TEXAS INSTRUMENTS

Low Cost TRIAC Control With MSP430 16-Bit Microcontroller

Application Report

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Contents

List of Figures

List of Tables

Low Cost Triac Control With MSP430 16-Bit Microcontroller

By Anton Muehlhofer,
Texas Instruments Incorporated

ABSTRACT

This application report describes a complete triac control system using the MSP430. An example of a triac control system is presented and analyzed. Methods for interference reduction, modification for high speed, and enhance security are explained. Alternate power supply circuits are included. A listing of triac control software available is presented in Appendixes A1 through A3.

1 Introduction

Triac controlled universal motors can be found in many consumer applications such as refrigerators, washing machines, vacuum cleaners, power drills, and many more. In addition to performing a phase-shift control algorithm, most applications require that the control system perform dedicated additional functions like speed/torque regulation, motor protection, ramp-up control, display and sequence control, and so on. From the system cost standpoint, the 16-bit MSP430 can provide significantly higher performance than 4-bit and 8-bit controllers, and can be used to implement additional features or sophisticated algorithms for higher energy efficiency.

This document describes a complete triac control system using the MSP430, including ready to use schematics and assembly software.

2 MSP430C112 Optimized for System Cost Reduction

The MSP430C11x family incorporates some valuable features, which reduce system cost significantly. One of them is its ultra-low power consumption, requiring a very small power supply circuit. The other is a fully-functional internal oscillator which runs at software-adjustable frequencies up to 5 MHz without the need of external components. Both features are very useful in cost-sensitive triac control applications.

Usually, RC power supplies are used for triac-controlled motor applications like washing machines. The capacitor required needs to withstand about 400 V, and has to deliver the current used by the control circuitry. This determines the capacitance value. If the supply current for the control circuitry can be reduced, a smaller capacitor can be used, which in turn reduces cost. A resistor-divider power supply can be used to reduce power supply cost even further.

3 Timer Usage

The MSP430C112 timer module, called *Timer A,* has a very powerful and highly flexible structure. It consists of a 16-bit timer that can be fed by the same clock as the CPU (up to 5 MHz). Three software-configurable capture/compare blocks are connected to this 16-bit timer. These blocks have dedicated multiplexed I/O pins that can be configured as inputs for capture function, or outputs for compare-set, clear, or toggle functions.
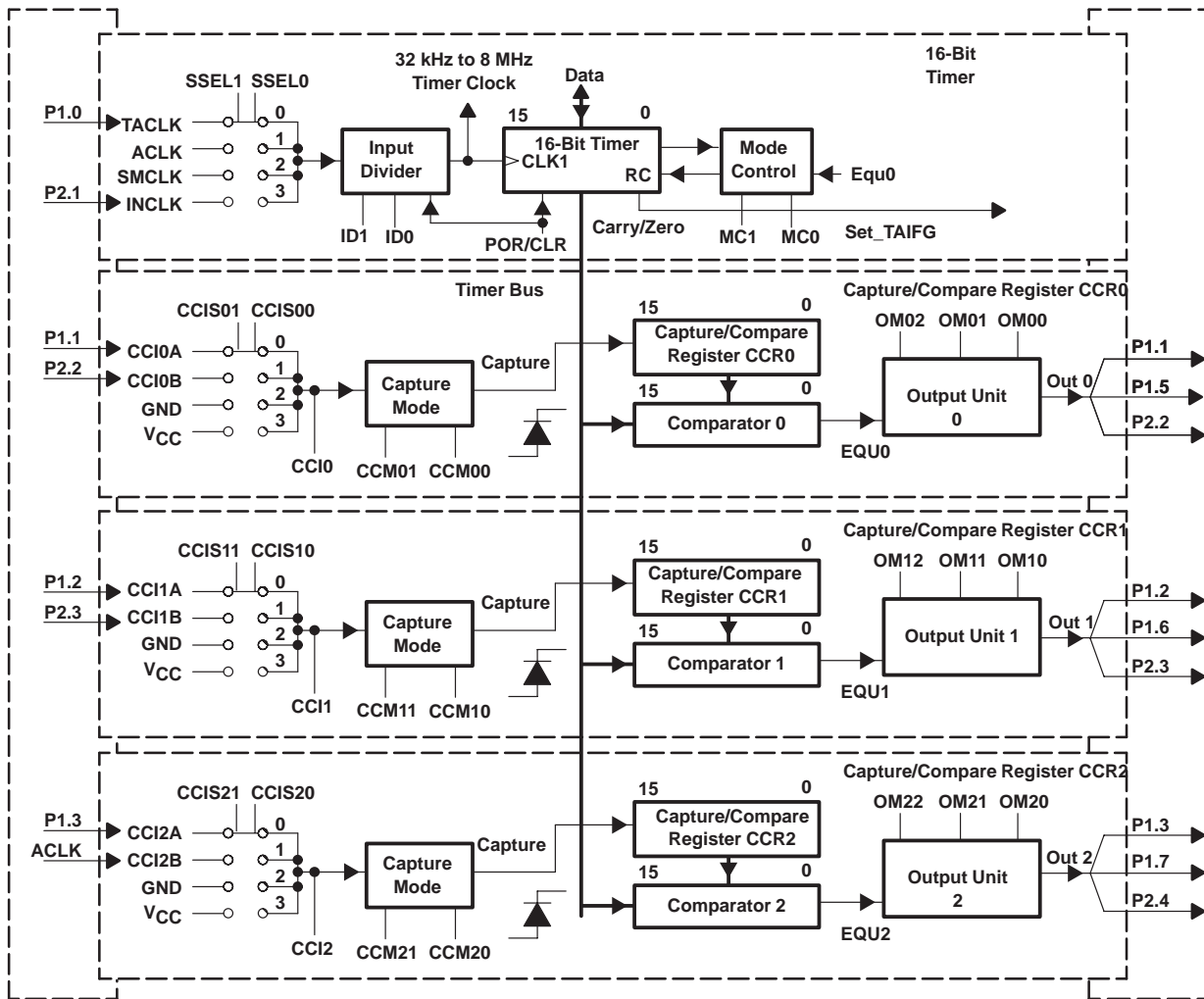
Figure 1. Timer A Block Diagram

In the following triac control application example, Timer A is used as shown in Table 1.

Table 1. Timer A Usage in Application Example

16-BIT TIMER	CONTINUOUS UP MODE
Capture/compare register CCR0	ADC measurement
Capture/compare register CCR1	Free for further improvements like SCI, UART
Capture/compare register CCR2	Zero-voltage detection and phase-shift control

4 Application Example

The application example described below has been built with a 12-V ac supply voltage to minimize hazards during development. It can be converted very easily to run on ac power by simply changing the power supply.
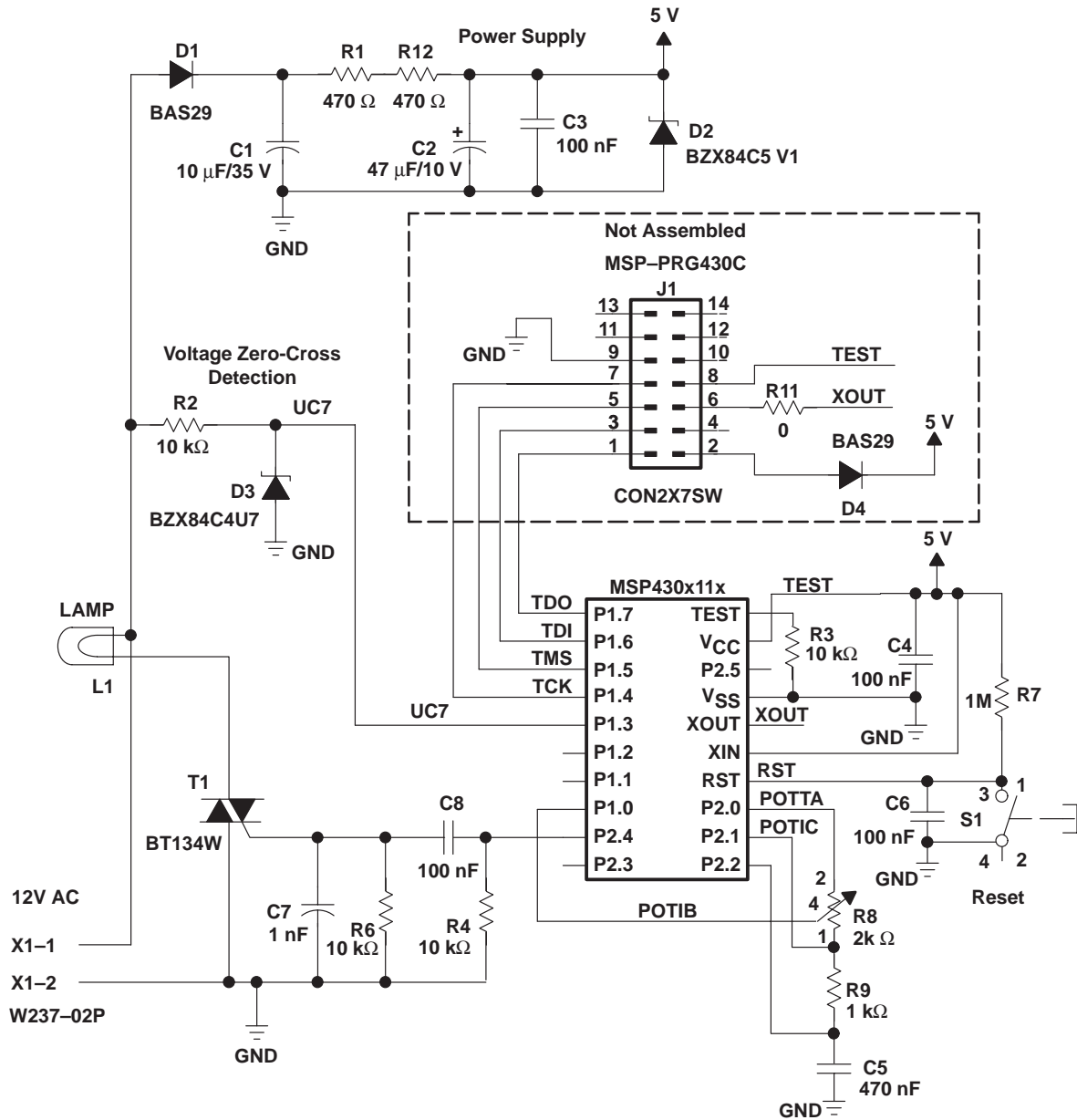
Figure 2. Triac Control Circuit Example

4.1 Power Supply

The power-supply circuitry has been built up with a simple one-way rectifier. The power dissipation of the resistor has been derived from the formula:

$$P_R = \frac{1}{R}\left(\frac{2}{\sqrt{3}} \times Ueff - Vz\right)^2 = \frac{1}{940\ \Omega} \times \left(\frac{2}{\sqrt{3}} \times 12\ V - 5.1\ V\right)^2 = 82\ mW$$

When assuming a 3-mA minimum current through the Zener diode, the maximum continuous current this power supply can provide is:

$$iout = \frac{1}{R}\left(\frac{2}{\sqrt{3}} \times Ueff - Vz\right) - 3\ mA = 5\ mA$$

The MSP430C112 requires only 770 μA at 5 V, 1 MHz in fully-running mode. All of the remaining current can be used to drive the triac gate. Dynamic drive is used to reduce triac gate current, with three pulses per triac fire signal.

4.2 Power Line Supply

In applications where ac power is used as the source of energy, the circuitry shown in Figure 3 can be used.
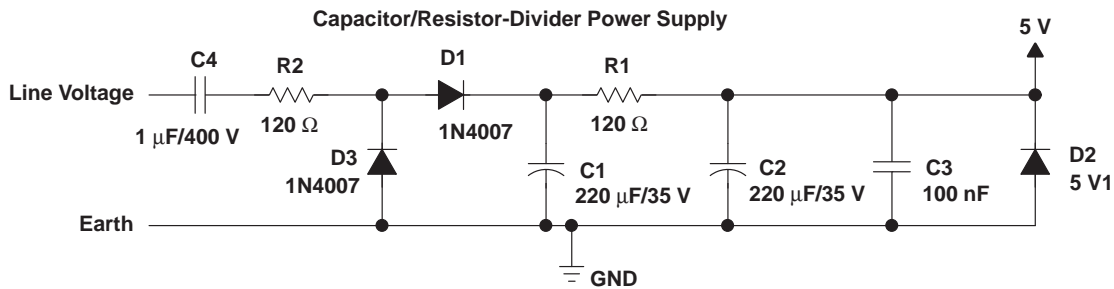
Figure 3. Power Line Supply

A capacitor-resistor divider power supply circuit like that shown in Figure 3 can be used to supply currents up to 10 mA with a capacitor C4 of 1 μF. The overall power supply current, the power-on rise-time, and maximum ripple voltage, have to be taken into account during parts selection. The component values in Figure 3 may be used to obtain 10 mA of current, with 30 mV maximum ripple voltage and 400 ms power-on rise-time.
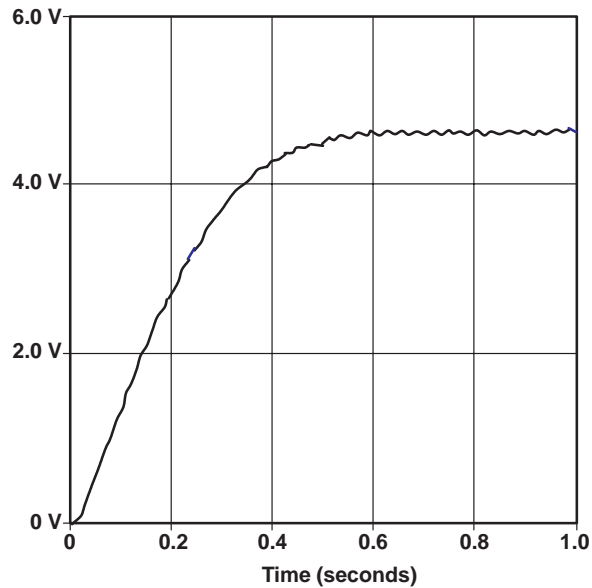
Figure 4. Settle Time With Power Line Supply

4.3   JTAG Connection

The JTAG pins (TDO, TDI, TMS, TCK) are multiplexed with digital I/O pins. In this application example they are used as JTAG functions only to allow in-circuit programming. This can be done without a power supply connection. The programming adapter supplies the MSP430 through a diode for reverse-current protection. The programming adapter must be revision C or higher.

4.4   Analog-to-Digital Conversion

A potentiometer is used for phase-angle-control set-point adjustment. Since the MSP430C112 has no A/D converter onboard, the value of the potentiometer is determined by the single-slope conversion approach with the help of the internal 16-bit timer-capture registers.
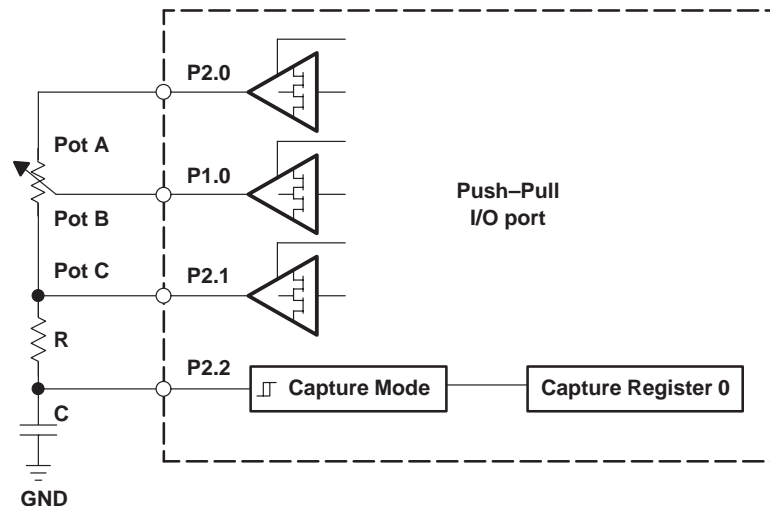
Figure 5. A/D Conversion Diagram
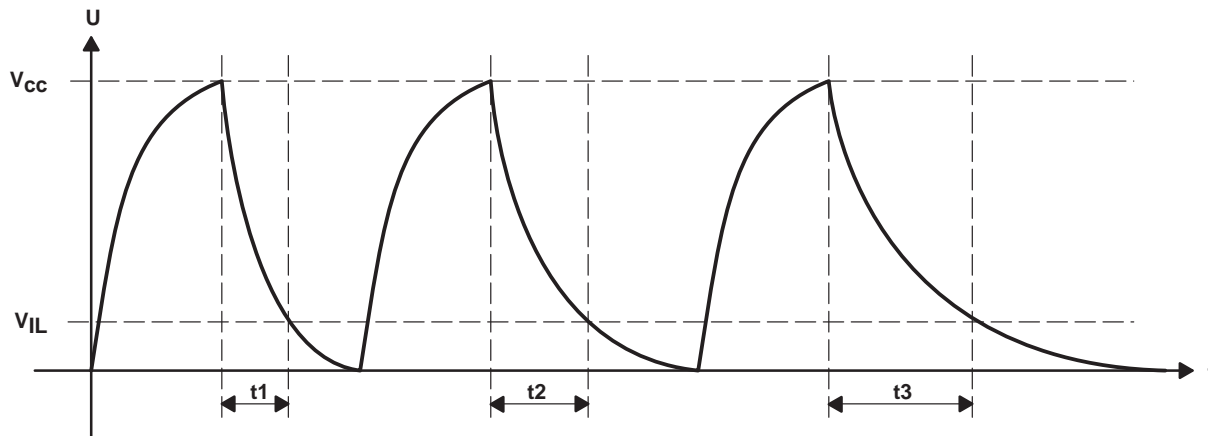
t1: **Time to discharge C through PotC (=R)**
t2: **Time to discharge C through PotB+R**
t3: **Time to discharge C through PotA+R**

Figure 6. A/D Conversion Principles

The timer is clocked directly with the free-running DCO clock frequency, which depends on temperature and supply voltage. To increase measurement accuracy, the time measurement of the capacitor discharge through the potentiometer is started when the line voltage crosses zero. This approach ensures that the supply voltage is the same for every A/D conversion, and eliminates the influence of voltage on the timer clock caused by supply-voltage ripple.

The pot terminals are connected to I/O pins of the MSP430, which have the capability to charge the capacitor (set pin output high), discharge the capacitor (set pin output low), or do nothing (set pin to input). All potentiometer terminals are measured to determine the position of the potentiometer wiper. The whole range of possible positions is divided into 256 discrete values using the following formula:

$$Pot\_Value[0.255] = \frac{(PotB - PotC) \times 256}{PotA - PotC}$$

The values PotA, PotB, and PotC are the number of cycles needed to discharge the capacitor over each pot terminal. The number of cycles is measured from the start of discharge, until the negative-going threshold voltage of the capture input pin (which has a Schmitt-trigger characteristic) is reached. With this approach, the absolute frequency of the timer does not influence the result, provided it is sufficiently high to guarantee adequate resolution, and remains stable while the three conversions are being performed.

In this application example, the capture/compare register CCR0 is used in capture mode and is connected to the discharge capacitor via pin P2.2.

4.5   TRIAC Gate Driver

Triac circuit development can be simplified by using triacs designed for low-gate current requirements which may be directly driven by microcontroller outputs. They are suitable for controlling load currents in the 0.5 A–1 A range. However, the triac gate-control circuitry used is designed to fire common triacs like the TIC206, as well as gate sensitive devices. For higher gate-drive triacs, several outputs of the MSP430 can be connected together to provide higher current peaks for charging/discharging the gate-driving capacitor.

Due to positive and negative pulses occurring on the gate, the triac gate control used has the advantage of being triggered in the quadrant where the lowest trigger current is required. This significantly reduces overall supply current.

It must be taken into account that when triacs are triggered at medium-phase angle and drive inductive loads, they generate heavy voltage peaks on the gate, which could damage the microcontroller. The following diagram shows two possible triac gate drivers and their gate voltages for systems fed from ac power.

Circuitry:
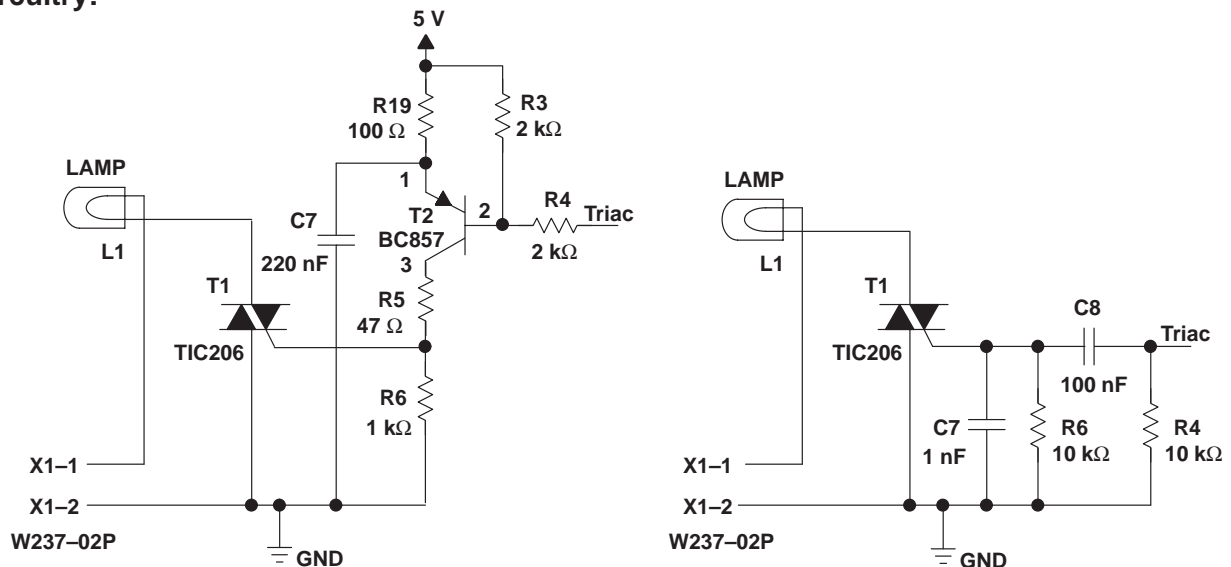
Figure 7. Triac Trigger Signals

Triggered triac gate with positive voltage on MT1:
  Channel 1: voltage on triac gate
  Channel 2: voltage on label triac
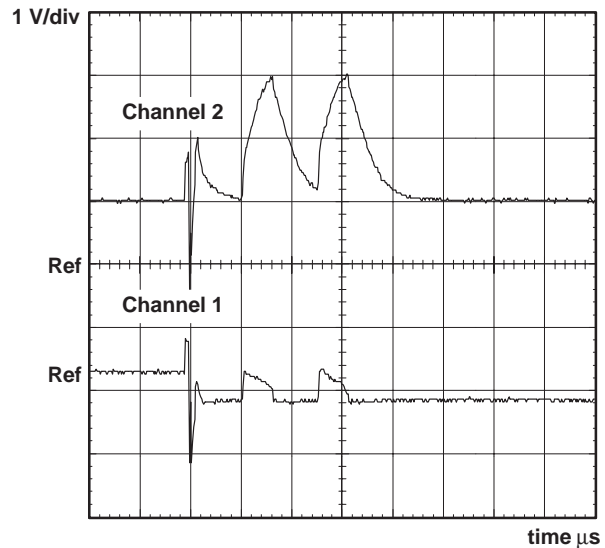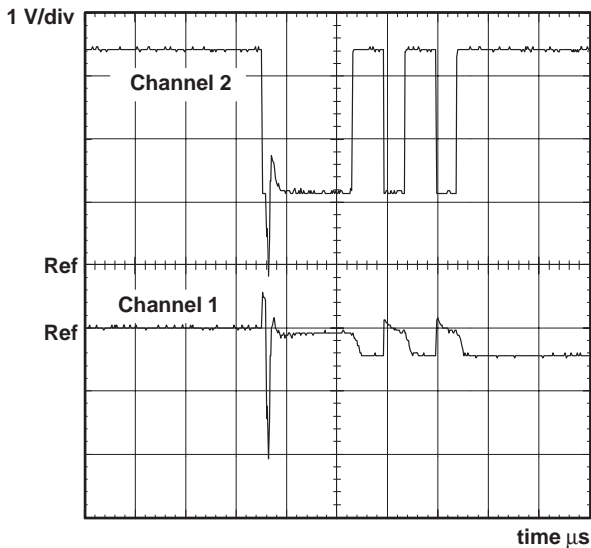
Figure 7. Triac Trigger Signals (Continued)

Figure 7 shows the voltage peaks on the TRIAC gate and the I/O pin when the TRIAC is driven at power-line voltage. With the passive circuit shown, the peaks are much lower than when using the transistor driver. The MSP430 can withstand negative voltage peaks much better than peaks over VCC because the negative peaks are drained away through the big substrate. Therefore, the driving signal is active-high, which results in small positive voltage peaks and larger negative voltage peaks. Otherwise it would generate peaks much higher than V_{CC}, peaks which are more dangerous to the MSP430.

4.6   Phase-Angle Generation

To generate the right phase-angle, the zero-crossing of the line voltage must be detected. This example uses a Zener diode with a series resistor to perform the zero-crossing detection. Every edge of the resulting signal triggers the timer-capture register CCR2 and stores the timer value as a starting point for the phase-shift control. Adding a certain number representing the phase-shift to this 16-bit captured timer value, and reconfiguring it to a compare function with interrupt capability generates the fire signal for the triac gate. The output unit can be configured to set the output pin as required when the compare value matches. This feature is used to generate the first trigger pulse for the triac gate. The following two pulses are controlled by the compare interrupt service routine. Using only the 16-bit capture/compare CCR2 register for voltage zero-crossing detection and triac gate firing, the multiplexed I/O pins implemented in the MSP430C11x family and connected to this capture/compare block become necessary. This is shown in Figure 8.
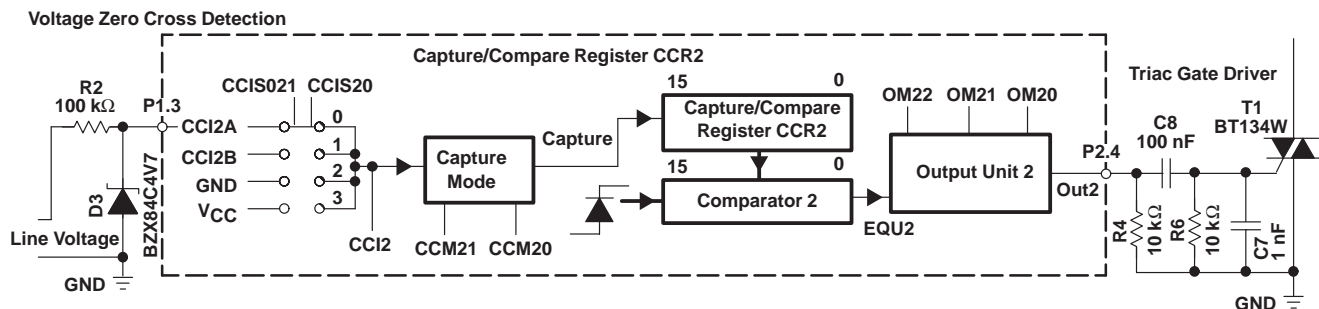
Figure 8. Usage of CCR2

4.7   Phase-Angle Calculation

To eliminate timer clock variations due to voltage/temperature drift, the timer clock is synchronized with the line voltage zero-crossing. To reduce error caused by the threshold of the Zener diode, a whole supply cycle (2 zero-crossings) is measured. This number is then divided by 512, which results in 256 taps per half wave.

EXAMPLE:
The timer runs at 1 MHz, line frequency is 60 Hz:
Timer value after 2 voltage zero-crossings: VZC_2delta = 16,667

$$Speed\_Tap = \frac{VZC\_2delta}{512} = 32.553$$

With this division, 256 discrete phase-angles are possible. This results in a resolution of 0.7 degrees. Commonly used control algorithms like PI or PID have 8-bit output data which can be fed directly into this application software. The actual value to be stored in the compare register which controls the triac fire-pulse is calculated as follows:

Fire_Angle = Speed $\times$ Speed_Tap
                    ; Fire Angle: Compare value to generate triac fire pulse
                    ; Speed: 8-bit output value from control algorithm

To reduce component cost, the phase-shift between current and voltage is not measured but can be adjusted by defining the Speed_Limit constant. This constant has the same scale factor as the 8-bit phase-angle (1 corresponds to 0.7 degrees). The maximum and minimum phase angles are calculated by the following formulas:

Speed_Min = Speed_Tap $\times$ Speed_Limit
Speed_Max = Speed_Tap $\times$ (256 – Speed_Limit)

4.8 Oscillator Adjustment

Since the oscillator in the MSP430P112 device is a free running RC type, the nominal frequency after reset could vary between 650 kHz and 1.2 MHz. This frequency is highly dependent on the value of the time-constant of the integrated resistor/capacitor network. The nominal frequency cannot be accurately specified due to manufacturing variations. To obtain a more accurate frequency value during operation, the frequency must be adjusted either by software or by connecting a small external tolerance-resistor from pin R_{OSC} to V_{CC}.

In this application, the frequency is adjusted during the initialization routine by reference to the power line frequency. The desired number of MCLK cycles, which is also used for the timer and watchdog, should be 10,000 per power line voltage half-wave. This results in 1 MHz for a 50-Hz ac-system, and 1.2 MHz for a 60-Hz ac-system, and offers adequate performance with sophisticated control algorithms. However, the operating frequency could be increased to up to 5 MHz just by using software.

4.9 Interference Reduction

The load has inductive characteristics in almost every triac application. To protect the circuitry and the power line from fast triac-generated voltage surges, which may cause interference on low and medium radio frequencies, a filter can be inserted between the triac and the load as shown in Figure 9.
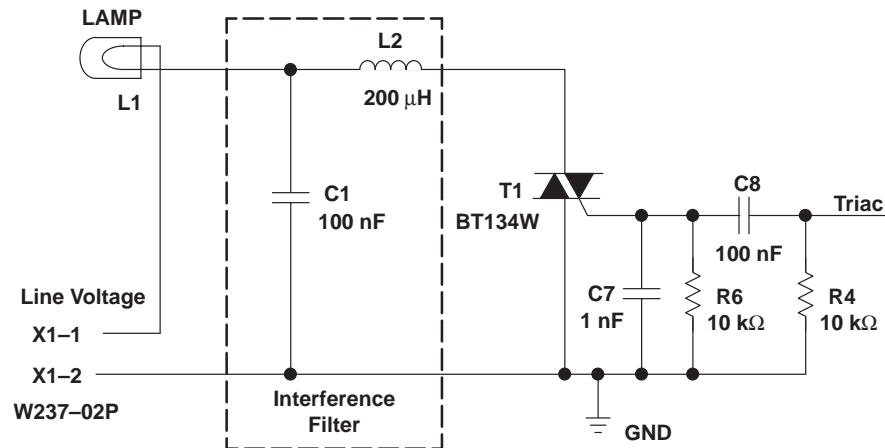
$$H(s) = \frac{1}{s^2 LC + 1} \Rightarrow \omega_0 = \sqrt{\frac{1}{LC}} \Rightarrow f_0 = \frac{1}{2\pi} \times \sqrt{\frac{1}{LC}} = 35.6 \ kHz$$

Figure 9. Interference Filter

The filter used has a double pole at 35 kHz, which is sufficiently higher than the power line frequency to prevent additional phase-shift or attenuation of the power line voltage.

Another way to prevent distortion to the power line is to use wave packaging. This could be accomplished by keeping the same hardware and modifying the software. With this approach the interference filter previously described may be removed.

4.10 Modification for High Speed

In motor-control applications where high speed is a requirement the phase angle must be very small. With highly-inductive motors the phase-shift between current and voltage must be higher than the phase angle to achieve speed desired. In this case the triac must be triggered again after the current crosses zero (the triac will be switched off). The circuitry in Figure 10 can be used to detect current zero-crossings.
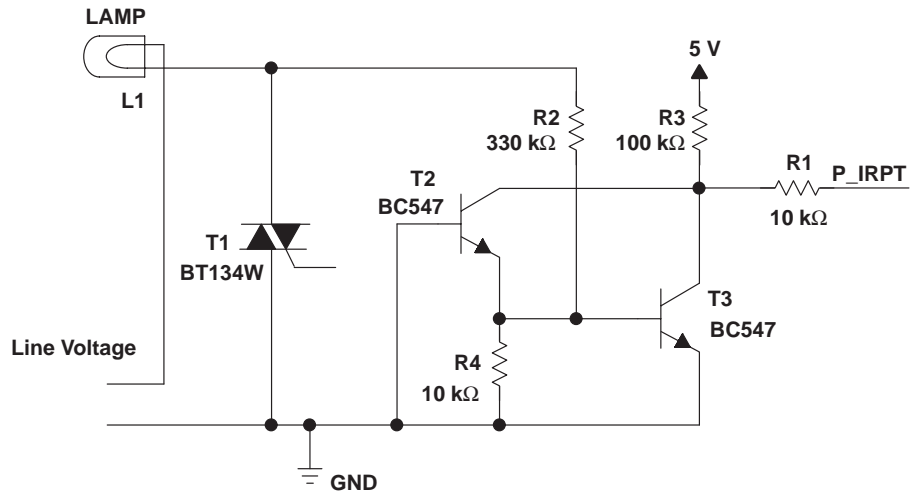
Figure 10. Current Zero-Cross Detection

When current is flowing the triac is on, resulting in 0 V at the base of T3. T2 and T3 are off. If no current is flowing, the triac is off, resulting in a positive or negative voltage at the base of T3. T2 or T3 is on. The output signal P_IRPT could be connected to an input port with interrupt capability to detect current switching conditions.

4.11 Enhanced Security

Although the MSP430 family has a power-on reset circuit on the chip, it could become necessary to add an external supply voltage supervisor (SVS) or even a low-drop output regulator (LDO) behind the Zener diode of the power supply to detect brown-out or power-fail conditions. Listed below are some popular supervisory circuits.

DEVICE	DESCRIPTION
TLC7703	3-V supply voltage supervisor
TLC7705	5-V supply voltage supervisor
TPS3823–33	3.3-V supply voltage supervisor + watchdog timer
TPS3823–50	5-V supply voltage supervisor + watchdog timer
TPS3705–33	3.3-V supply voltage supervisor + power fail + watchdog timer
TPS3705–50	5-V supply voltage supervisor + power fail + watchdog timer

5 Conclusions

The MSP430C11x family makes a 16-bit CPU controlled triac application possible in terms of system cost. The excess performance capability may be used in sophisticated control algorithms or communication tasks. Even with no additional tasks to be performed, the CPU could be set into sleep-mode to reduce current even further. Wake-up from low-power 3-mode (I_{CC}~2 mA) is accomplished within 2 μs.

All necessary software for triac control is available and can be easily adapted to any particular requirements.

Appendixes A1 through A3 list some of the software code routines available.

6 References

1. *MSP430 Family Architecture Guide and Module Library Data Book*, 1996
2. *MSP430 Family Software User's Guide*, 1994
3. *MSP430 Family Metering Application Report*, 1997
4. *MSP430 Family Assembler User's Guide*, 1994
5. BT134W Triac Data Sheet Rev. 1.200, Philips Semiconductors
6. TLC7705 Data Sheet, SLCS087I
7. TPS3823 Data Sheet, SLVS165

http://www.ti.com/sc/docs/msp/msp430/msp430.htm

http://www.ti.com

Appendix A   TRIAC Control Software

A.1   MATH.ASM Listing

```
;----------------------------------------------------------------------
; FUNCTION DEF: Mpy
; DESCRIPTION:  unsigned 16x16 multiplication, R12*R13=R10R11
; REGISTER USE: R12 is op1
;               R13 is op2
;               R10 is result msword
;               R11 is result lsword
; CALLS:    -
; ORIGINATOR:  Anton Muehlhofer
; DATE:  12. March. 98
;----------------------------------------------------------------------
Mpy         push   R5
            clr    R10   ; clr result msw
            clr    R11   ; clr result lsw
            mov    #16,R5
Mpy_Loop    rla    R11   ; previous result*2
            rlc    R10   ; cy to msd
            rla    R13   ; next bit->cy
            jnc    Mpy_L$1
            add    R12,R11
            adc    R10
Mpy_L$1 dec R5
            jnz    Mpy_Loop
            pop    r5
            ret
;----------------------------------------------------------------------
; FUNCTION DEF: Div
; DESCRIPTION:  unsigned 32/16 division, R12|R13 / R14 = R15, Remainder
in R12
; REGISTER USE: R12 is dividend high word
;               R13 is dividend low word
;               R14 is divisor
;               R15 is result
;               R11 is counter
; CALLS:    -
; ORIGINATOR: Metering Application Report
;----------------------------------------------------------------------
Div         clr    R15
            mov    #17,R11
Div_L$1     cmp    R14,R12
            jlo    Div_L$2
```

```
                        sub     R14,R12
            Div_L$2     rlc     R15
                        jc      Div_L$4
                        dec     R11
                        jz      Div_L$4
                        rla     R13
                        rlc     R12
                        jnc     Div_L$1
                        sub     R14,R12
                        setc
                        jmp     Div_L$2
            Div_L$4     ret
```

A.2 110.INC Listing

```
;=======================================================================
; Control register address definitions
;=======================================================================
IE1             .equ    0h
IFG1            .equ    02h
;-----------------------------------------------------------------------
; IE1 bit definitions
;-----------------------------------------------------------------------
WDTIE           .equ    01h         ; Watchdog interrupt enable
OFIE            .equ    02h         ; Oscillator fault interrupt
enable
;-----------------------------------------------------------------------
; IFG1 bit definitions
;-----------------------------------------------------------------------
WDTIFG          .equ    01h         ; Watchdog interrupt flag
OFIFG           .equ    02h         ; Oscillator fault interrupt
flag
NMIIFG          .equ    10h         ; Signal at RST/NMI pin
;=======================================================================
; Status flag bit definitions
;=======================================================================
GIE             .equ    08h
CPUOFF          .equ    10h
OSCOFF          .equ    20h
SCG0            .equ    40h
SCG1            .equ    80h
;=======================================================================
; System Clock Control Register address definition
;=======================================================================
DCOCTL          .equ    056h
```

```
BCSCTL1             .equ    057h
BCSCTL2             .equ    058h
;----------------------------------------------------------------------
; BCSCTL1 bit definition
;----------------------------------------------------------------------
XT2OFF              .equ    80h
XTS                 .equ    40h
XT5V                .equ    08h
;----------------------------------------------------------------------
; BCSCTL2 bit definition
;----------------------------------------------------------------------
SELS                .equ    08h
DCOR                .equ    01h
;======================================================================
; Port 1 Control Register address definition
;======================================================================
P1IN                .equ    020h
P1OUT               .equ    021h
P1DIR               .equ    022h
P1IFG               .equ    023h
P1IES               .equ    024h
P1IE                .equ    025h
P1SEL               .equ    026h
;======================================================================
; Port 2 Control Register address definition
;======================================================================
P2IN                .equ    028h
P2OUT               .equ    029h
P2DIR               .equ    02Ah
P2IFG               .equ    02Bh
P2IES               .equ    02Ch
P2IE                .equ    02Dh
P2SEL               .equ    02Eh
;======================================================================
; Timer A Control Register address definition
;======================================================================
TAIV                .equ    12Eh
TACTL               .equ    160h
TAR                 .equ    170h
CCTL0               .equ    162h
CCR0                .equ    172h
CCTL1               .equ    164h
CCR1                .equ    174h
```

```
CCTL2               .equ    166h
CCR2                .equ    176h
;====================================================================
; Timer A Control Register bit definition
;====================================================================
CAP                 .equ    0100h
OUT                 .equ    0004h
CCIFG               .equ    0001h
P0IN0               .equ     001h
;====================================================================
; Watchdog Control Register address and bit definition
;====================================================================
WDTCTL              .equ    120h            ; watchdog control register
address
WDTPW               .equ    5A00h           ; password for watchdog access
WDTCL               .equ    8h              ; bit position for watchdog reset
WDTHold             .equ    80h
```

A.3   TRIAC112.ASM Listing

```
;*********************************************************************
; File Name: TRIAC112.asm
; Project:   MSP430c112 TRIAC demo
; Originator:Anton Muehlhofer   (Texas Instruments Deutschland)
;
; Target Sys:MSP430x110
;
; Description:  Main function is to capture the voltage zero cross and
;               generate a TRIAC fire pulse with variable phase angle.
;               The phase angle is determined by a poti measured by
;               Timer A input capture (Single slope approach).
;
; Status: tested with MSP430E112.
;
; Last Update:  July 16, 1998
;*********************************************************************
;-------------------------------------------------------------------
; Development Environment
;-------------------------------------------------------------------
                .length 120     ; 120 lines per page


; define interrupt vector table start address
Ivecs         .set   0FFE0h
; define Stack pointer and available RAM
RAM_Start     .set   00200h    ; Free Memory startadress
RAM_End       .set   00300h    ; RAM endadress
SP_Start      .set   00300h    ; stackpointer
EPROM_Start   .set   0f000h    ; start of 4k EPROM
; include available peripheral port addresses
                .include 110.inc
;--- RAM allocation for global word variables
                .bss   dummy,0,220h
                .even
                .bss   VZC_LastCap,2; last voltage zero cross capture
                                   ; value
                .bss   VZC_delta,2  ; timer value for half voltage wave
                .bss   VZC_2delta,2 ; timer value for voltage period
                .bss   Fire_Angle,2 ; timer value for TRIAC fire angle
                .bss   Speed_Tap,2  ; timer value for 1*256 of possible
                                   ; 256 speeds
                                   ; 16 bit value for accurate
measurement
                .bss   Speed_Min,2  ; timer value for TRIAC fire angle
                                   ; min. speed
```

```
                    .bss   Speed_Max,2    ; timer value for TRIAC fire angle
                                          ; max. speed
                    .bss   ADC_CAP0,2     ; start capture value for ADC measurement
                    .bss   ADC_CAP1,2     ; end capture value for ADC measurement
                    .bss   ADC_PotiA_CCR,2 ; capture value for PotiA measurement
                    .bss   ADC_PotiB_CCR,2 ; capture value for PotiB measurement
                    .bss   ADC_PotiC_CCR,2 ; capture value for PotiC measurement

        ;--- RAM allocation for global byte variables
                    .bss   Speed,1        ; nominal motor speed 1..255
                    .bss   Pulse_Ctr,1    ; counter of TRIAC fire pulses
                    .bss   Status,1       ; Status byte containing status bits
                    .bss   Cntr_1s,1      ; 1 second counter
                    .bss   TStat_10ms,1   ; status for task management
        ;------------------------------------------------------------------------
        ; Pin assignment
        ;------------------------------------------------------------------------
        VZC           .equ   008h         ; zero cross input is P1.3 / cc2
        TRIAC_gate    .equ   010h         ; compare output cc2 out P2.4
        TRIAC_gate_A  .equ   008h         ; alternative TRIAC output pin P2.3
                                          ; inverse to TRIAC_gate_A
        ADC_PotiA     .equ   01h          ; P2.0 Poti A
        ADC_PotiB     .equ   01h          ; P1.0 Poti B
        ADC_PotiC     .equ   02h          ; P2.1 Poti C
        ADC_Load      .equ   04h          ; P2.2 TA CCR0 inp. capture
        ;------------------------------------------------------------------------
        ; Constant definition
        ;------------------------------------------------------------------------
        Pulse_Max     .equ   3            ; # of TRIAC fire pulses
        Speed_Limit   .equ   40           ; limit for max. and min. speed to take
                                          ; care of  phase difference between
                                          ; u and i
        ;------------------------------------------------------------------------
        ; Status flag definition
        ;------------------------------------------------------------------------
        Task_Ovr      .equ   01h
        ADC_EOC       .equ   02h


                      .sect  "MAIN",EPROM_Start
        ;------------------------------------------------------------------------
        ; include external subroutines
        ;------------------------------------------------------------------------
                      .include math.asm
        ;------------------------------------------------------------------------
```

```
; Reset : Initialize processor
;----------------------------------------------------------------------
RESET
          MOV    #SP_Start,SP          ; initialize stackpointer
; initialize Watchdog
          mov    #(WDTCL+WDTPW),&WDTCTL ; Watchdog overflow rate is
                                        ; tMCLKx2^15 (32ms@1MHz)
                                        ; and reset WD
; output SMCLK at p1.4 for test purposes
          bis.b  #010h,P1SEL
          bis.b  #010h,P1DIR
; configure oscillator ( 1 MHz nominal frequency )
          bis    #OSCOFF,SR            ; switch XT1 off

; Clear Special Function Registers
      clr.b  IE1
      clr.b  IFG1            ; clears oscillator fault and WD interrupt
                             ; enable

; configure Timer A
      mov    #0204h,TACTL
                             ; counts up continuous
                             ; no interrupt generation at overflow
                             ; timer cleared
                             ; timer is stopped, need input selection
      bic    #04h,TACTL      ; release timer clear
      bis    #20h,TACTL      ; start timer with MCLK
; configure TRIAC fire pin / set TRIAC off
      bic.b  #TRIAC_gate,P2OUT    ; TRIAC gate output is digital I/O
      bis.b  #TRIAC_gate,P2DIR    ; set output pin to low -> clear
                                  ; TRIAC
      mov    #00h,CCTL2           ; set OUT2 low (TRIAC off)
      bis.b  #TRIAC_gate,P2SEL    ; select TRIAC gate to cmp output pin
      bic.b  #TRIAC_gate_A,P2OUT  ; altern. TRIAC gate output is
                                  ; digital I/O low
      bis.b  #TRIAC_gate_A,P2DIR  ; set output pin to low -> clear
                                  ; TRIAC

; configure VZC pin; comp/capt block 2 for zero cross and triac fire
      bic.b  #VZC,P1DIR           ; input capture voltage zero
                                  ; cross is P1.3
      mov    #1100100100100000b,CCTL2
                                  ; CC2 is in capture mode
                                  ; zero cross capture interrupt pin
                                  ; P1.3 (CCI2A)
                                  ; triac fire pulse on cmp output
                                  ; pin P2.4 (OUT2)
```

```
                                ; configure as synchronous cap mode
                                ; both edges triggers capture
                                ; cmp will set (TRIAC on)
                                ; disables cap2 interrupt

        bis.b  #VZC,P1SEL       ; input capture voltage zero cross is
        P2.4


        ; configure comp/capt block 0 for ADC measurement
        mov    #1001100100110100b,CCTL0
                                ; negative edge triggers capture
                                ; input CCI0B is selected
                                ; synchronized capture
                                ; capture mode
                                ; output mode when cmp matches
                                ; interrupt enabled
                                ; when in compare mode output
                                ; value is set
                                ; when match
        ; configure ADC measurement pins
        bis.b  #ADC_PotiC,P2OUT ; load C through Poti C
        bis.b  #ADC_PotiC,P2DIR
        bic.b  #ADC_PotiC+ADC_PotiA,P2SEL
        bic.b  #ADC_PotiB,P1OUT ; configure Poti B
        bic.b  #ADC_PotiB,P1DIR
        bic.b  #ADC_PotiB,P1SEL
        bic.b  #ADC_PotiA,P2OUT ; configure Poti A
        bis.b  #ADC_Load,P2SEL  ; configure capture input
        bic.b  #ADC_Load,P2DIR

        ; configure unused io pins to output low
        bic.b  #11100110b,P1SEL
        bic.b  #11100110b,P1OUT ; set unused pins port 1
        bis.b  #11100110b,P1DIR ; to output low
        bic.b  #11100000b,P2SEL ; set unused pins port 2
        bic.b  #11100000b,P2OUT ; to output low
        bis.b  #11100000b,P2DIR

        ; initialize global Variables for Task management
        clr.b  Status
        clr.b  TStat_10ms
        clr.b  Cntr_1s
        mov    #(WDTCL+WDTPW),&WDTCTL ; reset WD
        ; initialize global Variable for TRIAC control
        clr.b  Pulse_Ctr
```

```
; initialize global Variables VZC_LastCap
          mov    #1100100100100000b,CCTL2   ; capture works for both
edges
          call   #Get_CCR2
          mov    R5,VZC_LastCap
          mov    #(WDTCL+WDTPW),&WDTCTL      ; reset WD
; initialize global Variables VZC_delta and VZC_LastCap
          call   #Get_CCR2
          mov    R5,R6
          sub    VZC_LastCap,R5
          mov    R5,VZC_delta
          mov    R6,VZC_LastCap
          mov    #(WDTCL+WDTPW),&WDTCTL      ; reset WD
Init_DCO_Ctrlmov #5,R10                      ; load loop counter to
                                             ; get stable system
; initialize global Variables VZC_2delta, VZC_delta and VZC_LastCap
Init_VZC  call   #Get_CCR2
          mov    R5,R6
          sub    VZC_LastCap,R5
          mov    VZC_delta,VZC_2delta
          add    R5,VZC_2delta
          mov    R5,VZC_delta
          mov    R6,VZC_LastCap
          mov    #(WDTCL+WDTPW),&WDTCTL      ; reset WD
; adjust osciallator to provice 10k+-1k MCLK cycles per mains voltage
  half wave
; f(VAC)=50 Hz -> MCLK = 1MHz+-100kHz
; f(VAC)=60 Hz -> MCLK = 1.2MHz+-120kHz
;      -> VZC_delta = 10000+-1000
VZC_nom   .equ   10000
VZC_var   .equ   1000


          mov    VZC_delta,R5
          cmp    #VZC_nom-VZC_var,R5
          jl     Inc_DCO                     ; MCLK is too low
          cmp    #VZC_nom+VZC_var,R5
          jge    Dec_DCO                     ; MCLK is too high
          dec    R10                         ; check if system is stable
          jz     CCR2_norm                   ; yes, start with normal
                                             ; operation
          jmp    Init_VZC                    ; no, do another loop
Inc_DCO   add.b  #20h,&DCOCTL
          jmp    Init_DCO_Ctrl
Dec_DCO   sub.b  #20h,&DCOCTL
          jmp    Init_DCO_Ctrl
```

```
; initialize CCR2 for normal operation
CCR2_norm    bic   #0001,CCTL2              ; clear interrupt flag
       mov   #1100100100100000b,CCTL2       ; capture works for both edges
       mov   #(WDTCL+WDTPW),&WDTCTL         ; reset WD
; calculate Speed_Min, Speed_Max, Speed_Tap
       call  #Calc_Limits
; load slowest speed
       add   Speed_Min,CCR2
       mov   Speed_Min,Fire_Angle


; start TRIAC fire generation with next CMP2 equal
       bic   #CAP,CCTL2                      ; configure CCTL2 for cmp function
       bis   #0010h,CCTL2                    ; enable cctl2 interrupt
       mov   #(WDTCL+WDTPW),&WDTCTL          ; reset WD
       eint                                  ; global enable all interrupts
;----------------------------------------------------------------------
; Main loop
;----------------------------------------------------------------------
       .newblock
mainloop
       xor.b #20h,P2OUT
       bit.b #Task_Ovr,Status
       jz    mainloop
$1     mov   #(WDTCL+WDTPW),&WDTCTL          ; Watchdog overflow rate is
                                             ; tMCLKx2^15
       call  #Task_10ms
       jmp   mainloop
;----------------------------------------------------------------------
; Task management for tasks called ervery 10 ms (voltage zero cross)
;----------------------------------------------------------------------
Task_10ms mov.b  TStat_10ms,R5
          mov.b  Tbl_10ms(R5),R5
          add    R5,PC
Tbl_10ms
          .byte  PotiA1-Tbl_10ms
          .byte  PotiA2-Tbl_10ms
          .byte  PotiB1-Tbl_10ms
          .byte  PotiB2-Tbl_10ms
          .byte  PotiC1-Tbl_10ms
          .byte  PotiC2-Tbl_10ms
          .byte  PotiAC-Tbl_10ms
          .byte  Last-Tbl_10ms
PotiA1    bis.b #ADC_PotiA,P2DIR            ; discharge
```

```
                jmp     Dcharge
PotiA2          bit.b   #ADC_EOC,Status      ; check if new capture is available
                jz      ADC_Ret              ; if not, wait for another task
                                             ; timer frame
                bic.b   #ADC_PotiA,P2DIR     ; stop discharge
                mov     #ADC_PotiA_CCR,R6    ; address of 1. result
                jmp     Charge
PotiB1          bis.b   #ADC_PotiB,P1DIR
                jmp     Dcharge
PotiB2          bit.b   #ADC_EOC,Status      ; check if new capture is
available
                jz      ADC_Ret              ; if not, wait for another task
                                             ; timer frame
                bic.b   #ADC_PotiB,P1DIR     ; stop discharge
                jmp     Charge
PotiC1          bic.b   #ADC_PotiC,P2OUT
                bis.b   #ADC_PotiC,P2DIR
                jmp     Dcharge1
PotiC2          bit.b   #ADC_EOC,Status      ; check if new capture is available
                jz      ADC_Ret              ; if not, wait for another task timer
                                             ; frame
                bic.b   #ADC_PotiC,P2DIR     ; stop discharge
                jmp     Charge
PotiAC          call    #Calc_Limits
                call    #Calc_Poti
                mov     R15,R10
                call    #Calc_Speed
                mov.b   R15,Speed
                jmp     ADC_End
Last            clr.b   TStat_10ms
                jmp     ADC_Ret
Dcharge         bic.b   #ADC_PotiC,P2DIR     ; stop charging
Dcharge1        mov     TAR,ADC_CAP0         ; store initial Timer value
                jmp     ADC_End
Charge          bis.b   #ADC_PotiC,P2OUT
                bis.b   #ADC_PotiC,P2DIR     ; start charging
                mov     ADC_CAP1,R5          ; build timer difference
                sub     ADC_CAP0,R5
                mov     R5,0(R6)             ; result address
                incd    R6
                bic.b   #ADC_EOC,Status
ADC_End         inc.b   TStat_10ms
ADC_Ret         bic.b   #Task_Ovr,Status
                ret
```

```
;------------------------------------------------------------------------
; FUNCTION DEF: Calc_Poti
; DESCRIPTION:  Speed=(((PotiB-PotiC)*256)*2 / (PotiA-PotiC) ) /2
;      Speed = [0..255]
; REGISTER USE: R12, R13, R14, R15, R16
; CALLS:    -
; ORIGINATOR:Anton Muehlhofer
; DATE:       16. March. 98
;------------------------------------------------------------------------
Calc_Poti
        mov       ADC_PotiB_CCR,R7       ; PotiB-PotiC
        sub       ADC_PotiC_CCR,R7
        swpb      R7                     ; (PotiC-PotiA)*256
        mov       R7,R12
        bic       #0ff00h,R12
        mov       R7,R13
        bic       #00ffh,R13
        rla       R13
        rlc       R12
        mov       ADC_PotiA_CCR,R14      ; (PotiB-PotiA)
        sub       ADC_PotiC_CCR,R14
        call      #Div                   ; result is in R15
        rra       R15
        cmp       #00ffh,R15             ; limit result to 0-255
        jlo       Calc_Poti_end
        mov       #00ffh,R15
Calc_Poti_end    inv.b  R15
        ret
;------------------------------------------------------------------------
; FUNCTION DEF: Calc_Limits
; DESCRIPTION:   calculates runtime variable speed parameters
;      divide whole speed range into 256 taps
;      Speed_Tap = (VZC_2delta/256/2)*256 (for accurate calculations)
;      Speed_Min = speed_tap * speed_limit(0..60) / 256
;      Speed_Max = speed_tap * (256 - speed_limit) / 256
; REGISTER USE: R5
; CALLS:    -
; ORIGINATOR:Anton Muehlhofer
; DATE:    25. March. 98
;------------------------------------------------------------------------
Calc_Limits
; speed_tap = VZC2_delta/2
        mov       VZC_2delta,R5
```

```
        rra     R5                          ; /2
        mov     R5,Speed_Tap
; speed_max = (speed_tap * speed_limit(0..60))/256
        mov     Speed_Tap,R12
        mov.b   #Speed_Limit,R13
        call    #Mpy                        ; result is R10|R11
        swpb    R11
        bic     #0ff00h,R11
        swpb    R10
        bic     #00ffh,R10
        add     R11,R10
        mov     R10,Speed_Max
; speed_min = speed_tap * (256 - speed_limit) / 256
        mov     #256-Speed_Limit,R12
        mov     Speed_Tap,R13
        call    #Mpy
        swpb    R11
        bic     #0ff00h,R11
        swpb    R10
        bic     #00ffh,R10
        add     R11,R10
        mov     R10,Speed_Min
        ret
;-----------------------------------------------------------------------
; FUNCTION DEF: Calc_Speed
; DESCRIPTION:   input: 8 bit value Speed for motorspeed
;       output: 16 bit timer value Fire_Angle
;       Fire_Angle = Speed * Speed_Tap
;       Speed_Min = speed_tap * speed_limit(0..60)
;       Speed_Max = speed_tap * (256 - speed_limit)
; REGISTER USE: R10, R11, R12, R13
; CALLS:  Mpy
; ORIGINATOR:Anton Muehlhofer
; DATE:      13. March. 98
;-----------------------------------------------------------------------
Calc_Speed
; Fire_Angle = Speed * Speed_Tap / 256
        mov     Speed_Tap,R12
        mov.b   Speed,R13
        inv.b   R13
        call    #Mpy                    ; 32 bit result is in R10|R11
        swpb    R11
        bic     #0ff00h,R11
```

```
        swpb    R10
        bic     #00ffh,R10
        add     R11,R10
        cmp     Speed_Max,R10
        jhs     Calc_Speed_L$1
        mov     Speed_Max,Fire_Angle
        ret
Calc_Speed_L$1  cmp Speed_Min,R10
        jlo     Calc_Speed_L$2
        mov     Speed_Min,Fire_Angle
        ret
Calc_Speed_L$2  mov R10,Fire_Angle
        ret
;----------------------------------------------------------------------
; FUNCTION DEF: Get_CCR2
; DESCRIPTION:  waits for an capture interrupt flag in polling mode and
                provides the captured value in R5
; REGISTER USE: R5
; CALLS:    –
; ORIGINATOR:Anton Muehlhofer
; DATE:      16. July 1998
;----------------------------------------------------------------------
Get_CCR2  bic    #0001,CCTL2        ; clear interrupt flag
wait_vzc1 bit    #0001,CCTL2        ; wait for 1. voltage zero cross
                                    ; in polling mode
          jz    wait_vzc1           ; check interrupt flag
          mov   CCR2,R5
          bic   #0001,CCTL2         ; clear interrupt flag
          ret
;----------------------------------------------------------------------
; Voltage Zero Cross Interrupt Service Routine
;----------------------------------------------------------------------
Int_P2
Int_P1    reti
;----------------------------------------------------------------------
; FUNCTION DEF: void Int_TA_CC0( void )
; DESCRIPTION:
; REGISTER USE:
; CALLS:    –
; ORIGINATOR:Anton Muehlhofer
; DATE:    16. March. 98
;----------------------------------------------------------------------
Int_TA_CC0    mov    CCR0,ADC_CAP1
              bis.b #ADC_EOC,Status
              reti
```

```
;-----------------------------------------------------------------------
; Timer A Capture/Compare Interrupt Service Routine
;-----------------------------------------------------------------------
Int_TA_IV    add    &TAIV,PC         ; read TA interrupt vector and
                                     ; clear int flag
             reti
             jmp    HCCR1
             jmp    HCCR2
             jmp    Int_TA_end       ; for HCCR3
             jmp    Int_TA_end       ; for HCCR4
Int_TA_end reti
;-----------------------------------------------------------------------
; cap2/cmp2 interrupt  -- TRIAC Gate fire angle --
; used by MSP430x11x
;-----------------------------------------------------------------------
HCCR2
             bit    #CAP,CCTL2       ; check wether capt or cmp has
                                     ; triggered intrpt
             jnz    Cap2_Isr

Cmp2_Isr                            ; TRIAC is already on
             xor    #CAP,CCTL2       ; toggle function capture <-> compare
             bic    #CCIFG,CCTL2     ; reset interrupt flag
             eint
Fire_TRIAC   bis    #OUT,CCTL2       ; set TRIAC gate, set output high
             bis.b  #TRIAC_gate_A,P2OUT ; set altern. TRIAC gate, set
                                        ; output high
             bic    #00e0h,CCTL2     ; switch to output mode by out
                                     ; bit only
             nop
             nop
             bic    #OUT,CCTL2       ; clear TRIAC gate, set output low
             bic.b  #TRIAC_gate_A,P2OUT ; clear altern. TRIAC gate,
                                        ; set output low
             nop
             nop
             inc.b  Pulse_Ctr        ; update TRIAC gate pulse counter
             cmp.b  #Pulse_Max,Pulse_Ctr
             jlo    Fire_TRIAC       ; if a complete series of TRIAC
                                     ; gate pulses
             clr.b  Pulse_Ctr
             bis    #00a0h,CCTL2     ; switch back to compare output mode
             reti
```

```
        ;-----------------------------------------------------------------------
        ; Voltage Zero Cross Interrupt Service Routine
        ; used by MSP430x11x
        ;-----------------------------------------------------------------------
        Cap2_Isr
               push  R5
               xor   #CAP,CCTL2         ; toggle function capture <-> compare
               mov   CCR2,R5            ; measure voltage half period
               sub   VZC_LastCap,R5     ; R5 contains voltage half period
               mov   VZC_delta,VZC_2delta
               add   R5,VZC_2delta      ; actualize voltage full period
               mov   R5,VZC_delta
               mov   CCR2,VZC_LastCap
               add   Fire_Angle,CCR2    ; load new fire angle into cmp register
               eint                     ; reenable interrupts for nesting

               bis.b #Task_Ovr,Status   ; initiate Task proceeding
               inc.b Cntr_1s            ; update sw_counter
               cmp.b #100,Cntr_1s
               jlo   Cap2_End
               clr.b Cntr_1s
               bic   #CCIFG,CCTL2       ; reset interrupt flag
        Cap2_End  pop   R5
                  reti
        ;-----------------------------------------------------------------------
        ; CCR1 Interrupt Service Routine     - unused -
        ;-----------------------------------------------------------------------
        HCCR1    reti
        ;-----------------------------------------------------------------------
        ; WD Timer Interrupt Service Routine  - unused -
        ;-----------------------------------------------------------------------
        Int_WDT_T reti                   ; Watchdog / Timer
        ;-----------------------------------------------------------------------
        ; Interrupt vectors
        ;-----------------------------------------------------------------------
               .sect  "Int_Vect",Ivecs
               .word RESET               ; no source
               .word RESET               ; no source
               .word Int_P1              ; Port1
               .word Int_P2              ; Port2
               .word RESET               ; no source
               .word RESET               ; no source
               .word RESET               ; no source
```
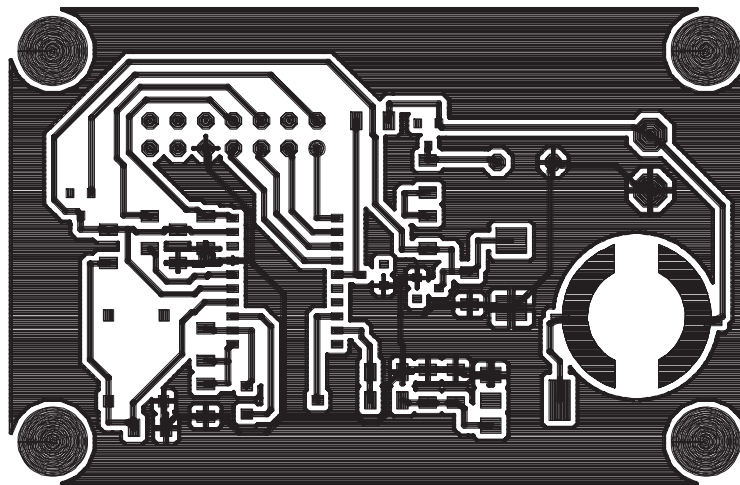
```
        .word  RESET              ; no source
        .word  Int_TA_IV          ; Timer A
        .word  Int_TA_CC0          ; Timer A
        .word  Int_WDT_T          ; Watchdog/Timer, Timer mode
        .word  RESET              ; no source
        .word  RESET              ; no source
        .word  RESET              ; no source
        .word  RESET              ; NMI, Osc. fault
        .word  RESET              ; POR, ext. Reset, Watchdog
        .end
```

A.4 PCB Layout

A.5 Component Placement