Implementing a Lamp Dimmer with an HC908Q Family MCU

by: Jefferson Daniel de Barros Soldera
Andre Luis Vilas Boas
Alfredo Olmos
Marcus Espindola
Brazil Semiconductor Technology Center — BSTC/SPS

Introduction

Many homes have lamps that can be made brighter or dimmer by a control on the on/off switch. This application note describes how to implement a low-cost lamp brightness control or dimmer using a member of the M68HC08 MCU Family. The circuit controls the amount of energy that reaches the bulb during each half-cycle of the AC power line. Moreover, a microcontroller may grant extra automation features to the circuit, such as soft start and programmable timing. Additionally, an application in which a lamp is turned on for a specific amount of time is described. The dimmer circuit implementation requires few external components.
Dimmer Features

- 110 V or 220 V, 60 Hz or 50 Hz supply voltage
- Up to 100 W lamp dimming
- Full wave AC phase control
- No transformer for AC power isolation
- Up/down touch control option
- Customized programmable timer
- Low-cost 8-pin MCU implementation

Control Method

Many homes have lamps that can be made brighter or dimmer by rotating or sliding a control on the on/off switch. Years ago, this was done using a device called a rheostat which consists of a large variable resistor. To control the amount of energy going to the light, the rheostat had to dissipate the excess energy as heat. For example, at half brightness for a 100-watt bulb, approximately 20 W would be converted to heat in the rheostat.

Modern dimmers work in an entirely different way. They use transistor-like devices called triacs to switch on the current to a lamp part way into each half-cycle. Unlike the silicon-controlled rectifier (SCR), the triac can conduct current in both half-cycles when turned on. As soon as it is triggered, the triac will allow the current to flow through the bulb until that current gets to zero, which happens whenever the voltage crosses zero.

The amount of energy that reaches the bulb during each half-cycle depends on how long the control waits before triggering the triac. The longer it waits, the less energy reaches the bulb and it will glow.

Triacs

To successfully apply triacs for power control, an understanding of the triac’s characteristics, ratings, and limitations is imperative.

Figure 1 shows the triac power control principle. Triacs are three-terminal AC semiconductor switches that are triggered into conduction when a low-energy signal is applied to their gate, allowing a full wave AC control. In Figure 1a, terminals MT1 and MT2 are the current-carrying terminals; G is the gate terminal used for triggering the device. To avoid confusion, it has become standard practice to specify all currents and voltages using MT1 as reference.

Triggering a triac requires meeting its gate energy specification. Therefore, the gate should be driven hard and fast to ensure complete gate turn-on, which helps to prevent false triggering. Usually that means a gate current of at least three times the gate turn-on current with a pulse train. It is also important to keep up the input trigger pulse synchronized with the AC power line in order to have a constant conduction angle.

The dashed region in Figure 1b corresponds to the voltage applied to the load. The delay angle is the angle, measured in electrical degrees, during which the device is blocking the line voltage. The period
during which the device is on is called the conduction angle ($\alpha$). Varying $\alpha$ will control the portion of the total AC sine wave applied to the load and, thereby, regulate the power flow to the load.

The main disadvantage of using phase control in triac applications is the generation of electro-magnetic interference (EMI). In incandescent lamps, phase control gives a continuous brightness and good performance.

![Triac Symbol](image1)

![Sine Wave Showing Phase Control Principles](image2)

**Figure 1. Performing Power Control with Triac — Triggering the Device**

**HC908Q Family Features**

High-performance 8-bit HC08 CPU:
- Object-code compatible with Freescale’s 68HC05 architecture for easy migration
- Enables the higher performance required of many 8-bit applications while saving development time — as fast as 125 ns minimum instruction cycle time
- Designed to allow efficient, compact modular coding in assembly or C with full 16-bit stack pointer and stack relative addressing
- Efficient instruction set with multiply and divide that is easy to learn and use

Memory:
- In-application, in-circuit re-programmable FLASH memory (1.5K to 4K bytes)
- 128 bytes of random access memory (RAM)

Peripherals:
- Two-channel, 16-bit timer with selectable input capture, output compare, or PWM
- Trimmable 5% accuracy internal clock oscillator
- Four-channel, 8-bit analog-to-digital converter (ADC) (on the MC68HC908QT2/QT4/QY2/QY4) provides an easy interface to analog inputs such as sensors
Application Description

- Flexible I/Os allow direct drive of LEDs and other circuits to eliminate external drivers and help reduce system cost
- System protection features, including watchdog timer and on-chip low-voltage detect/reset to help reduce cost and increase reliability
- Space-sensitive packages — 8 PDIP, 8 SOIC, 16 PDIP, 16 SOIC, and 16 TSSOP — with more to come as the Family develops

Application Description

The HC908Q Family allows the user to choose the MCU clock source. The microcontroller has three clock source options available. Because this application aims to be low cost, the clock frequency is internally generated. The internal oscillator circuit is designed to provide a clock source with tolerance less than ±25% untrimmed without external components. An 8-bit trimming register allows adjustment to a tolerance of less than ±5%.

Other possible choices, although more expensive and definitely not necessary for this sort of application, would be a built-in RC oscillator module that requires an external resistor connected to the chip. There is also a built-in XTAL oscillator module designed to operate with an external crystal or ceramic resonator to provide an accurate clock source.

The software accomplishes the brightness control by adjusting the conduction angle. The circuit does not require a transformer to supply DC voltage to the MCU. For this reason, the user must use caution during circuit assembly.

Figure 2 illustrates the complete schematic diagram of the dimmer. The supply voltage is connected to the AC line through the capacitor C1 that provides circuit isolation. C1 must be a 1 µF/600 V non-polarized polyester capacitor. The diodes D1 and D2 enforce the half-wave rectification and the capacitor C2 implements the ripple filtering. C3 is a ceramic bypass capacitor located as near as possible to the MCU power pins in order to suppress high-frequency noise. The zener diode helps provide a reasonable regulated voltage, which reduces the rectifier voltage to the desired supply voltage.

R1 and R2 provide the zero-crossing detection to the MCU to synchronize the triac trigger pulses with the AC power line, achieving an accurate control. All four diodes are 1n4007, which allows the circuit to be supplied by 115 V or 240 V AC.
The triac must be chosen according to the required load current. For a 100-W lamp, the load current at 115 V is 0.87 A and at 240 V, it is 0.42 A. Therefore, for the triac MCR22-8, 600 V isolation is a reasonable choice and can be used for both 115 V and 240 V AC. The resistor R3 limits the triac gate current.

All discrete devices in the circuit are not critical and similar devices can be substituted. The user must be careful about reverse voltage and direct current on diodes and zener, isolation voltage of capacitors, and maximum current in the triac.

Two push buttons and a switch are used to set the lamp brightness and turn the circuit on/off. When the up/down control is pressed, the MCU receives a low level and varies the conduction angle by shifting the short pulses that trigger the triac. The down button allows the MCU to apply pulses reducing $\alpha$ and the lamp brightness is reduced continuously. Conversely, the up button is used to increase $\alpha$, which increases lamp brightness.

Because there is no transformer for power-line isolation, the user must be very careful during assembly and testing to provide the appropriate isolation from the AC power line.

Figure 3, Figure 4, and Figure 5 show the power line signal (a), zero-crossing reference signal (b), MCU short pulses triggering the triac (c), and the waveform of the voltage applied to the load at different average powers (d). Notice that only a portion of the sine wave is applied to the load. The average power is proportional to absolute average voltage.
Figure 3. Triac Control with the HC908Q Family

a) The Power Line Signal

b) The Zero-Crossing Reference Signal

c) MCU Short Pulses Triggering the Triac

d) The Load supplied Approximately 75% of the Total Average Power
Figure 4. The Load Supplied at 50% of the Total Average Power

Figure 5. The Load Supplied Approximately 25% of the Total Average Power
Design Customization

This design works for many applications without modification. However, some customers may want to customize its functionality. A few variations for this circuit include:

- Modifying 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.
- Enhance the timer feature to allow the user to choose the period ("on" time).
- 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 a remote computer.
- Add a photo sensor to adjust a given brightness level in a room according to the ambient light.
- Isolate the load by an opto-isolator IC. This provides isolation between the load and the control circuit, especially when high isolation voltage is required. Applications involving industrial controls, vending machines, or motor controls would benefit from this technology. Figure 6 illustrates how to implement this type of modification. T1 must have a secondary of 9 V or 12 V so that the zener operates adequately.
- Another practical application is to keep the load turned on for a certain amount of time. In this case, the load is supplied at full power and the triac is triggered at a desired time. Refer to Figure 7 for a schematic diagram of the timer.

When the start button is pressed, the MCU is reset and enters into run mode. The software counts the desired time (set previously) and enters into stop mode, which turns off the lamp. When the MCU is in stop mode, all internal modules are disabled and the power consumption is negligible. The circuit is kept in this state until the start button is pressed again.

The software was set to a one minute delay time. This is the value used in most applications where it is required that a lamp must be turned on for a short time. However, the software can be easily be changed to set the desired delay time and lamp brightness.

For higher loads (greater than 100 watts), the triac must be changed to accommodate the maximum current and a heat sink might be required. A high-current triac requires a non-negligible gate current and it might not be possible to drive the triac gate directly from an MCU port. In this case, a driver is needed.

**NOTE**

The circuits above control resistive loads only. For inductive loads, it is recommended that the MCU be isolated from the load with opto couplers and that a triac snubber network be adopted as shown in Figure 8.
Implementing a Lamp Dimmer with an HC908Q Family MCU, Rev. 0

Figure 6. Isolated Dimmer with HC908Q Family

Figure 7. Lamp Timer with HC9098Q Family

DANGER
Electrocution Hazard
Software Description

Two software codes were developed for this application note. The first one implements a lamp dimmer; the second one was developed for an application where a lamp is turned on for a specific amount of time.

Lamp Dimmer Source Code

For the lamp dimmer source code, the MCU controls the lamp brightness by adjusting the conduction angle with the timer modulus as illustrated in Figure 9.

The code starts initializing configuration and timer registers, defining ports, and clearing variables and accumulators. The initial timer value is set to have almost the maximum brightness adjusting the constants InitTMODH and InitTMODL.

PTA0 senses the zero-crossing detection circuit. Each time a positive or negative edge is detected, the timer starts to count until the timer module value (composed by TMODH:TMODL) is reached.

When PTA0 recognizes a positive edge, the MCU verifies PTA5 and PTA4.

If PTA5 is in low level, the routine increments the timer modulus value if it is below the upper limit. PTA4 decrements the timer modulus value if it is above the lower limit when applied a low level. When a timer overflow occurs, PTA1 generates a pulse train triggering the triac.
Lamp Timer Source Code

Figure 10 shows the flowchart for a lamp timer.

The code starts by initializing configuration and timer registers, defining ports, and clearing variables and accumulators.

PTA0 senses the zero-crossing detection circuit. Each time a positive or negative edge is detected the timer starts to count until the timer modulus value (composed by $TMODH:TMODL$) is reached. When a timer overflow occurs, PTA1 generates a pulse triggering the triac.
Software Description

A 2-byte counter is incremented at each zero-crossing of 60 Hz (or 50 Hz if it is used) and compared to $CntHcmp$ and $CntLcmp$ constants. These constants may also be changed if the user desires to increase or decrease the timer. A simple formula can be used:

$$CntHcmp : CntLcmp = \frac{t}{1/f}$$  \hspace{1cm} \text{Equation (1)}

Where: $t =$ desired timer, [s]

$$f = \text{line frequency, [Hz]}$$

$CntHcmp:CntLcmp =$ constant values, [decimal]

The user must remember that for a 2-byte counter, the maximum time will be approximately 18 seconds for 60-Hz and 21 seconds for 50-Hz line frequency, according to the equation. Software timing techniques can be used to extend this delay time.

When the defined time is reached, the MCU enters stop mode to minimize current consumption.

The lamp timer turns on again after a reset.

![Flowchart for Lamp Timer](image)

Figure 10. Flowchart for Lamp Timer
Implementing a Lamp Dimmer with an HC908Q Family MCU, Rev. 0

Freescale Semiconductor

13
Software Description

;*******************************************************************************************
;* SUBROUTINES
;* This part includes subroutines
;*******************************************************************************************

org FlashStart

;*******************************************************************************************
;* Table used for timer value after zero-crossing detection
;* This table uses indexed addressing mode
;*******************************************************************************************

LSBTimer:   dc.b $73;
dc.b $90;
dc.b $AD;
dc.b $CA;
dc.b $E7;
dc.b $04;
dc.b $21;
dc.b $3E;
dc.b $5B;
dc.b $78;
dc.b $95;
dc.b $B2;
dc.b $CF;
dc.b $EC;
dc.b $09;
dc.b $26;
dc.b $43;
dc.b $60;
dc.b $7D;
dc.b $9A;
dc.b $B7;
dc.b $D4;
dc.b $F1;
dc.b $0E;
dc.b $2B;
dc.b $48;
dc.b $65;
dc.b $82;
dc.b $9F;
dc.b $BC;
dc.b $D9;
dc.b $F6;

MSBTimer:   dc.b $03;
dc.b $04;
dc.b $05;
dc.b $06;
dc.b $07;
dc.b $09;
dc.b $0A;
dc.b $0B;
dc.b $0C;
dc.b $0D;
dc.b $0E;
dc.b $0F;
dc.b $10;
dc.b $11;
dc.b $13;
dc.b $14;
dc.b $15;
dc.b $16;
dc.b $17;
dc.b $18;
dc.b $19;
DC.B $1A;
dc.b $1B;
dc.b $1D;
dc.b $1E;
dc.b $1F;
dc.b $20;
dc.b $21;
dc.b $22;
dc.b $23;
dc.b $24;
dc.b $25;

InitTimer:  mov   #initTim,TSC ;Timer - Cleared + Stopped.
            mov   #InitTMODH,TMODH ;Set max. brightness
            mov   #InitTMODL,TMODL ;after we start the timer.
            bra   Skip

;Subroutine for Thyristor gate control

Gate:       lda   #GateVal    ;Gate pulse duration
            loop:       bset  PTA1,PTA
                        nop
                        bclr  PTA1,PTA
                        dbnza loop
                        jmp   Skip

;Subroutine for Timer Overflow

TOverflow:  nop
            nop
            brclr TOF,TSC,TOverflow ;Wait for Timer Overflow
            lda   TSC
            and   #TSCClr
            sta   TSC         ;Clear TOF bit
            mov   #initTim,TSC ;STOP and RESET Counter
            bra   Skip

;Subroutine for Dimmer

IncTimer:   incx
            cpx   #IncTcomp
            bhi   Escape
            lda   MSBTimer,x
            sta   TMODH
            lda   LSBTimer,x
            sta   TMODL
            bra   Skip

Escape:     ldx   #IncTcomp
Software Description

lda MSBTimer,x
sta TMODH
lda LSBTimer,x
sta TMODL
bra Skip

DecTimer: decx
cpx #DecTcomp
blo EscapeDec
lda MSBTimer,x
sta TMODH
lda LSBTimer,x
sta TMODL
bra Skip

EscapeDec: ldx #DecTcomp
lda MSBTimer,x
sta TMODH
lda LSBTimer,x
sta TMODL
bra Skip

Delay: lda #Delval
Xloop: brn *
brn *
dbnza Xloop
Skip: rts

;*******************************************************************************************
;* Main Init
;* This is the point where code starts executing after a RESET.
;*******************************************************************************************

main:
mov #initCfg1,CONFIG1 ;Set config1 register
    ;(LVI and COP disabled)
mov #initCfg2,CONFIG2 ;set MCU to internal oscillator, IRQ enabled
mov #initDDRA, DDRA ;PTA0 -> Zero Crossing detection
bset DDRA1, DDRA ;PTA1 -> Pulses on Thyristor gate
    ;PTA2 as IRQb -> Turns on dimmer
    ;PTA3 as RSTb -> Turns on 1-minute timer
    ;PTA4 -> Dec. lamp brightness
    ;PTA5 -> Inc. lamp brightness
bset PTAPUE4,PTAPUE
bset PTAPUE5,PTAPUE
clr Counter1
clr Counter2
clrh
ldx #Xval
jsr InitTimer ;Goes config Timer
cli ;Allow interrupts to happen
Waitpta0:
  nop
  brclr PTA0,PTA,Waitpta0 ;Wait for a edge on PTA0 (Zero crossing)
  brclr PTA5,PTA,IncT ;Inc timer if PTA5 is clear
  brclr PTA4,PTA,DecT ;Dec timer if PTA4 is clear

Next:
  mov   #StartTim,TSC   ;Start the timer
  jsr   TOverflow       ;Go to Timer Overflow subroutine
  jsr   Gate            ;Go to Gate subroutine

Waitpta:
  nop
  brset PTA0,PTA,Waitpta ;Wait for a edge on PTA0 (Zero crossing)

Next1:
  mov   #StartTim,TSC ;Start the timer
  jsr   TOverflow     ;Go to Timer Overflow subroutine
  jsr   Gate          ;Go to Gate subroutine
  bra   Waitpta0

IncT:
  jsr   Delay
  cpx   #IncTcomp
  beq   Next
  jsr   IncTimer
  bra   Next

DecT:
  jsr   Delay
  cpx   #DecTcomp
  beq   Next
  jsr   DecTimer
  bra   Next

;**** Interrupt Vectors ************
  org    $FFFE
  dcw    main

END
 Republicans pushed to add a government-owned drug stockpile, arguing that it's needed to ensure drug supply in emergencies or natural disasters. The stockpile could also help combat global pandemics and address other supply chain issues. The plan would help mitigate supply shortages that have occurred in recent years. Republicans want to ensure that critical medications are available when they are needed most. They argue that adding a government-owned stockpile is the only way to guarantee that drugs will be available to patients when they need them. The stockpile would also be used to address global pandemics and prevent shortages of critical medications. The plan would provide a safety net for patients who rely on these drugs. The Republican plan is already gaining traction among representatives and senators. Leaders want to ensure that patients have access to critical medications during emergencies and pandemics. The plan is expected to pass in the near future.
Software Description

IMPLEMENTING A LAMP DIMMER WITH AN HC908Q FAMILY MCU, REV. 0

Freescale Semiconductor

;*******************************************************************************************
;* SUBROUTINES
;* This part includes subroutines
;*******************************************************************************************

org FlashStart

InitTimer:  mov  #initTim,TSC ;Timer - Cleared + Stopped.
              mov  #InitTMODH,TMODH ;Set max. brightness
              mov  #InitTMODL,TMODL ;after we start the timer.
              bra  Skip

;Subroutine for Thyristor gate control
Gate:       lda  #GateVal    ;Gate pulse duration

loop:       bset  PTA1,PTA
              nop
              bclr  PTA1,PTA
              dbnz  loop

jmp   Skip

;Subroutine for Timer Overflow
TOverflow:  nop
              nop
              brclr TOF,TSC,TOverflow ;Wait for Timer Overflow

              lda  TSC
              and  #TSCClr
              sta  TSC         ;Clear TOF bit

              mov  #initTim,TSC ;STOP and RESET Counter
              bra  Skip

Skip:       rts

;*******************************************************************************************
;* Main Init
;* This is the point where code starts executing after a RESET.
;*******************************************************************************************

main:

              mov  #initCfg1,CONFIG1 ;Set config1 register
                     ;(LVI and COP disabled)

              mov  #initCfg2,CONFIG2 ;set MCU to internal oscillator, IRQ enabled

              mov  #InitDDRA,DDRA ;PTA0 -> Zero Crossing detection
              bset  DDRA1,DDRA  ;PTA1 -> Pulses on Thyristor gate
                     ;PTA3 as RSTb -> Turns on 1-minute timer

clr  Counter1
clr  Counter2
clrh
clrx

Implementing a Lamp Dimmer with an HC908Q Family MCU, Rev. 0
Software Description

```
jsr InitTimer ; Goes config Timer
cli ; Allow interrupts to happen

ZeroDetect: nop
brclr PTA0, PTA, ZeroDetect ; Wait for a edge on PTA0 (Zero crossing)
mov #StartTim, TSC ; Start the timer
jsr TOverflow ; Go to Timer Overflow subroutine
jsr Gate ; Go to Gate subroutine

ZeroDetect: nop
brset PTA0, PTA, ZeroDetect ; Wait for a edge on PTA0 (Zero crossing)
mov #StartTim, TSC ; Start the timer
jsr TOverflow ; Go to Timer Overflow subroutine
jsr Gate ; Go to Gate subroutine

inc Counter1 ; Increment 1st byte Counter for charge time OVF period
lda #CntLcmp
cbeq Counter1, Count1
bra ZeroDetect

Count1: inc Counter2 ; Increment 2nd byte Counter for charge time OVF period
lda #CntHcmp
cbeq Counter2, Out
bra ZeroDetect

Out: clr Counter1
clr Counter2
stop

;**** Interrupt Vectors ************
org $FFFE
dcw main

END
```
Software Description

/* Title: dimmer.equ                               Copyright (c) Freescale 2004 */
/* Author: Marcus Espindola - Freescale SPS/BSTC */
/* Description: Constants and variables definitions for MC68HC908QY4 and MC68HC908QT4. */
/* Documentation: HC908QY4 Data Sheet (MC68HC908QY4/D) for register and bit explanations */
/* Include Files: */
/* Assembler: P&E Microcomputer Systems - CASM for HC08 */
/* Metrowerks CodeWarrior Compiler for HC08 V-5.0.17 */
/* Revision History: */
/* Rev # Date Who Comments */
/* ----- ----------- --------- -------------------------------------------- */
/* 0.1 09-Feb-04 Espindola Initial data entry */
/* Freescale reserves the right to make changes without further notice to any product */
/* herein to improve reliability, function, or design. Freescale does not assume any */
/* liability arising out of the application or use of any product, circuit, or software */
/* described herein; neither does it convey any license under its patent rights nor the */
/* rights of others. Freescale products are not designed, intended, or authorized for */
/* use as components in systems intended for surgical implant into the body, or other */
/* applications intended to support life, or for any other application in which the */
/* failure of the Freescale product could create a situation where personal injury or */
/* death may occur. Should Buyer purchase or use Freescale products for any such */
/* intended or unauthorized application, Buyer shall indemnify and hold Freescale and */
/* its officers, employees, subsidiaries, affiliates, and distributors harmless against */
/* all claims, costs, damages, and expenses, and reasonable attorney fees arising out */
/* of, directly or indirectly, any claim of personal injury or death associated with */
/* such unintended or unauthorized use, even if such claim alleges that Freescale was */
/* negligent regarding the design or manufacture of the part. */
/* Freescale is a registered trademark of Freescale Semiconductor, Inc. */
/* Constants and Variables for this file */

initCfg1:   equ   %00010011   ;Config1 Register value
            ||||||||   CONFIG1 is a write once register
            +---COPD - 1 disable COP Watchdog
            +----STOP - 1 enable STOP instruction
            +-----SSREC - 0 4096 cycle STOP recovery
            +------LVI5OR3 - 0 set LVI for 3V system
            +--------LV5IPWRD - 1 disable power to LVI system
            +---------LVIRSTD - 0 enable reset on LVI trip
            +----------LV1STOP - 0 disable LVI in STOP mode
            +------------COPRS - 0 long COP timeout

initCfg2:   equ   %01000001   ;Config2 Register value
            ||||||||   CONFIG2 is a write once register
            +---RSTEN - 1 Reset function active in pin
            +---R - 0 Reserved bit
            +---R - 0 Reserved bit
            +---OSCOP0 - 0 Set oscillator option as internal
            +---OSCOP1 - 0 Set oscillator option as internal
            +---R - 0 Reserved bit

Implementing a Lamp Dimmer with an HC908Q Family MCU, Rev. 0

Freescale Semiconductor
Notes

; +--------IRQEN - 1 enable IRQ function
; +--------IRQPUD - 0 Internal pullup connect IRQ and VDD

initTim:  equ %00110001   ;Timer Status and control Reg. value
; |||||+++PS0 - 1 Prescaler select bit
; |||||+++PS1 - 0 Prescaler select bit
; |||||+++PS2 - 0 Tim clock source int. bus
; ||+++++-0 - 0
; ||++++-TRST - 1 TIM reset bit
; ++-------TSTOP - 1 TIM counter stopped
; ++-------TOIE - 0 disable TIM overflow interrupts
; ++-------TOF - 0 TIM overflow flag bit

StartTim:  equ %00000001   ;Timer Status and control Reg. value
; |||||||| TIM Status and Control Register
; ||||||++PS0 - 1 Prescaler select bit
; ||||||++PS1 - 0 Prescaler select bit
; |||||++PS2 - 0 Tim clock source int. bus
; ||||++++-0 - 0
; |||++++-TRST - 0 TIM reset bit
; |||++++-TSTOP - 0 TIM counter started
; ++-------TOIE - 0 disable TIM overflow interrupts
; ++-------TOF - 0 TIM overflow flag bit

InitDDRA:  equ %00000010   ;PTA0 -> Zero Crossing detection
; PTA1 -> Pulses on Thyristor gate
; PTA2 -> Increment Dimmer
; PTA4 -> Decrement Dimmer
; PTA5 -> Turns on 1-minute timer.

InitIRQ:  equ $00   ;IRQ configuration
InitTMODH: equ $00   ;Set max. brightness
InitTMODL: equ $FF   ;after we start the timer.

GateVal:  equ $50   ;Gate pulse duration
TSCClr:  equ $7F   ;Value to clear TOF bit on TSC register

Counter1:  rmb 1
Counter2:  rmb 1
IncTcomp:  equ $1F
DecTcomp:  equ $01
CntLcmp:  equ $00
CntHcmp:  equ $0E
Delval:  equ $FF
Xval:  equ $01

Implementing a Lamp Dimmer with an HC908Q Family MCU, Rev. 0

Freescale Semiconductor
This page is intentionally blank.
How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale
Semiconductor products, please visit
http://www.freescale.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts.

Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized use, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.