  07
00F3 0C14    MOVLW  14h
00F4 0087    SUBWF  07,W
00F5 0603    BTFSC  03,0
00F6 0AFA    GOTO   00FAh
00F7                                       PercentOn = Maxdim;
00F7 0209    MOVF   09,W
00F8 0028    MOVWF  08
00F9 0800    RETLW  00h                    return;
                                        }
                                        while(TestCount++ < 25)     //Turn on for a short period
00FA 020C    MOVF   0C,W                {
00FB 02AC    INCF   0C
00FC 0027    MOVWF  07
00FD 0C19    MOVLW  19h
00FE 0087    SUBWF  07,W
00FF 0603    BTFSC  03,0
0100 0B04    GOTO   0104h
0101                                       PercentOn = Maxbrt;
0101 0CFD    MOVLW  FDh
0102 0028    MOVWF  08
0103 0800    RETLW  00h                    return;
                                        }
```

```
                                          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 0603    BTFSC  03,0
010A 0B0E    GOTO   010Eh
010B                                         PercentOn = Maxdim;
010B 0209    MOVF   09,W
010C 0028    MOVWF  08
010D 0800    RETLW  00h                      return;
                                          }
010E 0209    MOVF   09,W                   PercentOn = Maxdim;
010F 0028    MOVWF  08
0110 006C    CLRF   0C                    TestCount = 0;         //Reset to beggining of test sequence
0111 02AB    INCF   0B                     if(++Outcount == 255)     //Run 255 cycles of test mode
0112 0CFF    MOVLW  FFh
0113 008B    SUBWF  0B,W
0114 0743    BTFSS  03,2
0115 0B1A    GOTO   011Ah
0116                                      {
0116 006A    CLRF   0A                       TestCheck = 0;     //Clear Test mode flag
0117 0C03    MOVLW  03h                      DelayCnt = NotInTest;
0118 002D    MOVWF  0D
0119 006B    CLRF   0B                       Outcount = 0;
                                          }
011A 0800    RETLW  00h                    return;
                                        }

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

```
013A 0C02    MOVLW  02h         if(TestCheck == 2 && !Brtbut && !Dimbut) //Check for both
                                                                          pressed again
013B 008A    SUBWF  0A,W
013C 0743    BTFSS  03,2
013D 0B48    GOTO   0148h
013E 0606    BTFSC  06,0
013F 0B48    GOTO   0148h
0140 0666    BTFSC  06,3
0141 0B48    GOTO   0148h
0142                                {
0142 0C03    MOVLW  03h                TestCheck = 3;         //Enable test mode
0143 002A    MOVWF  0A
0144 006C    CLRF   0C                 TestCount = 0;
0145 0209    MOVF   09,W               PercentOn = Maxdim;
0146 0028    MOVWF  08
0147 006B    CLRF   0B                 Outcount = 0;
                                    }
0148 0666    BTFSC  06,3             if(!Dimbut && !Brtbut)          //If both pressed
0149 0B5B    GOTO   015Bh
014A 0606    BTFSC  06,0
014B 0B5B    GOTO   015Bh
014C                                {
014C 022E    MOVF   0E                 if(LastBoth == 0)     //Don't flash if held
014D 0743    BTFSS  03,2
014E 0B5A    GOTO   015Ah
014F                                   {
014F 0C01    MOVLW  01h                  LastBoth = 1;
0150 002E    MOVWF  0E
0151 0208    MOVF   08,W                 if(PercentOn == Maxdim)   //If full off...
0152 0089    SUBWF  09,W
0153 0743    BTFSS  03,2
0154 0B58    GOTO   0158h
0155 0CFD    MOVLW  FDh                   PercentOn = Maxbrt; // turn full on...
0156 0028    MOVWF  08
0157 0B5A    GOTO   015Ah                ELSE
0158 0209    MOVF   09,W                  PercentOn = Maxdim; // otherwise turn off
0159 0028    MOVWF  08
                                       }
                                    }
015A 0B5C    GOTO   015Ch           else
015B 006E    CLRF   0E                 LastBoth = 0;
015C 0606    BTFSC  06,0             if(!Brtbut && Dimbut)       //Check for brighten cmd
015D 0B60    GOTO   0160h
015E 0666    BTFSC  06,3
015F 02A8    INCF   08                 PercentOn ++;
0160 0706    BTFSS  06,0             if(Brtbut && !Dimbut)       //Check for dim cmd
0161 0B64    GOTO   0164h
0162 0766    BTFSS  06,3
0163 00E8    DECF   08                 PercentOn --;
0164 0CFD    MOVLW  FDh              if(PercentOn > Maxbrt)        //If greater than full bright
0165 0088    SUBWF  08,W
0166 0743    BTFSS  03,2
0167 0703    BTFSS  03,0
0168 0B6B    GOTO   016Bh
0169 0CFD    MOVLW  FDh                PercentOn = Maxbrt;
016A 0028    MOVWF  08
016B 0209    MOVF   09,W             if(PercentOn < Maxdim)        //If less than full dim
016C 0088    SUBWF  08,W
016D 0743    BTFSS  03,2
016E 0603    BTFSC  03,0
016F 0B72    GOTO   0172h
0170 0209    MOVF   09,W               PercentOn = Maxdim;
0171 0028    MOVWF  08
0172 0800    RETLW  00h           }
0000 0A01    GOTO   0001h
ROM USAGE MAP
```

```
    0000 to 0172
    Total ROM used 0173
Errors              :    0
Warnings            :    0
```

NOTES:

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-1189-21-5858 Fax: 44-1189-21-5835

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Müchen, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

10/31/97

DNV Certification, Inc. USA
ACCREDITED REGISTRAR
ANSI-RAB
DNV
DNV MSC The Netherlands Accredited by the RvA
ISO 9001
REGISTERED FIRM