AN1763

Driving LCD Displays Using the MC68HC705L16 Microcontroller

By Ed Stellini
Transportation Systems Group Design Engineering
Austin, Texas

Introduction

This application note describes how to use the MC68HC705L16 microcontroller (MCU) as an LCD (liquid crystal display) controller/driver. By doing so, all LCD control and drive functions are performed by a single chip, which also provides all of the functionality of a microcontroller.

A description of the voltages and waveforms used to drive and control an LCD panel is included as well as an explanation of how the designer can use the MC68HC705L16 to interface directly to a simple LCD display. Also, the source code for controlling a multiplexed display is included at the end of this application note.

Liquid Crystal Displays

To understand the types of waveforms that drive LCD displays, it is helpful to understand a few fundamentals about LCDs.

For example, liquid crystal displays are composed of a polarizing liquid crystalline material in between two plates of glass. Typically, one plate is called the common or backplane, and the other is called a segment or frontplane. In a reflective LCD panel (one that has no back light), a voltage difference applied across the two electrodes will result in a polarization which will prevent the light from reflecting back to the observer. This will appear as a dark segment and is, therefore, considered ON. A lack of voltage difference will allow the light to reflect back and is considered OFF.

Contrast

Due to the chemical nature of the liquid crystal material, DC voltages cannot be used to drive the segments or else permanent damage can occur to the LCD. To avoid this problem, voltage levels are applied to the electrodes for a short period and then the levels are reversed to the electrodes for an equal period. This AC waveform will produce an RMS voltage across the LCD, yet it has a net DC value of 0 volts. As a result, LCD material has its contrast specified in terms of an RMS voltage. A typical voltage characteristic for a reflective LCD display is shown in **Figure 1**.

The ON voltage for a segment should be greater than the point where incident light is reduced by 90 percent. The OFF voltage should be less than the point where incident light is reduced by 10 percent. For maximum contrast, the ratio of ON to OFF voltage should be as large as possible. Examples of how to calculate RMS voltages are shown in a later section.

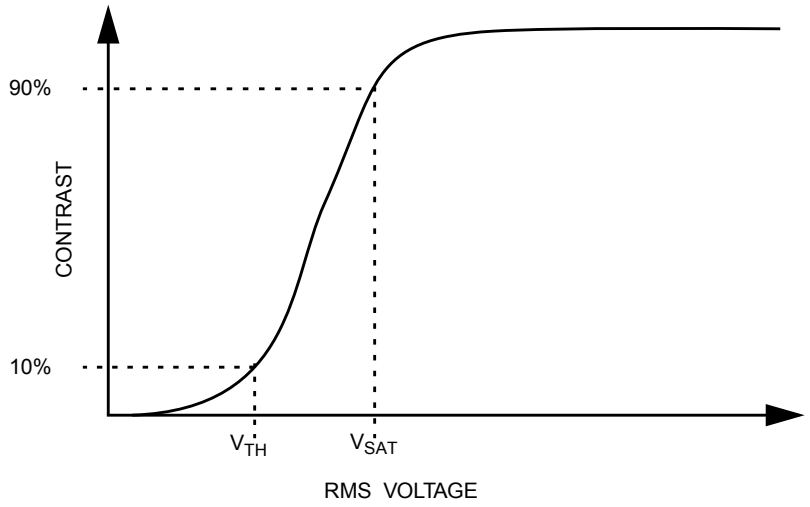

Figure 1. Typical Contrast Characteristic of LCD

Static Mode

Typically, LCD displays are made up of segments or pixels. Segment displays usually have anywhere from 8 to 16 segments for displaying each character, while dot matrix displays typically have arrays of 5 x 7 pixels for each character.

Each of these segments or pixels needs to be driven independently in order for it to be turned on or off independently. The simplest way to do this is to have a separate frontplane driver for each segment or pixel and have a single backplane driver for the entire display. This is known as direct drive or static mode.

Example waveforms of the frontplane and backplane drivers for static mode are shown in **Figure 2**. The voltage across a segment is the difference of the backplane waveform and the frontplane waveform. See **Figure 3**.

For a segment to be OFF, its frontplane waveform and backplane waveform will have the same amplitude and will be completely in phase. This causes the voltage across the segment to be 0 volts.

For a segment to be ON, its frontplane and backplane waveforms will be exactly out of phase. This will produce a difference across the segment equal to the top LCD voltage.

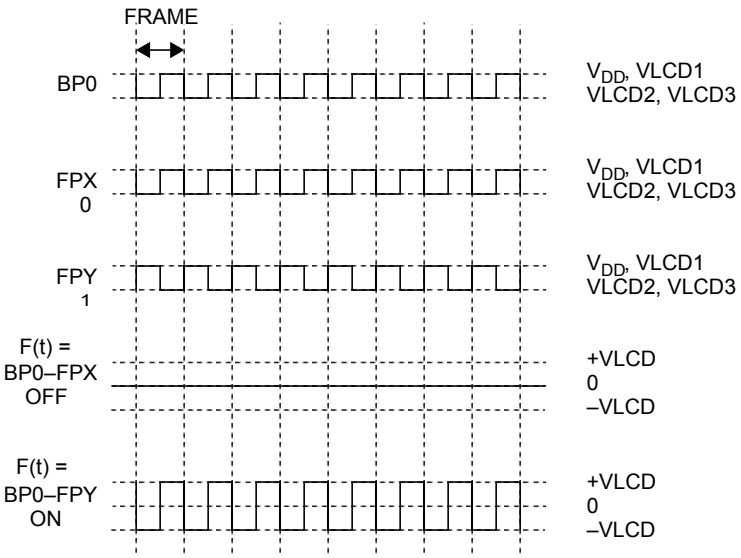

Figure 2. LCD 1/1 Duty and 1/1 Bias Timing Diagram, $V_{LCD1} = V_{DD}$, $V_{LCD2} = V_{LCD3} = V_{DD}–V_{LCD}$

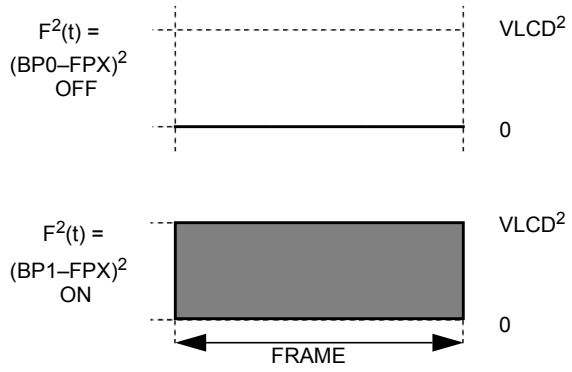

Figure 3. Waveform Components for Calculating ON and OFF RMS Voltages

Since static mode has only two voltage levels, the three LCD pins on the L16 must be connected in a specific manner as shown in **Figure 4**. Here V_{LCD1} is connected to V_{DD}, a resistor is connected between V_{LCD1} and V_{LCD2}, and V_{LCD2} and V_{LCD3} are connected directly. A variable resistor can be connected from V_{LCD3} to ground to allow manual contrast control.

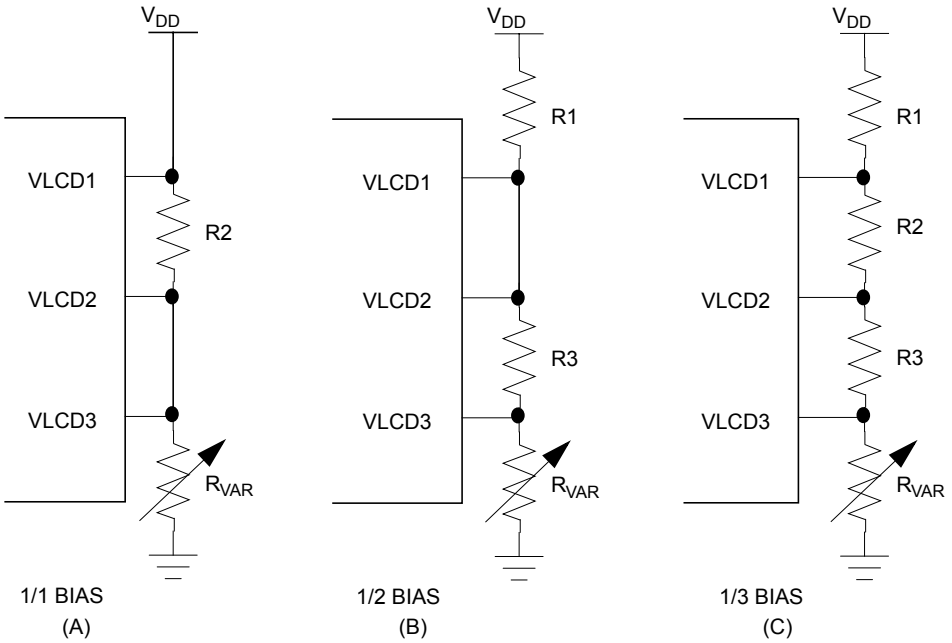

Figure 4. External Connections for 1/1, 1/2, and 1/3 Bias

Now the RMS voltages for the static waveform can be calculated from this formula:

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

The function f(t) here is the waveform, BPX–FPY. **Figure 3** shows $f^2(t)$ of the ON and OFF voltage waveforms for one frame. This serves as a graphical aid to illustrate the RMS voltages, which are simply the area under these curves. For the OFF segment, the RMS voltage is obviously 0 volts.

For the ON segment:

$$V_{RMSON} = \sqrt{\frac{1}{2} \cdot \left(V_{LCD}{}^2 + V_{LCD}{}^2\right)} = V_{LCD}$$

where $V_{LCD} = V_{DD} - V_{LCD2}, V_{LCD3}$. This V_{RMSON} voltage, typically, is well above the 90 percent ON threshold, thus producing excellent contrast.

The total number of pins needed for static mode drive is equal to the number of total segments in the display plus one. Unfortunately, as the size of the LCD display increases, the number of required LCD driver pins becomes very large.

As Table 1 shows, for more than a few 8-segment characters, the number of pins required becomes unreasonable. For small displays, though, this type of LCD drive is commonly used since it produces excellent contrast.

Table 1. LCD Driver Pins Versus Multiplexing

Display Type	Total Segments/ Pixels	Number of Driver Pins				
		Static	1/2 Duty	1/3 Duty	1/4 Duty	1/32 Duty
1 16-segment digit	16	17	10	9	8	N/A
4 7-segment digits	32	33	18	14	12	N/A
8 alphanumeric characters	120	121	62	43	34	36
32 5 x 8 pixel characters	1280	1281	642	430	324	72
General case	S	S +1	S/2 + 2	S/3 + 3	S/4 + 4	S/32 + 32

Multiplex Modes

To reduce the number of drivers required, the data for each frontplane can be multiplexed to control multiple segments by using multiple backplanes. This is done by multiplexing the driving voltages in time.

For instance:

- If each frontplane controls two segments, two backplanes are needed. This is called duplex mode.
- Triplex mode is where each frontplane driver controls three segments and three backplanes are needed.
- Similarly, quadraplex mode has each frontplane driver controlling four segments and has four backplane drivers.

Duplex Mode

The waveforms for duplex mode multiplexing are shown in **Figure 5**. The first thing to notice here is that there are now three voltage levels in each waveform. This is known as 1/2 bias. Connections to the V_{LCD} pins for this configuration should be made as shown in **Figure 4**.

Also obvious from these waveforms is that there are two time cycles in each waveform which make up a frame.

They are:

- When frontplanes connected to segments with backplane 0 are active
- When frontplanes connected to segments with backplane 1 are active

In time cycle one, frontplane X is ON, while in cycle two, it is OFF. Frontplane Y is OFF in both cycles.

Looking at the waveform for BP0–FPX, the ON data produces the maximum voltage swing, V_{LCD}, during its active time, cycle one. The waveform BP1–FPX has OFF data which produces a 0 voltage swing during its active time, cycle two.

NOTE: *Notice that when both of these segments are not active, they have the same voltage swing, $V_{LCD}/2$, even though one has OFF data when non-active and the other has ON data when non-active. This is important because the RMS voltages for ON and OFF waveforms should be independent of the data during non-active cycles. Otherwise, there would be multiple ON RMS voltages as well as OFF RMS voltages.*

Figure 6 shows the components for calculating the RMS waveforms for duplex mode ON and OFF cases. The ON and OFF RMS voltages are calculated as:

$$V_{RMSON} = \sqrt{\frac{1}{2} \cdot \left[V_{LCD}{}^2 + \left(\frac{V_{LCD}}{2} \right)^2 \right]} = 0.79 \cdot V_{LCD}$$

$$V_{RMSOFF} = \sqrt{\frac{1}{2} \cdot \left(\frac{V_{LCD}}{2} \right)^2} = 0.353 \cdot V_{LCD}$$

For V_{LCD} = 5 volts, V_{RMSON} = 3.95 volts, and V_{RMSOFF} = 1.75 volts

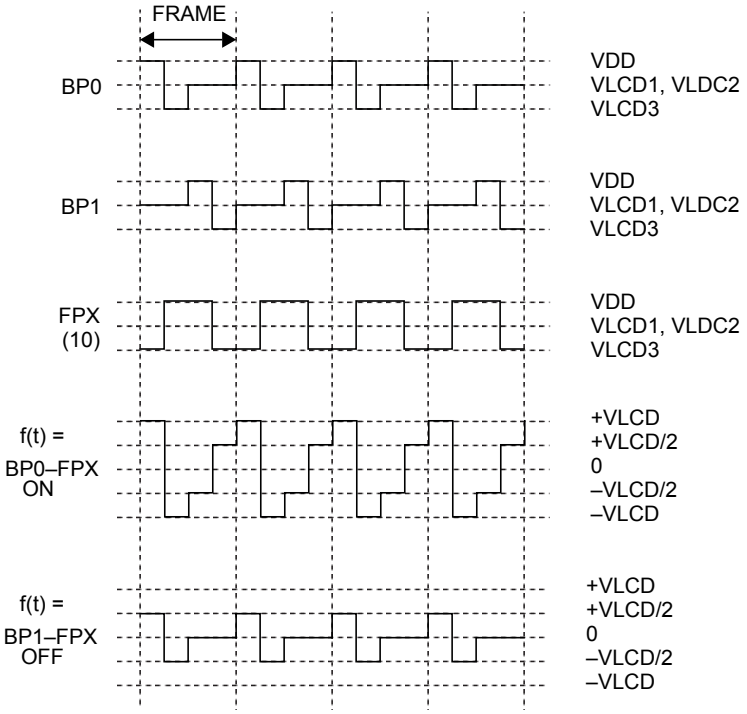

Figure 5. LCD 1/2 Duty and 1/2 Bias Timing Diagram, $V_{LCD1} = V_{LCD2} = V_{DD}-V_{LCD}/2$, $V_{LCD3} = V_{DD}-V_{LCD}$

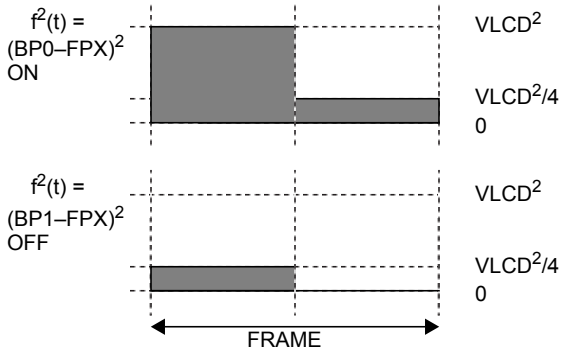

Figure 6. Waveform Components for Calculating ON and OFF RMS Voltages

Triplex Mode

Triplex multiplexing uses four voltage levels (1/3 bias) and utilizes three time cycles per frame. See **Figure 4** for V_{LCD} pin connections. Waveforms for triplex (1/3 duty) are shown in **Figure 7**. Again, notice that during the active cycle, an ON voltage swing across a segment is $\pm V_{LCD}$. The OFF voltage swing is only $\pm V_{LCD}/3$.

The components for calculating the RMS voltages are shown in **Figure 8** and are calculated as:

$$V_{RMSON} = \sqrt{\frac{1}{3} \cdot \left[V_{LCD}{}^2 + 2 \cdot \left(\frac{V_{LCD}}{3} \right)^2 \right]} = 0.638 \cdot V_{LCD}$$

$$V_{RMSOFF} = \sqrt{\frac{1}{3} \cdot 3 \cdot \left(\frac{V_{LCD}}{3} \right)^2} = 0.333 \cdot V_{LCD}$$

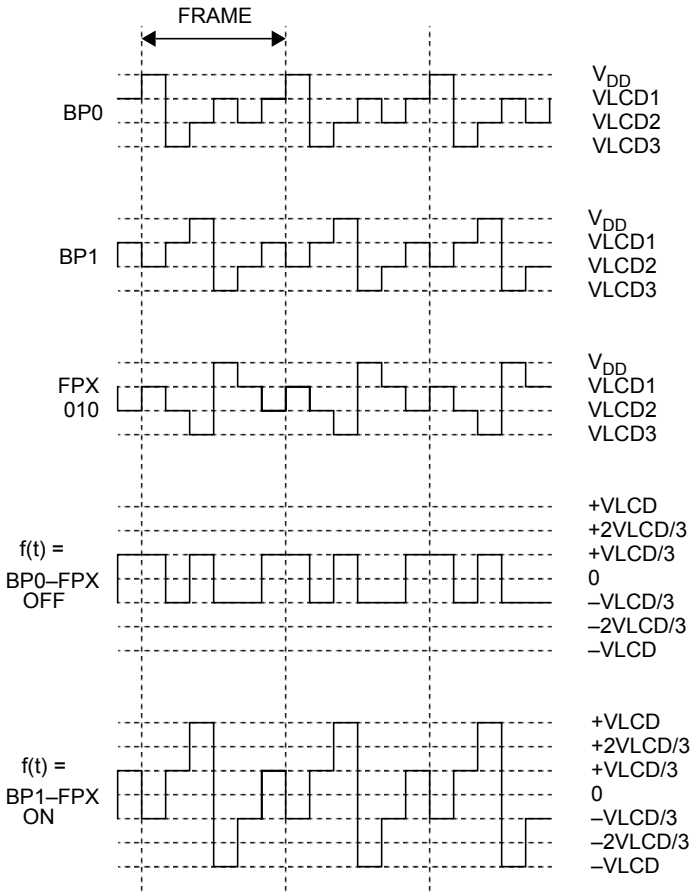

Figure 7. LCD 1/3 Duty and 1/3 Bias Timing Diagram, $V_{LCD1} = V_{DD}–V_{LCD}/3$, $V_{LCD2} = V_{DD}–2V_{LCD}/3$, $V_{LCD3} = V_{DD}–V_{LCD}$

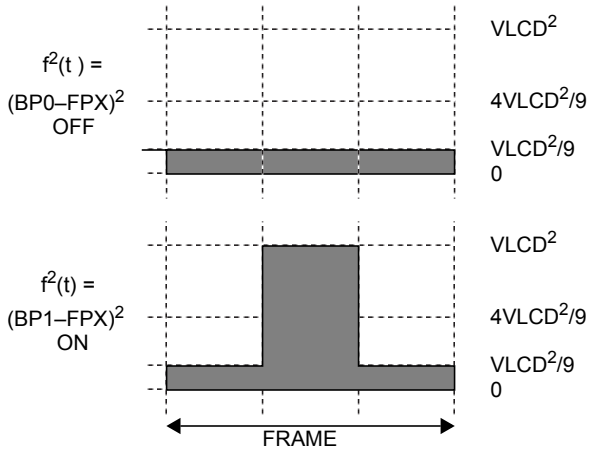

Figure 8. Waveform Components for Calculating ON and OFF RMS Voltages

Quadruplex Mode

The highest multiplexing capable by the MC68HC705L16 is quadruplex (1/4 duty). Large dot matrix displays require much larger multiplexing. For instance, a 4 x 4 0 display (4 rows of 40 pixels) needs 1/32 duty. (See *Automatic Contrast Control of LCD Displays Using the 68HC708LN56 Microcontroller*, Freescale document order number AN1762/D, for information on driving this type of display.) Waveforms for quadruplex multiplexing are shown in **Figure 9**. Again, 1/4 bias is used here.

The components for calculating the RMS waveforms for ON and OFF cases of quadruplex muxing are shown in **Figure 10**.

The voltages are calculated as:

$$V_{RMSON} = \sqrt{\frac{1}{4} \cdot \left[V_{LCD}{}^{2} + 3 \cdot \left(\frac{V_{LCD}}{3} \right)^{2} \right]} = 0.577 \cdot V_{LCD}$$

$$V_{RMSOFF} = \sqrt{\frac{1}{4} \cdot 4 \cdot \left(\frac{V_{LCD}}{3} \right)^{2}} = 0.333 \cdot V_{LCD}$$

It should now be obvious that as the amount of multiplexing increases, the RMS voltages decrease. Contrast, measured as the ratio of V_{RMSON}/V_{RMSOFF}, is called the discrimination ratio.

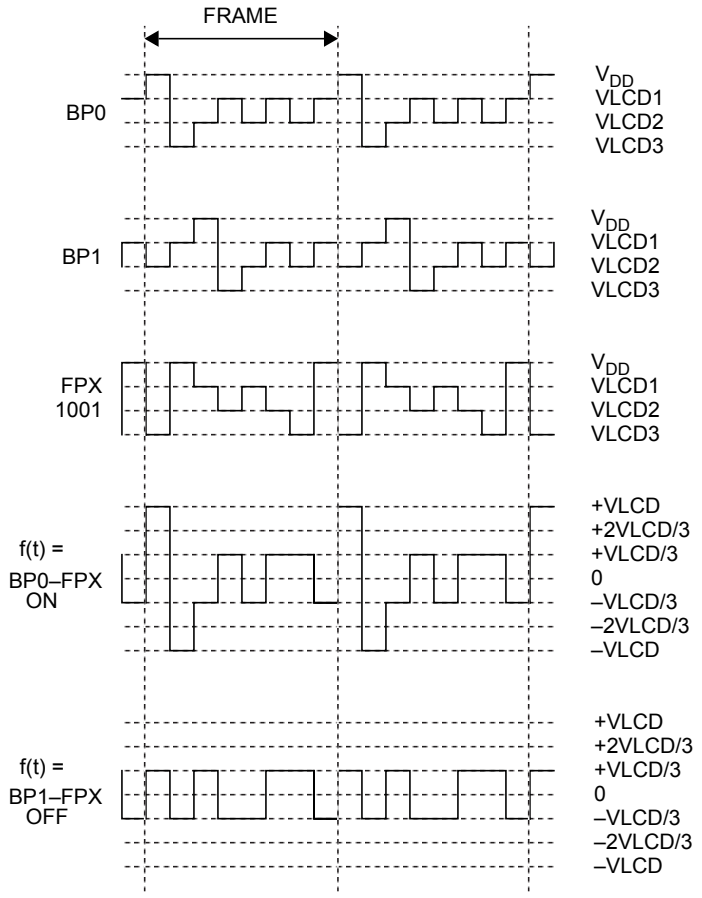

Figure 9. LCD 1/4 Duty and 1/3 Bias Timing Diagram
$V_{LCD1} = V_{DD}–V_{LCD}/3$, $V_{LCD2} = V_{DD}–2V_{LCD}/3$, $V_{LCD3} = V_{DD}–V_{LCD}$

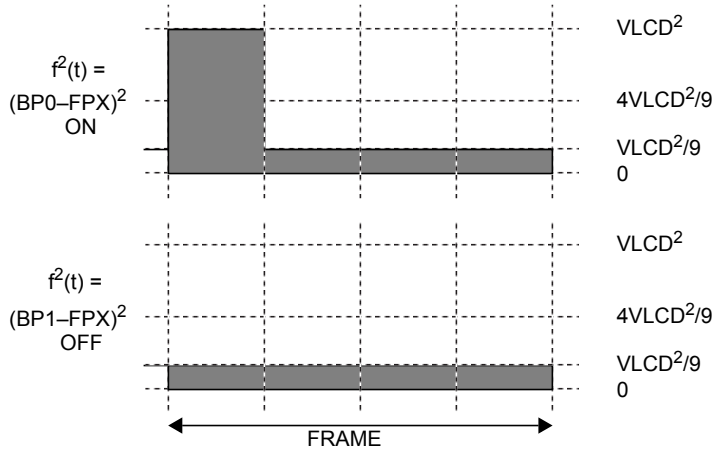

Figure 10. Waveform Components for Calculating ON and OFF RMS Voltages

Sample Application

To demonstrate how simple it is to use the MC68HC705L16 to drive an LCD panel directly, a simple application is described in which a text message is displayed on an 8-digit, 15-segment display (Planar-Standish Model 4228). The display will be driven by 1/4 duty and 1/3 bias.

Figure 11 shows a schematic diagram of the circuit with all connections labelled. A resistor divider from V_{DD} is used to generate the three voltage levels, V_{LCD1}, V_{LCD2}, and V_{LCD3}, for the waveforms. A variable resistor at the bottom of the divider allows manual contrast adjustment. The four backplane pins from the MCU are connected to the four common pins on the LCD panel. Since the 1/4 duty is being used, four segments can be driven by each frontplane driver and, therefore, 32 frontplanes drivers are needed. The first 32 frontplane pins from the MCU are connected to the LCD panel, while the remaining seven are not used.

These connections from the MCU to the LCD panel determine the mapping of the LCD data registers to the segments of the LCD panel. Each digit on the panel is composed of 16 segments, controlled by two consecutive 8-bit LCD data registers. Each LCD data register controls two frontplanes. Therefore, four frontplanes are required to drive each digit of the display. **Figure 12** shows the mapping of the register bits to the segments in one of the characters on the display.

For example, the letter G would be represented by the two bytes: $05E4. The first byte in register LCDR1 would be 00000101. The second byte in register LCDR2 would be 11100100. Together, the lit segments would create the letter G. See **Figure 13**.

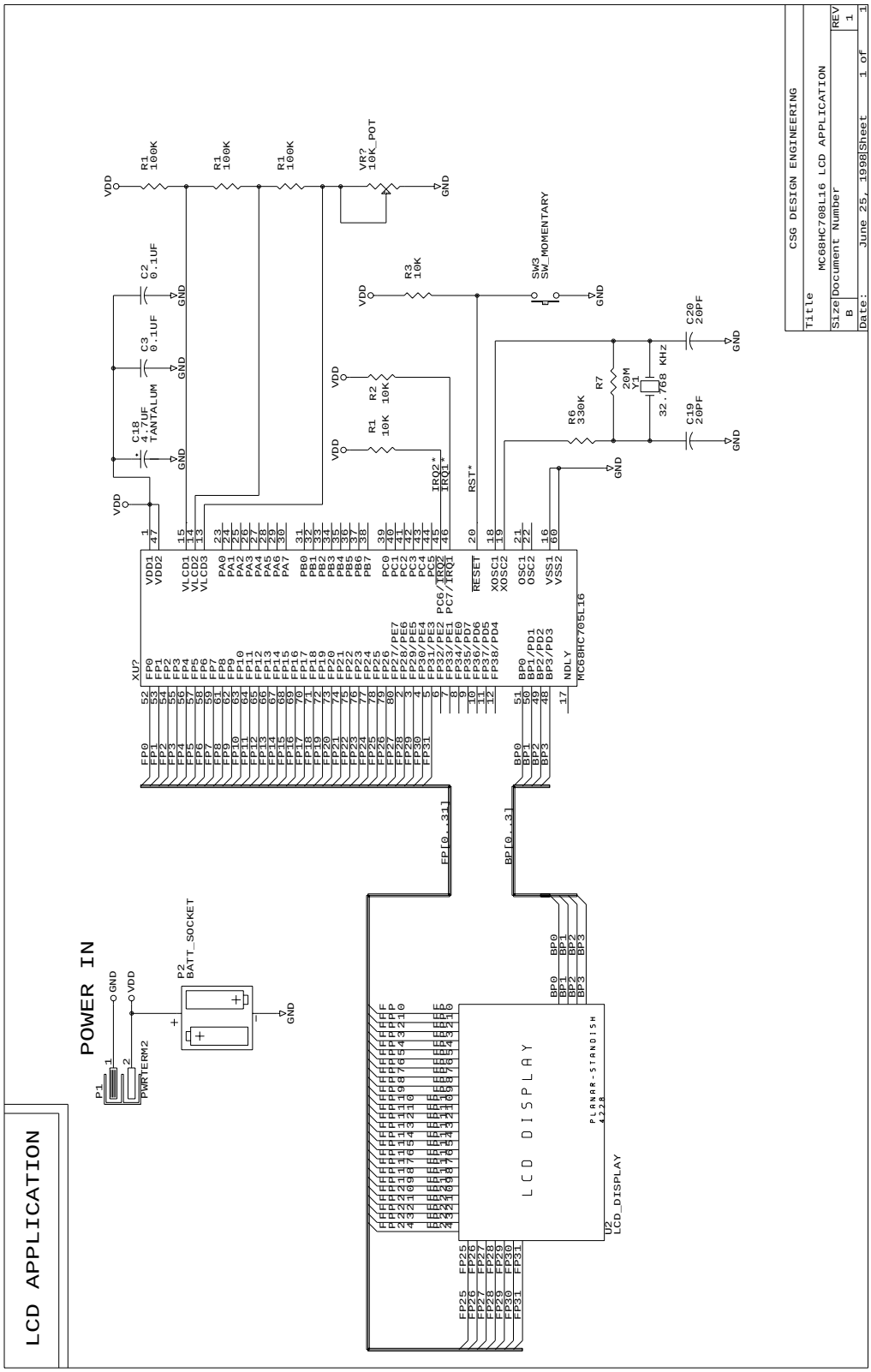

Figure 11. Schematic Diagram of Sample LCD Application

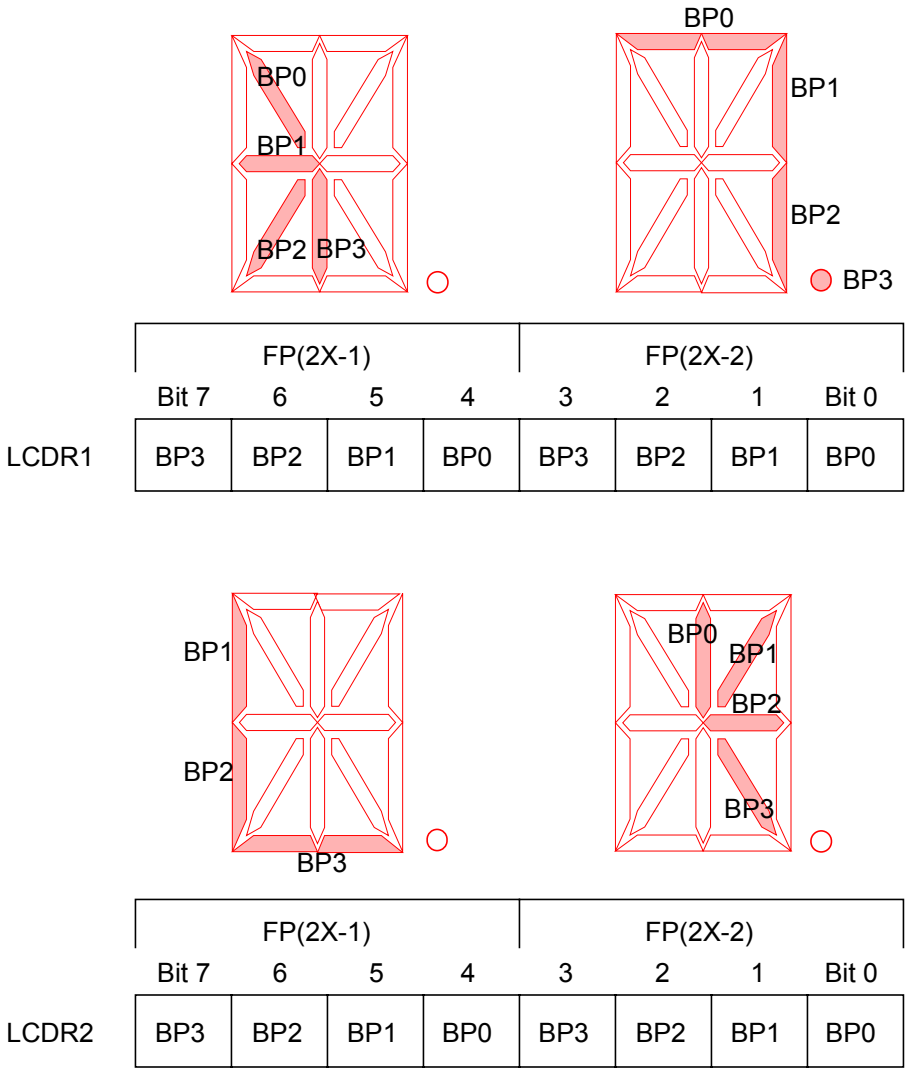

Figure 12. Mapping of LCD Register Bits to Display Segments

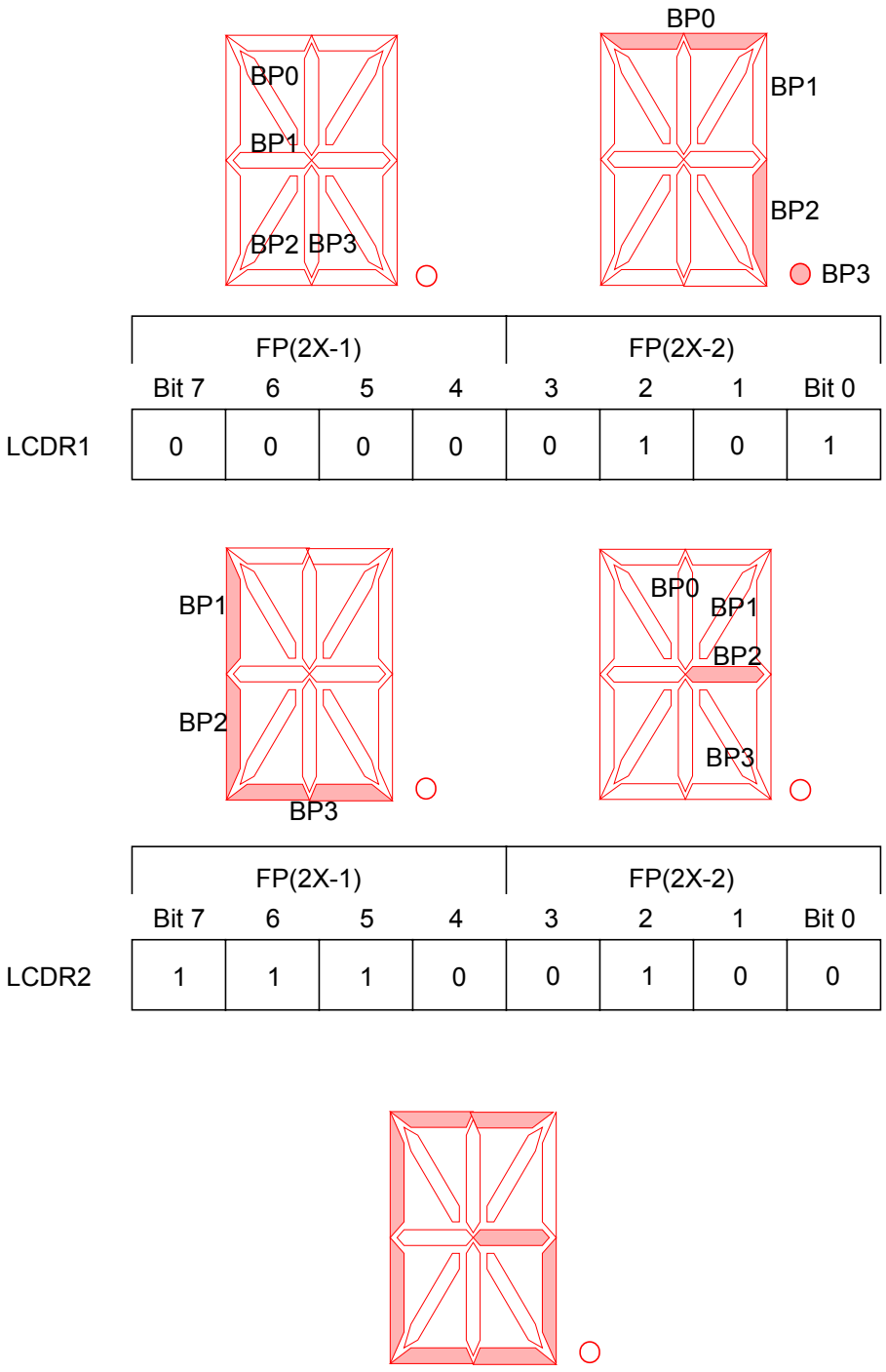

Figure 13. Example Display of the Letter G

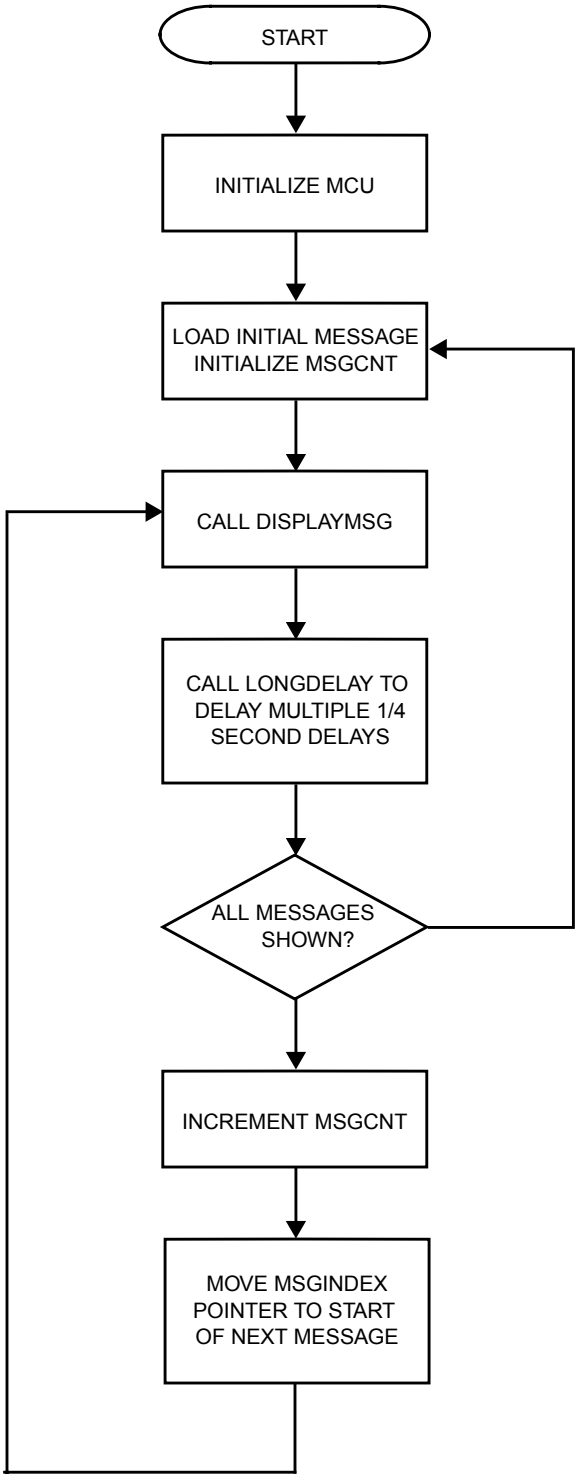

Figure 14. Main Program Flow

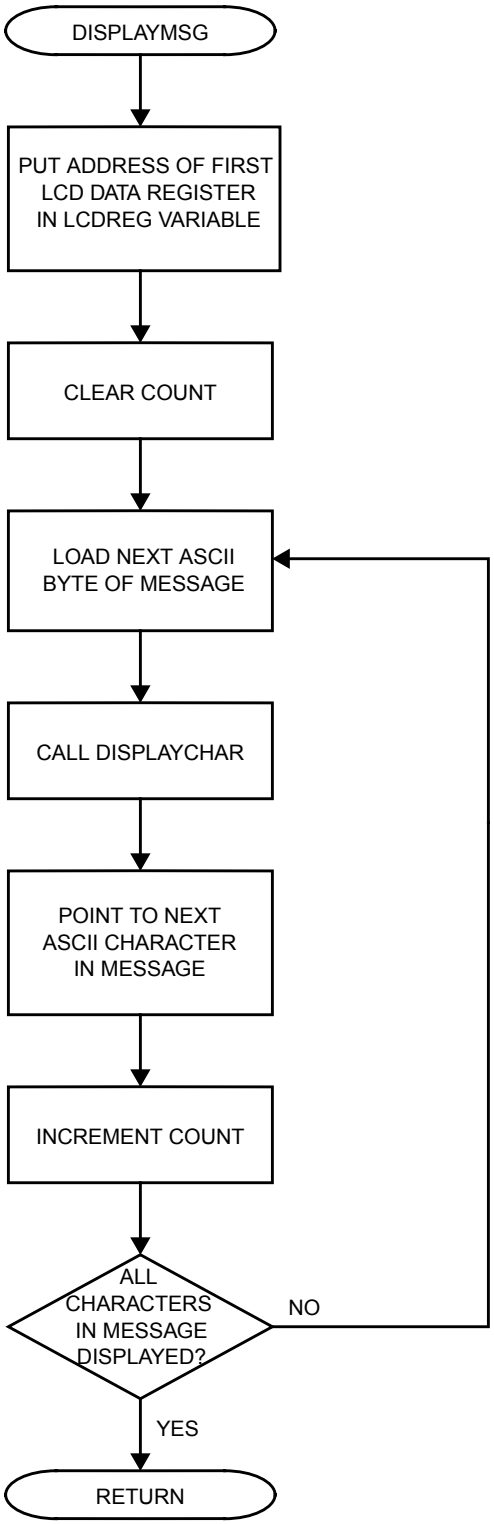

Figure 15. DisplayMsg Subroutine

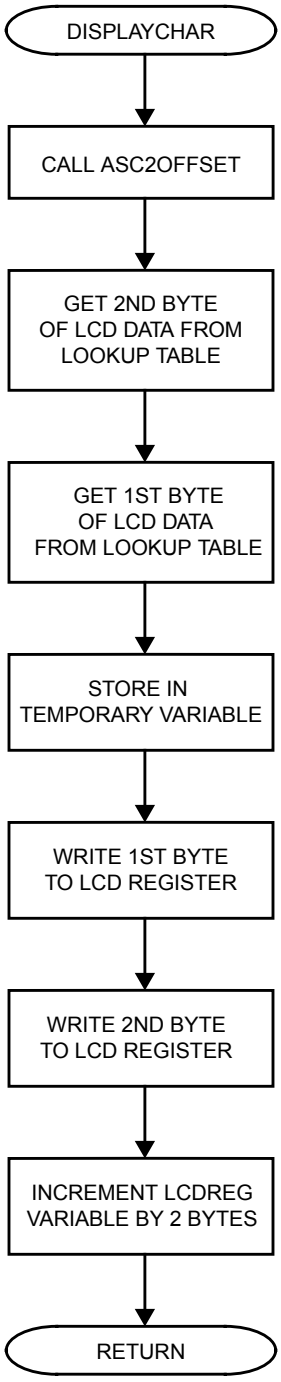

Figure 16. DisplayChar Subroutine

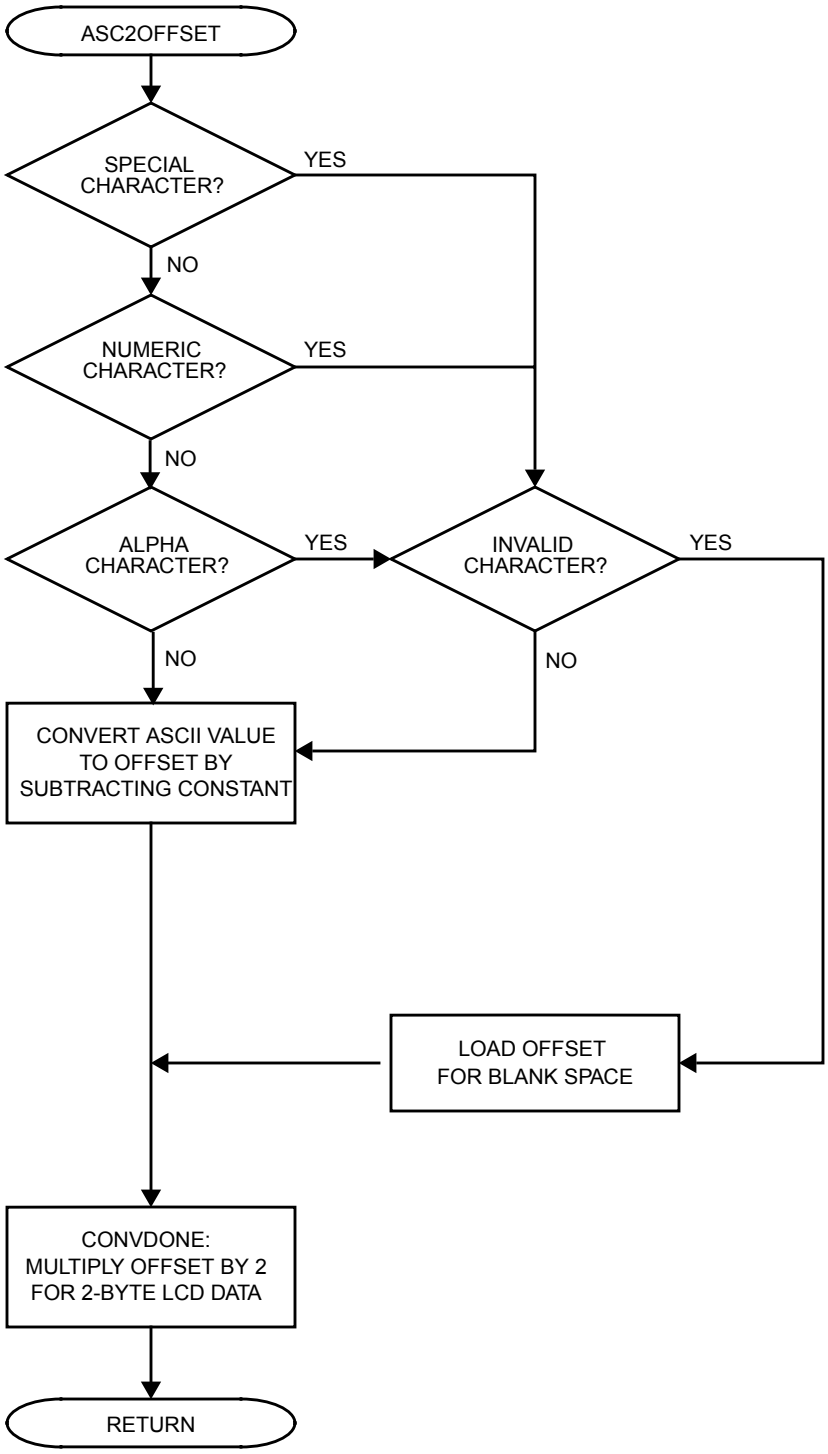

Figure 17. ASC2OFFSET Subroutine

Code Listings

```
*****************************************************************************
* LCD_DISPLAY.ASM
*****************************************************************************
* Ed Stellini, 06/06/98
* CSG Design Engineering
* Freescale
*
* Software written to demonstrate direct drive of LCD display
* using MC68HC705L16 microcontroller.
*
* The LCD used is a Planar-Standish Model 4228 Multiplex
* 15-segment, 8-digit panel. (1/4 duty, 1/3 bias)
*
*****************************************************************************
$BASE          10T                          ;Default assembler number base

*****************************************************************************
* Memory Equates
*****************************************************************************
RAMSPACE       EQU      $0040               ;Start of user RAM
ROMSPACE       EQU      $1000               ;Start of user ROM
RESETVEC       EQU      $FFFE               ;Reset vector

*****************************************************************************
* Register Equates
*****************************************************************************
* Registers
MISC           EQU      $3E                 ;Miscellaneous register
TBCR1          EQU      $10                 ;Time base control register 1
LCDCR          EQU      $20                 ;LCD control register
LCDDR          EQU      $21                 ;First LCD data register location

* Bit locations
LCDE           EQU      $07                 ;LCD enable bit in LCDCR
SYS0           EQU      $02                 ;SYS0 bit in MISC
SYS1           EQU      $03                 ;SYS1 bit in MISC

*****************************************************************************
* LCD Equates
*****************************************************************************
MAXCHARS       EQU      $08                 ;Maximum characters per line of LCD
NUMMSGS        EQU      $05                 ;Number of message lines to display
QTRSECS        EQU      $14                 ;20 quarterseconds = 5 seconds
EOT            EQU      $04                 ;End of string marker (ASCII EOT)
```

```
;*******************************************************************************
; RAM Variables
;*******************************************************************************
              ORG     RAMSPACE              ;Start of user RAM
TempX         RMB     1                     ;Temporary register storage
TempA         RMB     1                     ;Temporary register storage
TempData      RMB     1                     ;Temp storage for LCD segment data
LCDReg        RMB     1                     ;8-bit address pointer
Count         RMB     1                     ;Counter variable
MsgIndex      RMB     1                     ;Index counter variable
MsgCount      RMB     1                     ;Current message count

;*******************************************************************************
; Start of program code
;*******************************************************************************
              ORG     ROMSPACE              ;Start of user EPROM
Start         BCLR    SYS0,MISC             ;Setup for f_op = f_osc/2
              BCLR    SYS1,MISC
              LDA     #$20                  ;XOSC for time base
              STA     TBCR1                 ;LCD clock = XOSC/128 = 256Hz
              BSET    LCDE,LCDCR            ;Enable LCD

;*******************************************************************************
; Initialize string to be initially displayed.
;*******************************************************************************
Initial       LDA     #Msg1                 ;Load offset of desired string
              STA     MsgIndex              ;Setup the message index
              LDA     #!1                   ;
              STA     MsgCount              ;

;*******************************************************************************
; Main loop
; Display each message from memory in sequential order.
;*******************************************************************************
MainLoop      LDX     MsgIndex              ;Start at current message
              JSR     DisplayMsg            ;Show current message
              JSR     LongDelay             ;Delay for #QTRSECS quarterseconds

              LDA     MsgCount              ;Get current count
              CMP     #NUMMSGS              ;Check if through all messages
              BEQ     Initial               ;Start with Msg1 again
              INC     MsgCount              ;Next message
              LDA     MsgIndex              ;Get current message index
              ADD     #MAXCHARS             ;Move index to next message
              STA     MsgIndex              ;Store new message index
              BRA     MainLoop              ;Repeat
```

```
*****************************************************************************
* SUBROUTINES
*****************************************************************************

*****************************************************************************
* Show the current string portion on the display.
* When called, the X register contains the index offset.
*****************************************************************************
DisplayMsg      LDA     #LCDDR              ;First LCD data register
                STA     LCDReg              ;LCDReg = First LCD data register
                CLR     Count               ;Clear the counter variable
NextByte        LDA     Msgs,X              ;Load ASCII byte of string
                JSR     DsplayChar          ;Display character
                INCX                        ;Increment the index
                INC     Count               ;Increment the counter
                LDA     Count               ;Check the counter
                CMP     #MAXCHARS           ;for LCD display length
                BEQ     Done                ;End of display line reached
                BRA     NextByte            ;Ready the next byte
Done            RTS                         ;Return

*****************************************************************************
* DsplayChar converts an ASCII character value in Register A to
* an offset into the character table. The two bytes at the offset
* location of the table define the segment values for displaying
* the character on the display. Then use the offset into the LCD
* data table to get the 2 bytes for the LCD position, and store
* them in the appropriate LCD data registers.
*****************************************************************************
DsplayChar      STX     TempX               ;Save X register
                JSR     ASC2Offset          ;Convert ASCII byte into table offset
                TAX                         ;Put offset into X
                LDA     Table+1,X           ;Get second LCD data byte
                STA     TempData            ;Store it temporarily
                LDA     Table,X             ;Load A with first LCD data byte
                LDX     LCDReg              ;Point X to current LCD data register
                STA     0,X                 ;Store first byte to LCD data register
                LDA     TempData            ;Load A with second data byte
                STA     1,X                 ;Store it to second LCD data register
                INC     LCDReg              ;Increment LCDreg pointer to
                INC     LCDReg              ;point to the next position's regs.
                LDX     TempX               ;Restore X register
                RTS                         ;Return
```

```
*******************************************************************************
* Convert ASCII character byte in A to an offset value into
* the table of LCD segment values.
* The software also checks for an invalid or unusable ASCII
* character value, and shows a blank space in its place.
* Valid ASCII values are (decimal): 32-47, 48-57, 65-90
*******************************************************************************
ASC2Offset      CMP     #!48                ;Check for "special" character
                BLO     Special
                CMP     #!65                ;Check for numeric character
                BLO     Numeric
Alpha           CMP     #!90                ;Check for invalid value
                BHI     ConvError
                SUB     #!39                ;Convert to table offset
                BRA     ConvDone
Special         CMP     #!32                ;Check for invalid value
                BLO     ConvError
                SUB     #!32                ;Convert to table offset
                BRA     ConvDone
Numeric         CMP     #!57                ;Check for invalid value
                BHI     ConvError
                SUB     #!32                ;Convert to table offset
                BRA     ConvDone
ConvError       CLRA                        ;Invalid value shows as blank space
ConvDone        ROLA                        ;Multiply offset by 2
                RTS                         ;(2 bytes data per LCD position)
*******************************************************************************
* BlankSpace shows a space ($0000) at the current display position's
* LCD data registers.
*******************************************************************************
BlankSpace      LDX     LCDReg              ;Point to current LCD data register
                CLR     0,X                 ;Clear first data byte
                CLR     1,X                 ;Clear second data byte
                INC     LCDReg              ;Increment LCDreg pointer to
                INC     LCDReg              ;point to the next position's regs.
                RTS                         ;Return

*******************************************************************************
* Delay for number of quarterseconds = QTRSEC
*******************************************************************************
LongDelay       CLRX                        ;
Delayloop       LDA     #!250               ;Load accumulator with #ms to delay
                JSR     Delay               ;Jump to #ms delay subroutine
                INCX                        ;
                TXA
                CMP     #QTRSECS            ;
                BEQ     Finish              ;
                BRA     Delayloop           ;
Finish          RTS                         ;Return
```

```
*****************************************************************************
* Delay for time = Accumulator*1ms (fop = 1MHz)
* Accumulator contains the number of 1ms delays desired
*****************************************************************************
Delay           CMP     #$00                ;Check for remaining delays
                BEQ     DDone               ;Done?

MsDelay         STA     TempA
                LDA     #$5A
MsLoop          CMP     #$00
                BEQ     MsDone
                DECA
                BRA     MsLoop
MsDone          LDA     TEMPA

                DECA                        ;Decrement count
                BRA     Delay               ;Repeat
DDone           RTS                         ;Return

*****************************************************************************
* ROM Constants
*****************************************************************************
*****************************************************************************
* LCD Messages
* Each individual message is identified by its offset into the
* base address labelled Msgs.
* This limits user to 8 bits of offset (255 characters worth.
* If more than 255 characters are desired for messages, one can
* use some 2-byte variable which can contain multiple base addresses.
*
* Valid characters are 0-9, A-Z (UPPERCASE ONLY!), and certain
* special characters defined in the table as valid.
*****************************************************************************
Msgs            EQU     *                   ;Base address of messages
*****************************************************************************
Msg1            EQU     *-Msgs              ;First message offset
                FCB     "THE L16 "
*****************************************************************************
Msg2            EQU     *-Msgs              ;Second message offset
                FCB     "DRIVES  "
*****************************************************************************
Msg3            EQU     *-Msgs              ;Third message offset
                FCB     "LCD     "
*****************************************************************************
Msg4            EQU     *-Msgs              ;Fourth message offset
                FCB     "DISPLAYS"
*****************************************************************************
Msg5            EQU     *-Msgs              ;Fifth message offset
                FCB     "DIRECTLY"
*****************************************************************************
EndMsgs         EQU     *-Msgs              ;End of messages label
```

```
*******************************************************************************
* Lookup table of LCD segment values for ASCII character values
* Some characters can not be displayed on 15-segment LCD, so
* they are marked as invalid, and will be displayed as a blank space.
*******************************************************************************
Table           FDB     $0000                   ;' '
                FDB     $0000                   ;'!' INVALID
                FDB     $0201                   ;'"'
                FDB     $0000                   ;'#' INVALID
                FDB     $A5A5                   ;'$'
                FDB     $0000                   ;'%' INVALID
                FDB     $0000                   ;'&' INVALID
                FDB     $0001                   ;'''
                FDB     $000A                   ;'('
                FDB     $5000                   ;')'
                FDB     $F00F                   ;'*'
                FDB     $A005                   ;'+'
                FDB     $0000                   ;',' INVALID
                FDB     $2004                   ;'-'
                FDB     $0800                   ;'.'
                FDB     $4002                   ;'/'
                FDB     $47E2                   ;'0'
                FDB     $0602                   ;'1'
                FDB     $23C4                   ;'2'
                FDB     $2784                   ;'3'
                FDB     $2624                   ;'4'
                FDB     $21A8                   ;'5'
                FDB     $25E4                   ;'6'
                FDB     $0700                   ;'7'
                FDB     $27E4                   ;'8'
                FDB     $27A4                   ;'9'
                FDB     $2764                   ;'A'
                FDB     $8785                   ;'B'
                FDB     $01E0                   ;'C'
                FDB     $8781                   ;'D'
                FDB     $21E4                   ;'E'
                FDB     $2164                   ;'F'
                FDB     $05E4                   ;'G'
                FDB     $2664                   ;'H'
                FDB     $8181                   ;'I'
                FDB     $06C0                   ;'J'
                FDB     $206A                   ;'K'
                FDB     $00E0                   ;'L'
                FDB     $1662                   ;'M'
                FDB     $1668                   ;'N'
                FDB     $07E0                   ;'O'
                FDB     $2364                   ;'P'
                FDB     $07E8                   ;'Q'
                FDB     $236C                   ;'R'
                FDB     $25A4                   ;'S'
                FDB     $8101                   ;'T'
```

```
                FDB     $06E0                   ;'U'
                FDB     $4062                   ;'V'
                FDB     $4668                   ;'W'
                FDB     $500A                   ;'X'
                FDB     $9002                   ;'Y'
                FDB     $4182                   ;'Z'
EndTable        EQU     *-Table                 ;End of table label

*******************************************************************************
* Vector definitions
*******************************************************************************
                ORG     RESETVEC                ;Reset vector
                FDB     Start
```

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