AN1762

Automatic Contrast Control of LCD Displays Using the MC68HC708LN56 Microcontroller

By Ed Stellini
Transportation Systems Group Design Engineering
Austin, Texas

Introduction

This application note describes how to implement automatic contrast control of an LCD (liquid crystal display) using the MC68HC08LN56 microcontroller.

In applications where the power supply voltage can vary, such as in battery-powered systems, maintaining a constant contrast on the LCD display is desirable. This can be achieved in software on the MC68HC08LN56 by using the A/D (analog-to-digital) converter in conjunction with the LCD controller.

This application note discusses how factors such as the amount of multiplexing, type of bias, and voltage levels can affect LCD contrast. This is followed by a description of a system in which the A/D converter on board the MC68HC708LN56 samples the system power supply and the contrast control for the LCD is updated based on the result.

Also included here is the source code for implementing this system.

Contrast Control

The contrast of characters which appear on an LCD display is controlled by the average voltage difference across the segments or pixels in the character. In general, a larger applied voltage causes pixels to appear darker, or ON, and smaller voltages cause pixels to appear lighter, or OFF.

Due to the nature of the liquid crystal material, DC voltages applied across them will cause permanent damage. As a result, the relative contrast of an LCD is characterized by the magnitude of the RMS voltage across it. In order for a pixel to be OFF, the RMS voltage across it must be below the LCD threshold voltage, V_{TH}. For it to be ON, the RMS voltage must be above the LCD saturation voltage, V_{SAT}.

A typical contrast versus voltage characteristic curve is shown in **Figure 1**.

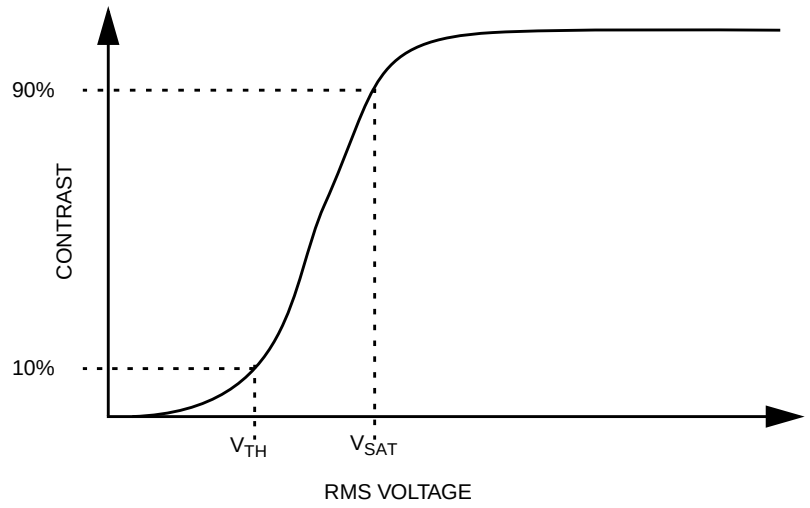

Figure 1. Typical LCD Contrast Characteristic

When selecting an LCD display for a particular application, the ON and OFF RMS voltages should be used to ensure that the proper contrast can be achieved. The ON RMS voltage of the LCD controller should be greater than V_{SAT} of the LCD glass. The OFF RMS voltage should be less than V_{TH} of the LCD glass.

The MC68HC708LN56 Microcontroller

The MC68HC708LN56 MCU has an onboard LCD controller/driver module which is capable of directly driving a dot matrix display with 32 backplanes and up to 40 frontplanes. Typically, this is a display composed of four rows of eight characters or two rows of 16 characters.

All necessary voltage levels for the LCD waveforms are generated by an onboard charge pump. Six voltage levels are produced for what is called 1/7 bias. The levels are adjustable and scale as a fraction of the top LCD voltage, V_{LL7}, as described later. Integrating the LCD controller/driver with the microcontroller reduces system cost for the user.

The LCD waveforms have 1/32 duty, which means they are multiplexed in 32 time intervals.

Figure 2 shows the waveforms for a backplane and a frontplane with alternating ON and OFF data. The voltage across a pixel is the difference of the backplane and frontplane waveform, shown as the waveform BP0–FPX.

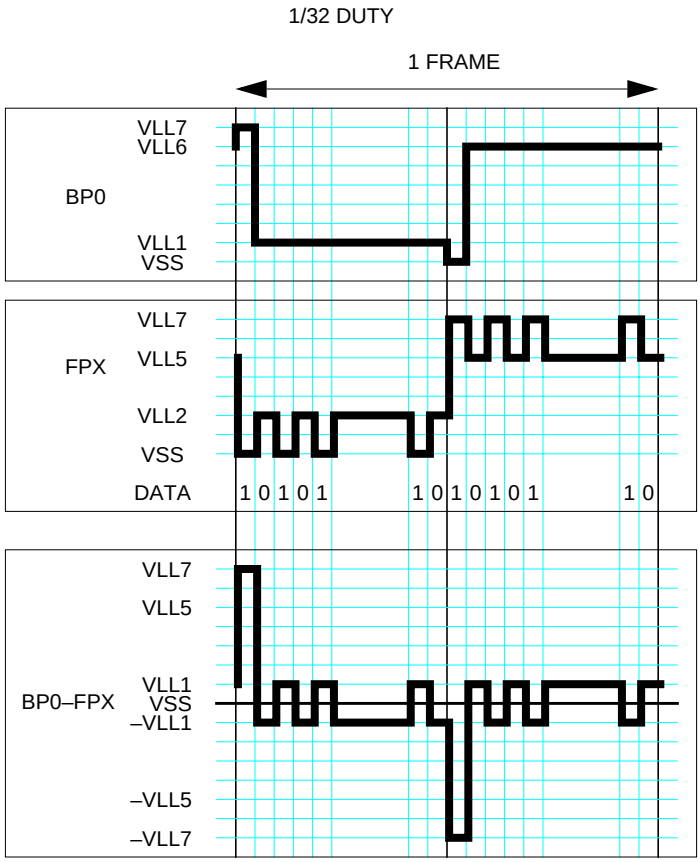

Figure 2. LCD Waveforms Example for Data of 01...10101

The RMS voltage across a pixel is calculated using this equation.

$$V_{RMS} = \sqrt{\frac{1}{T} \cdot \left(f^2\right)(t)dt}$$

Since FPX has ON data during the time period when BP0 is active, the waveform BP0–FPX is an ON waveform. The voltage during the active time is $\pm V_{LL7}$ and only $\pm V_{LL1}$ the remainder of the time.

The RMS voltage for this waveform is calculated with this equation:

$$V_{RMSON} = \sqrt{\frac{1}{64} \cdot \left(\left(\frac{7}{7} \cdot VLL7\right)^2 \cdot 2 + \left(\frac{1}{7}\ VLL7\right)^2 \cdot 62\right)} = \frac{\sqrt{10}}{14} \cdot VLL7$$

Figure 3 shows the same frontplane waveform along with backplane 1 (BP1) and their difference waveform BP1–FPX. In the period when BP1 is active, FPX has OFF data and, therefore, the waveform BP1–FPX is an OFF waveform. During its active time, the voltage swings between $\pm V_{LL5}$ and between $\pm V_{LL1}$ when inactive.

This gives an RMS voltage of:

$$V_{RMSON} = \sqrt{\frac{1}{64} \cdot \left(\left(\frac{5}{7} \cdot VLL7\right)^2 \cdot 2 + \left(\frac{1}{7} \cdot VLL7\right)^2 \cdot 62\right)} = \frac{\sqrt{5}}{14} \cdot VLL7$$

The actual values of the RMS ON and OFF voltages depend on the top LCD voltage, V_{LL7}. A typical value of V_{LL7} is 7.0 volts in which case V_{RMSON} =1.58 volts and V_{RMSOFF} = 1.32 volts.

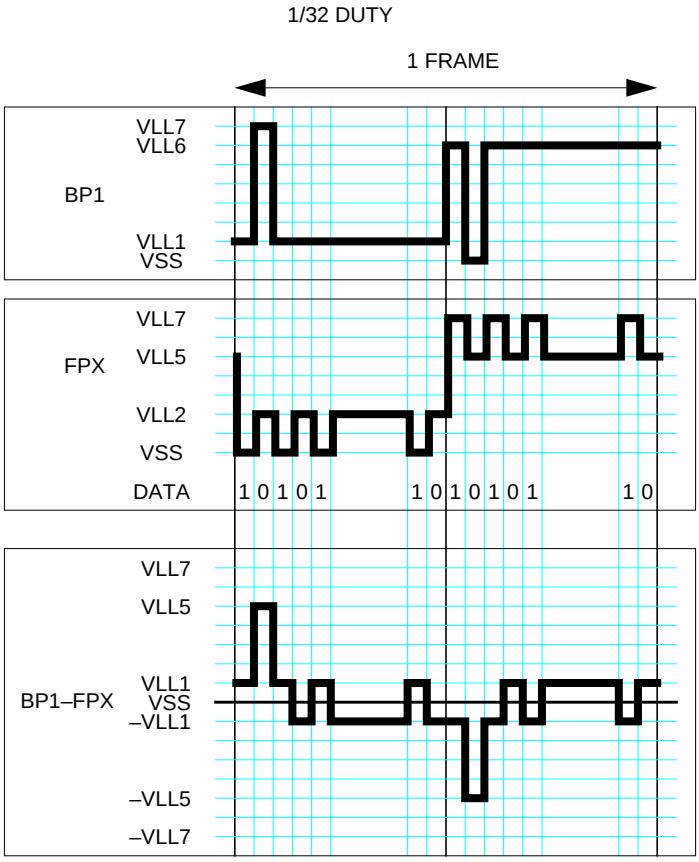

Figure 3. LCD Waveforms Example for Data of 01...10101

The MC68HC708LN56 allows the user the ability to set the value of V_{LL7} and, therefore, to move the operating points of V_{RMSON} and V_{RMSOFF}.

For a desired value of V_{LL7}, the contrast control register should be set to the value resulting from this equation:

$$LCDCCR = RND\left(\frac{VLL7}{VDD} \cdot (47.143 \cdot SUPV + 94.286) - 160\right)$$

SUPV is the value of a bit set depending on the range in which V_{DD} is operating, SUPV = 1 for 3-volt operation, and SUPV = 0 for 5-volt operation. This LCDCCR register value can be written at any time during operation, and the LCD waveforms will scale in proportion to the value of V_{LL7}.

Sample Application

In a typical battery application, the power supply gradually decays over time. Since the top LCD voltage is dependent on V_{DD}, to maintain a constant contrast on the LCD display, the contrast register must be updated with a new value that will result in the same top LCD voltage at all times. This can be done by using the A/D converter to periodically measure V_{DD} and then to determine the correct setting for the contrast control register. By doing this continuously in software, automatic contrast control is achieved.

In this sample application, the contrast on the LCD display is maintained at a constant value for a battery-powered supply ranging from 4.2 volts down to 3.2 volts. Manual contrast control also is implemented by up and down contrast buttons which allows the user to manually increment or decrement the contrast setting.

A schematic diagram of the circuit for this application is shown in **Figure 4**. PB0 is used as the input channel for the A/D. A MAXIM 6125 voltage reference chip generates V_{REFH} for the A/D. This reference of 2.5-volt must be greater than the maximum A/D input voltage. A simple resistor divider is used to divide V_{DD} in half, which allows a maximum A/D input voltage of 2.1 volts (max V_{DD} = 4.2 volts). The LCD display used in this circuit is a custom display with 32 backplanes and 40 frontplanes. Displays of this size can be ordered readily from many manufacturers and can be optimized to the user's specifications.

The software uses the timebase module to create a periodic interrupt every set interval of time. At this time, the A/D samples V_{DD}. An offset is generated to a lookup table of contrast control values from the A/D value. The contrast control register is written with this value which will keep V_{LL7} at the original value.

The two buttons used for manual contrast control are connected to PA1/KBD1 and PA0/KBD0 and when pushed, generate a keyboard interrupt. These keyboard interrupts will either increase or decrease the conversion factor for the lookup table by a constant amount. In this application, the manual adjustments were designed to change V_{LL7} by ~100-mv steps. All automatic contrast control adjustments are then relative to the latest manual adjustment.

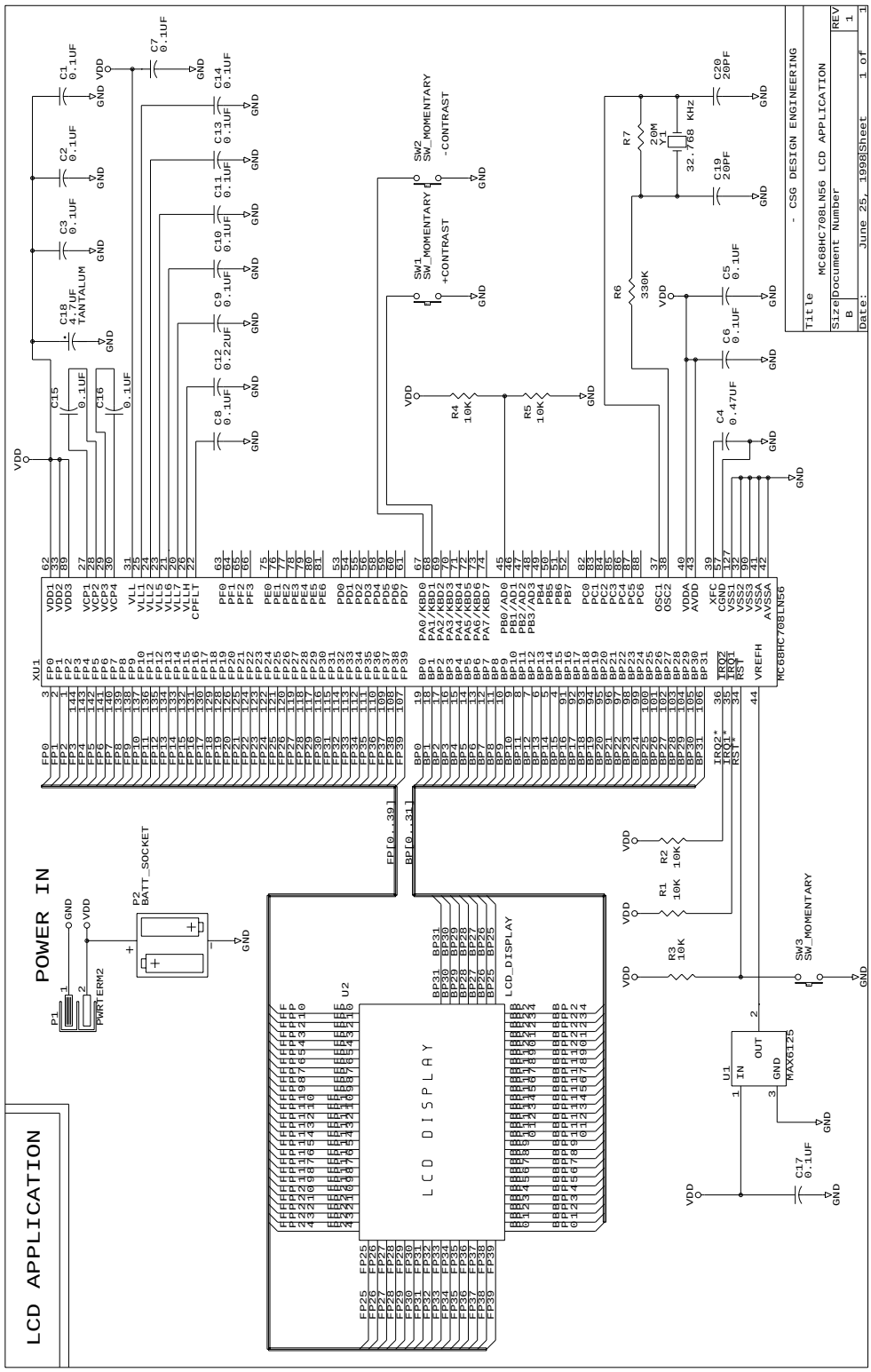

Figure 4. Schematic Diagram of Sample LCD Application

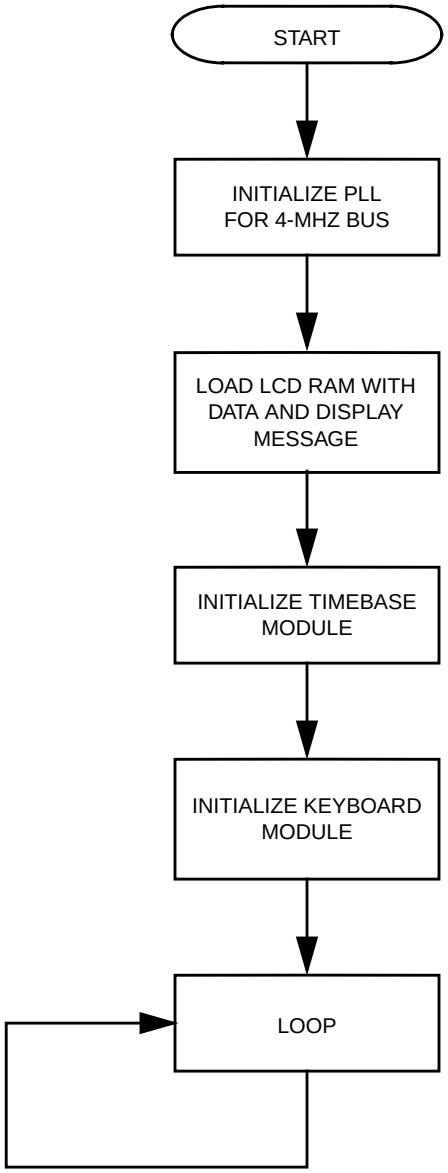

Figure 5. Main Program Flow

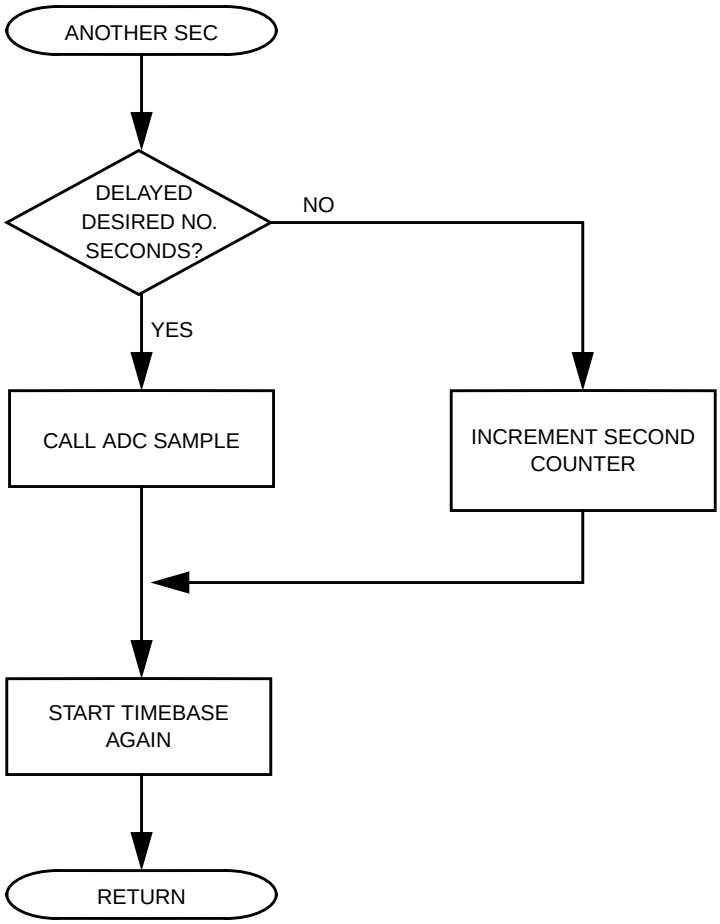

Figure 6. Timebase Interrupt Routine

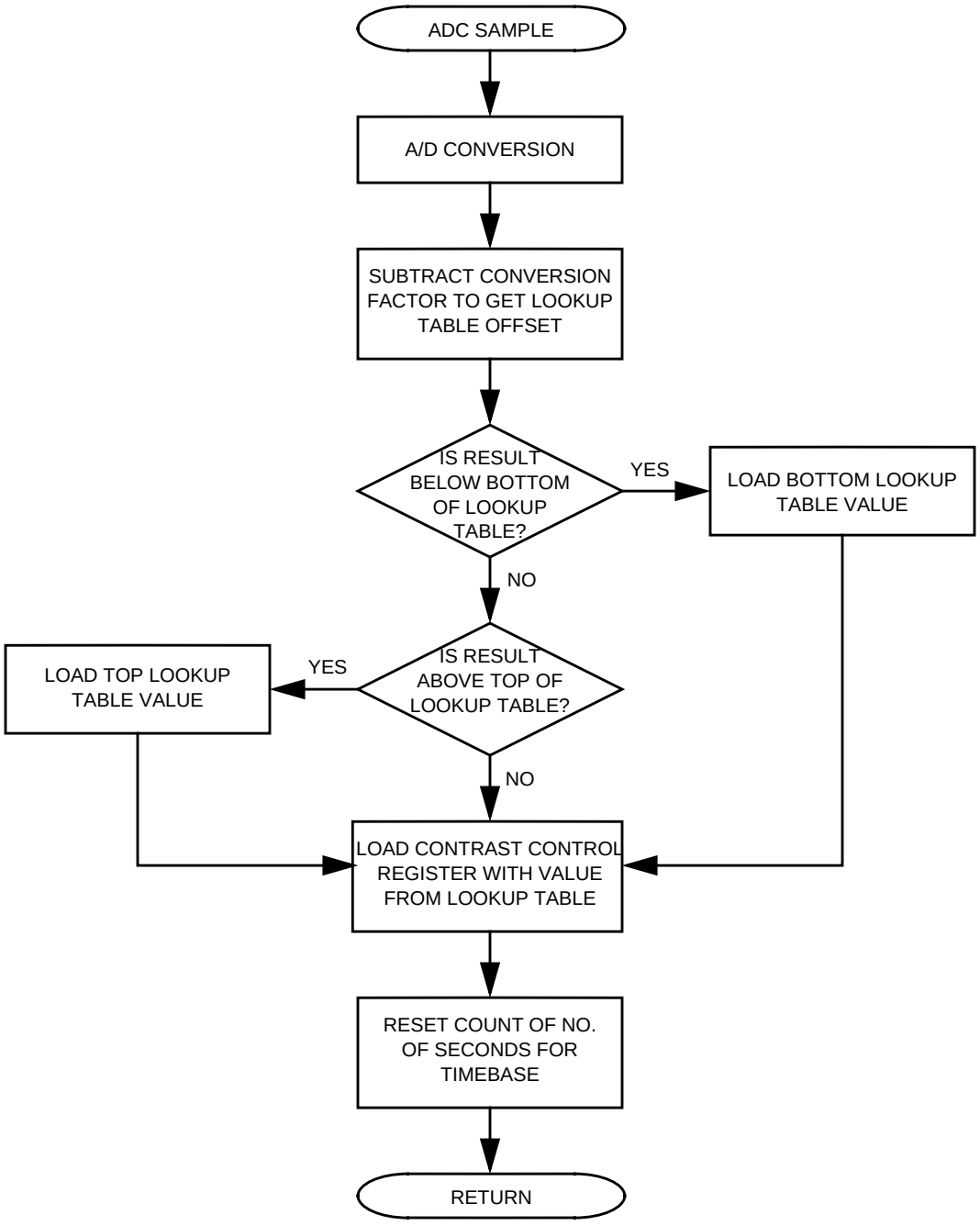

Figure 7. ADC Sample Subroutine

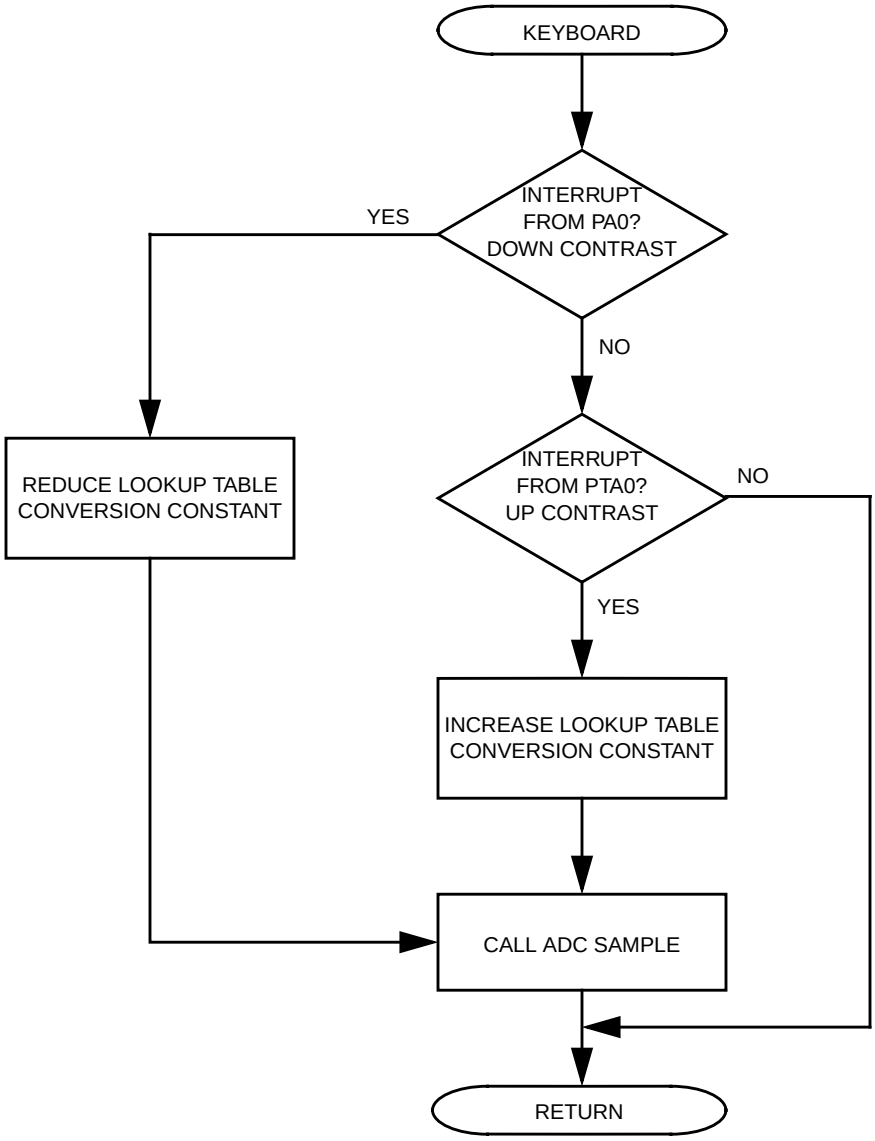

Figure 8. Keyboard Interrupt Routine

Code Listings

```
*****************************************************************************
* ADCLCD.ASM
*****************************************************************************
* Ed Stellini, 06/06/98
* CSG Applications Engineering
*
*
* This software demonstrates automatic contrast control of an
* LCD display using the MC68HC708LN56 microcontroller.
*
*****************************************************************************

*****************************************************************************
* Memory Equates
*****************************************************************************
RAMSPACE          EQU     $0050
ROMSPACE          EQU     $2000
LCDRAM1           EQU     $0E00
LCDRAM2           EQU     $0E80
LCDRAM3           EQU     $0F00
LCDRAM4           EQU     $0F80

*****************************************************************************
* Register Equates
*****************************************************************************
* PTD Registers
PTA               EQU     $0000
DDRA              EQU     $0004
* LCD Registers
LCDFL0            EQU     $0033
LCDFL1            EQU     $0034
LCDFL2            EQU     $0035
LCDFL3            EQU     $0036
LCDFL4            EQU     $0037
LCDCR             EQU     $0038
LCDCCR            EQU     $0039
LCDDIV            EQU     $003A
LCDFR             EQU     $003B
* ADC Registers
ADSCR             EQU     $0040
ADR               EQU     $0041
ADCLKR            EQU     $0042
* PLL Registers
PCTL              EQU     $004A
PBWC              EQU     $004B
PMSH              EQU     $004C
PMSL              EQU     $004D
```

```
PVRS              EQU     $004E
PRDS              EQU     $004F
* Time Base Registers
TBCR              EQU     $003F
* Keyboard Registers
KBSCR             EQU     $001A
KBIER             EQU     $001B
* MOR Register
MOR               EQU     $001F
* Interrupt Vectors
ADC               EQU     $FFDA
KEY               EQU     $FFDC
TBM               EQU     $FFD8
RESET             EQU     $FFFE

*****************************************************************************
* Constant Equates
*****************************************************************************
ADJUST            EQU     !153                ;initial lookup conv factor
NUMSECS           EQU     !05                 ;number of sec between A/D samples

*****************************************************************************
* RAM Variables
*****************************************************************************
                  ORG     RAMSPACE
TBMCNT            RMB     1
ADCTOCCR          RMB     1

*****************************************************************************
* Start of Program Code
*****************************************************************************
                  ORG   ROMSPACE
                  MOV   #$01,MOR              ;Disable COP

BEGIN             CLRX
                  CLRH
                  MOV  #ADJUST,ADCTOCCR       ;store conversion factor
                                              ;in ram variable

*****************************************************************************
* Initialize PLL
* Program the PLL for 4 MHz Bus from 32.768KHz crystal
*****************************************************************************
                  MOV   #$01,PCTL
                  MOV   #$80,PBWC
                  MOV   #$01,PMSH
                  MOV   #$E8,PMSL
                  MOV   #$D0,PVRS
                  MOV   #$01,PRDS
                  MOV   #$21,PCTL
                  BRCLR 6,PBWC,*
                  MOV   #$31,PCTL
```

```
*****************************************************************************
* Initialize LCD by loading each bank of LCD RAM with part of
* the message.
*****************************************************************************
LOOP1             LDA   LCDBANK1,X
                  STA   LCDRAM1,X
                  INCX
                  CPX   #28
                  BNE   LOOP1
                  CLRX

LOOP2             LDA   LCDBANK2,X
                  STA   LCDRAM2,X
                  INCX
                  CPX   #28
                  BNE   LOOP2
                  CLRX

LOOP3             LDA   LCDBANK3,X
                  STA   LCDRAM3,X
                  INCX
                  CPX   #28
                  BNE   LOOP3
                  CLRX

LOOP4             LDA   LCDBANK4,X
                  STA   LCDRAM4,X
                  INCX
                  CPX   #28
                  BNE   LOOP4

                  MOV   #$01,LCDDIV           ;to get LCD clk to 32KHz
                  MOV   #$04,LCDFR            ;for frame rate
                  MOV   #$17,LCDCCR           ;starting CCR value
                  MOV   #$80,LCDCR            ;turn on LCD

*****************************************************************************
* Initialize Time Base Module
*****************************************************************************
                  CLI                         ;clear interrupts
                  MOV   #$4A,TBCR             ;cgmxclk div 32768
                                              ;start TBM
                  MOV   #!01,TBMCNT           ;initialize TBM count

*****************************************************************************
* Initialize Keyboard Module
*****************************************************************************
                  MOV  #$02,KBSCR             ;mask interrupts
                  MOV  #$03,KBIER             ;enable interrupts bits 1 and 0
                  MOV  #$06,KBSCR             ;acknowledge false interrupts
                  MOV  #$00,KBSCR             ;unmask interrupts, edge sens
```

```
*****************************************************************************
* Main Loop
*****************************************************************************
                    BRA    *                        ;wait here for timer interrupt

*****************************************************************************
* Time Base Interrupt Routine
*****************************************************************************
ANOTHERSEC          LDA   TBMCNT                    ;load TBM count
                    CMP   #NUMSECS                  ;have we delayed enough seconds?
                    BNE   INCREMENT                 ;don't take ADC sample yet
                    JSR   ADCSAMPLE                 ;go take ADC sample
                    BRA   RETURN
INCREMENT           INC   TBMCNT                    ;increment second counter
RETURN              MOV   #$4A,TBCR                 ;ack $ start time base again
                    RTI                             ;delay some more

*****************************************************************************
* Take ADC Sample and Update LCD CCR
*****************************************************************************
ADCSAMPLE           MOV    #$50,ADCLKR              ;use bus clk and /4 (1MHz)
                    MOV    #$01,ADSCR               ;PTB1 input to ADC
WAIT                BRCLR 7,ADSCR,WAIT              ;stay here til conversion done
CONVDONE            LDA    ADR                      ;load accum with ADC value
                    SUB    ADCTOCCR                 ;convert to offset from table
                    BMI    BOTTOM                   ;went below zero
                    CMP   #!56                      ;check if over top
                    BGT   TOP                       ;went above top
                    BRA   CORRECTED                 ;still within valid range
BOTTOM              LDA   #!00                      ;don't let go past bottom
                    BRA   CORRECTED                 ;return
TOP                 LDA   #!56                      ;don't let go past top
CORRECTED           TAX                             ;put in X to use as index
                    LDA    Table,X                  ;get CCR value from lookup table
                    STA    LCDCCR                   ;put table value in CCR
                    MOV   #!01,TBMCNT               ;start count over
                    RTS                             ;return

*****************************************************************************
* Keyboard Interrupt Routine
*****************************************************************************
KEYBOARD            MOV   #$00,DDRA                 ;make PTA inputs to read
                    BRCLR 0,PTA,DOWN                ;down contrast button pushed
                    BRCLR 1,PTA,UP                  ;up contrast button pushed
DOWN                LDA   ADCTOCCR                  ;load conversion constant
                    SUB   #!03                      ;subtract 3 from constant
                    BRA   KEYDONE                   ;return
UP                  LDA   ADCTOCCR                  ;load conversion constant
                    ADD   #!03                      ;add 3 to constant
KEYDONE             STA   ADCTOCCR                  ;write value back to variable
```

```
                JSR  ADCSAMPLE              ;go update LCDCCR value
                RTI                         ;return

*****************************************************************************
* Lookup table of contrast control register values.
*****************************************************************************
Table           FCB   $3C
                FCB   $3B
                FCB   $39
                FCB   $38
                FCB   $36
                FCB   $35
                FCB   $34
                FCB   $32
                FCB   $31
                FCB   $30
                FCB   $2E
                FCB   $2D
                FCB   $2C
                FCB   $2A
                FCB   $29
                FCB   $28
                FCB   $27
                FCB   $26
                FCB   $24
                FCB   $23
                FCB   $22
                FCB   $21
                FCB   $20
                FCB   $1F
                FCB   $1E
                FCB   $1D
                FCB   $1C
                FCB   $1A
                FCB   $19
                FCB   $18
                FCB   $17
                FCB   $16
                FCB   $15
                FCB   $14
                FCB   $13
                FCB   $12
                FCB   $11
                FCB   $10
                FCB   $10
                FCB   $0F
                FCB   $0E
                FCB   $0D
                FCB   $0C
                FCB   $0B
                FCB   $0A
                FCB   $39
                FCB   $08
```

```
                FCB   $08
                FCB   $07
                FCB   $06
                FCB   $05
                FCB   $04
                FCB   $03
                FCB   $03
                FCB   $02
                FCB   $01
                FCB   $00
*******************************************************************************

*******************************************************************************
* Data for writing the LCD display to say:
*           Freescale
*           PRESENTS
*           THE HC08
*           CHINOOK
*******************************************************************************
* THE HC08  (backwards)
LCDBANK1        FCB   $36,$49,$49,$49,$36,$00,$3e,$41,$41,$3e
                FCB   $00,$41,$41,$41,$41,$7f,$00,$7f,$08,$08
                FCB   $08,$7f,$00,$49,$49,$49,$49,$7f,$00,$7f
                FCB   $08,$08,$08,$7f,$00,$01,$01,$7f,$01,$01

* CHINOOK   (backwards)
LCDBANK2        FCB   $61,$32,$0c,$08,$7f,$00,$7f,$41,$41,$41
                FCB   $7f,$00,$7f,$41,$41,$41,$7f,$00,$7f,$38
                FCB   $0c,$06,$7f,$00,$00,$7f,$7f,$00,$00,$7f
                FCB   $08,$08,$08,$7f,$00,$41,$41,$41,$41,$7f

* FREESCALE  (forwards)
LCDBANK3        FCB   $7f,$06,$0c,$06,$7f,$00,$7f,$41,$41,$7f
                FCB   $00,$01,$7f,$7f,$01,$00,$7f,$41,$41,$7f
                FCB   $00,$7f,$19,$29,$4f,$00,$7f,$41,$41,$7f
                FCB   $00,$7f,$40,$40,$40,$00,$7e,$09,$09,$7e

* PRESENTS  (forward)
LCDBANK4        FCB   $7f,$09,$09,$0f,$00,$7f,$09,$39,$6f,$00
                FCB   $7f,$49,$49,$49,$00,$4f,$49,$49,$79,$00
                FCB   $7f,$49,$49,$49,$00,$7f,$0e,$38,$7f,$00
                FCB   $01,$7f,$7f,$01,$00,$4f,$49,$49,$79,$00
*******************************************************************************

                ORG   RESET                   ;
                FDB   ROMSPACE                ;Go to beginning of ROM on reset
                ORG   TBM                     ;
                FDB   ANOTHERSEC              ;time base interrupt routine
                ORG   KEY                     ;
                FDB   KEYBOARD                ;keyboard interrupt routine
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com