AN1743

Scrolling Message Software

By Brad Bierschenk
Consumer Systems Group, Applications Engineering
Austin, Texas

Freescale Semiconductor, Inc.

Introduction

Many MCU applications use displays, such as LCD or LED panels, to provide useful output. Modern displays are an efficient and affordable way for microcontrollers to communicate with the outside world.

However, one limitation of such displays is the amount of information that can be presented at one time. To output a message that is longer than its display, MCU software needs a method to “scroll” information across the display screen. This method should be divided into independent tasks, allowing for normal paced-loop program execution. This application note documents such a technique.

LCD Displays

Many different types of displays are used in MCU applications, but the most common is the LCD (liquid crystal display). These can come in a wide variety of styles.

© Freescale Semiconductor, Inc., 2004. All rights reserved.

© Motorola, Inc., 1998

For More Information On This Product,
Go to: www.freescale.com

The two most basic LCD styles typical of 8-bit MCU applications are:

- Dot-matrix character display
- Segmented character display

Dot-matrix Displays

Dot-matrix character displays usually have on-board controllers, which handle the character mapping that converts ASCII character input to dot-matrix character output. The LCD controller also generates the driving waveforms for the display. These displays are commonly accessed by a serial connection, using command and data bytes to control the LCD module. These types of displays are easy to interface, but tend to be larger and more expensive.

Segmented Displays

Segmented LCD displays, on the other hand, generally are interfaced in a parallel fashion. Each numeric or alpha-numeric digit is composed of a specific number of segments, usually seven to 16 per digit. Each segment of the display has its own input line. To decrease the number of input lines required, displays can be multiplexed. This is done by the use of multiple backplanes, referred to as the “duty” of the display (for example, 1/4 duty implies the use of four backplanes for multiplexing). This allows one frontplane line to control several segments.

A segmented, multiplexed LCD display is controlled by the use of waveforms which provide various voltage levels, referred to as the “bias” of the display. The theory behind these waveforms is beyond the scope of this document. The control waveforms can be generated by software control of I/O pins, by a separate driver chip, or by dedicated MCU circuitry.

Some Freescale MCUs provide built-in LCD drivers. One specific example is the MC68HC705L16 microcontroller. This useful MCU can effectively drive an LCD with up to 156 segments. The output of the drivers is controlled by data register values.

The method presented in this document assumes the use of a segmented display. This method can be adapted easily to other displays, such as dot-matrix type LCD modules, with a few changes in software. The use of an “intelligent” dot-matrix LCD would decrease

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

software complexity and memory requirements, and would require a serial or parallel MCU interface.

The 68HC705L16 Microcontroller

The MC68HC705L16 microcontroller is especially suited for LCD applications. It provides an internal LCD driver, which supports up to four backplanes and 39 frontplanes, for control of 156 segments through 43 pins.

An external resistor ladder provides the bias levels for the LCD waveform. The output waveforms are generated automatically by the MCU, which is driven by data registers. The user simply writes to the data registers to control the LCD segments.

A Software Method of Scrolling

If an MCU application requires visual output, predetermined “canned” messages will be displayed. If the message shown is longer than the display, the message needs to progress across the display. In a paced-loop program structure typical of MCU software, tasks are executed in a deliberate order. The software technique must allow for normal task processing, while appearing to scroll a message across the display continuously.

Message Storage

The software stores messages as ASCII character strings in memory. The end of a message is marked by a special end-of-text character. The start of each message is identified by its offset from a base address.

The base address of a group of message strings is identified with a label. This allows the beginning of a particular message string to be calculated as an offset from the base address. Because the message strings and their characters are referenced using indexed addressing, blocks of messages are limited to 255 bytes. String storage capability can be

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

extended by using multiple base address labels (for instance, ErrorMsgs, WarningMsgs, and InputMsgs) to label strings categorically.

The first step in displaying a message using this method is to identify the string to be shown. Two index variables keep track of the starting and current offset of the message string. The index variables and the base address of a message group are used to access the character data from the string. The main loop of the program is where the user's normal tasks would be carried out. The scrolling software should not impede the execution of other system tasks.

Displaying Messages

The routine to update the display is called as a normal task in the main loop. This accomplishes the goal of scrolling the string by showing successive portions of a message. After a display's worth of characters are shown, the message index is incremented. Once the end of the string has been reached, the software continues to scroll the string off the display, “padding” unused display positions with blank spaces. Once the message has scrolled off the display, the software resets the message index variables and the message is displayed again from its beginning.

Message strings are stored in the MCU as ASCII character values. There needs to be a way of relating the ASCII character bytes to LCD data register values. LCD data register values are a bitmap of segment values for a particular LCD digit. By setting the segment values appropriately, characters can be represented on the display.

Character Conversion

The relationship between ASCII characters and LCD segment data is handled by a lookup table. Each entry in the table contains two bytes which represent the segment values required to display a particular character. A conversion subroutine is called, with the ASCII character value to be converted as an argument. The conversion checks to see if the character is a valid alphabetic, numeric, or a special character. A predefined operation on the ASCII value converts it into an offset into the lookup table.

The segment bitmap for the character can then be accessed, using the offset and the base address of the lookup table. After this conversion, a character can be displayed by taking its data bytes from the table and

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

placing them in the appropriate LCD data registers. This process is repeated for all the data registers that correspond to the frontplane outputs being used for the display.

By showing one display worth of characters at a time, the work of outputting a scrolling message can be divided into discrete time segments. Incrementing the index into the current string before updating the display gives the impression that a message is scrolling continuously across the display.

Sample Application

In this simple application, an 8-digit, 15-segment display (Planar-Standish Model 4228) is used to show text messages. This particular display has four backplane pins and 32 frontplane pins. Connections are made to a Freescale MC68HC705L16 microcontroller through an emulator module. The four backplane lines from the MCU are connected to the four common backplane pins of the LCD panel, and the first 32 frontplane lines from the MCU are connected to the 32 frontplane pins of the display.

Table 1 shows the connections made between the MCU and the LCD panel.

Table 1. Connections between the MCU and LCD Panels

L16 MCU pin	LCD panel function	LCD panel pin number
BP0	COM1	21
BP1	COM2	40
BP2	COM3	1
BP3	COM4	20
FP0	8A	38
FP1	8B	39
FP2	8C	3
FP3	8D	2
FP4	7A	36
FP5	7B	37
FP6	7C	5

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 1. Connections between the MCU and LCD Panels (Continued)

L16 MCU pin	LCD panel function	LCD panel pin number
FP7	7D	4
FP8	6A	34
FP9	6B	35
FP10	6C	7
FP11	6D	6
FP12	5A	32
FP13	5B	33
FP14	5C	9
FP15	5D	8
FP16	4A	28
FP17	4B	30
FP18	4C	12
FP19	4D	10
FP20	3A	26
FP21	3B	27
FP22	3C	15
FP23	3D	14
FP24	2A	24
FP25	2B	25
FP26	2C	17
FP27	2D	16
FP28	1A	22
FP29	1B	23
FP30	1C	19
FP31	1D	18

The connections between the frontplane drivers and the LCD panel determine the segment assignments of the LCD data registers. **Figure 1** illustrates the meaning in this particular application. There are two LCD data registers for each position of the LCD display.

Each bit in an LCD data register represents a segment in an LCD position. Therefore, each table entry stores the 16 segment values necessary to display a given character on the display. **Table 2** shows the segment bit-mapping for this application.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

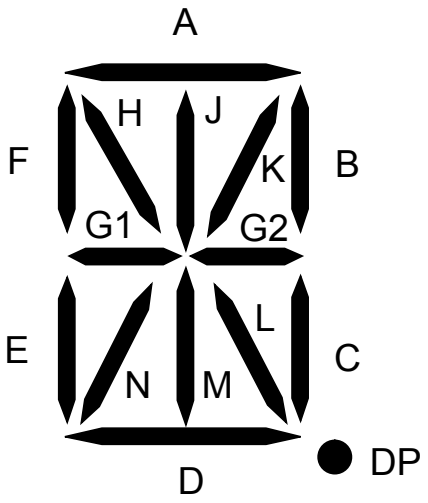

Figure 1. LCD Segment Assignments

Table 2. Segment Bit-Mapping

Register	B7	B6	B5	B4	B3	B2	B1	B0
LDATn	M	N	G1	H	DP	C	B	A
LDATn+1	D	E	F	—	L	G2	K	J

For example, to display the letter A, the segments A, B, C, E, F, G1, and G2 need to be lit. This would require data register values of $27 and $64 in the corresponding LDATn and LDATn+1 registers.

A resistor ladder is connected to the VLCD1, VLCD2, and VLCD3 pins to provide the voltage levels for the LCD waveform. A variable resistor in the ladder allows the display contrast to be adjusted.

Figure 2 shows a circuit diagram of relevant connections to the LCD display. Common connections (power supply, oscillator, etc.) are not shown.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

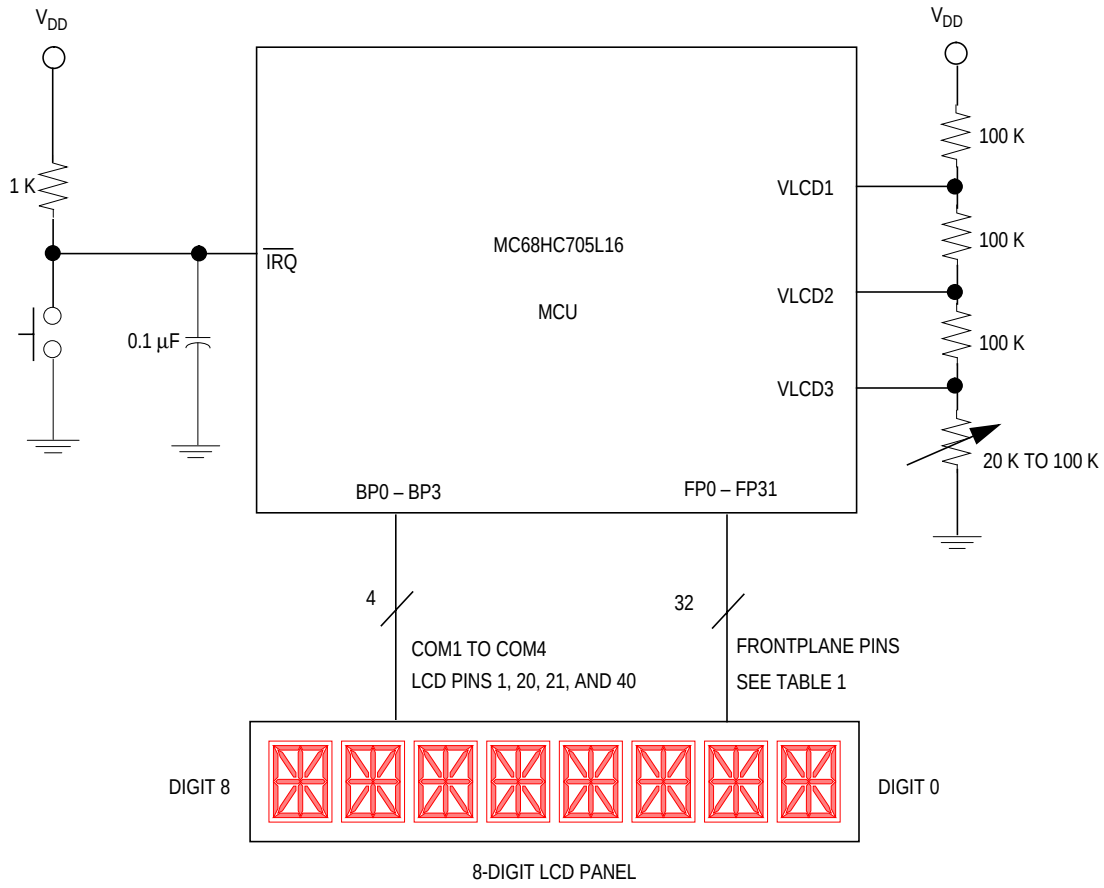

Figure 2. LCD Connections

A button switch connected to the IRQ line allows triggering of an external interrupt. On an external interrupt, the service routine loads the next message in the message block.

For this application, a lookup table is specified. The size of the lookup table is determined by the flexibility of the display. **Table 3** shows how an ASCII value is converted to an offset into the lookup table in this application.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 3. Lookup Table

Character types	ASCII value (decimal)	Table offset (decimal)	Conversion operation
Special	32 – 47	0 – 15	ASCII – 32
Numeric	48 – 57	16 – 25	ASCII – 32
Alphabetic	65 – 90	26 – 51	ASCII – 39

Once the offset is calculated, it is multiplied by 2 because there are two segment data bytes for every character. The software also checks for invalid values 0–31, 58–64, and 91–255 (ASCII decimal). These values are invalid because they cannot be displayed on the LCD panel.

This application is intended to be a simple demonstration of the scrolling message software, but it could be expanded easily to provide more functionality.

This method can also be adapted for connection to a smart LCD module. In this case, the routine ShowChar would be modified to display a character differently, but all other program flow would remain the same. The method of connection should not affect the basic scrolling algorithm.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Conclusion

Alternatives and Trade-offs

There are several methods of integrating an LCD into a microcontroller system. Trade-offs in cost, complexity, and convenience must be considered.

Table 4 illustrates the advantages and disadvantages of different LCD implementations.

Table 4. LCD Connection Methods

Method	Advantages	Disadvantages
MCU with on-board hardware drivers and raw glass	Fewer components Reliable LCD output Application e xibility	Requires specialized MCU
MCU with software drivers and raw glass	Fewer components Wide range of MCUs	More software overhead
MCU, LCD driver chip, and raw glass	Less software overhead Wide range of MCUs	More components
MCU and smart LCD module	Less software overhead Fewer components Wide range of MCUs	Higher cost

Software Drivers

The most basic method is to drive a display panel through software which generates LCD waveforms.

The advantages of this method:

- It can be implemented with practically any MCU.
- Costs will be minimized.

The disadvantage:

- It requires much more software overhead.

"Smart" LCD Modules

The most convenient method is to connect a "smart" LCD module through a serial or parallel MCU connection. The MCU can send command and data bytes to the LCD module with a minimum amount of software or hardware overhead.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

The advantages of this method are:

- It has easy interface with practically any MCU.
- It requires less software and hardware overhead.

The disadvantages are:

- This method may be more expensive.
- The functionality might be limited to the capabilities of the LCD module.

LCD Driver ICs

A wide variety of integrated circuit LCD drivers is also available. These components can be used as an interface between any MCU and a glass panel.

The 705L16 MCU

The use of the MC68HC705L16 MCU provides a practical compromise between cost and complexity. The advantages of using the 705L16 include:

- The MC68HC705L16's 16,384 bytes of EPROM provide a large amount of storage for code and message strings.
- The MC68HC705L16's built-in LCD drivers provide reliable and autonomous LCD waveform generation.
- If combined with keypad input, the MC68HC705L16 and LCD display can provide a large amount of user input and output with one MCU.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

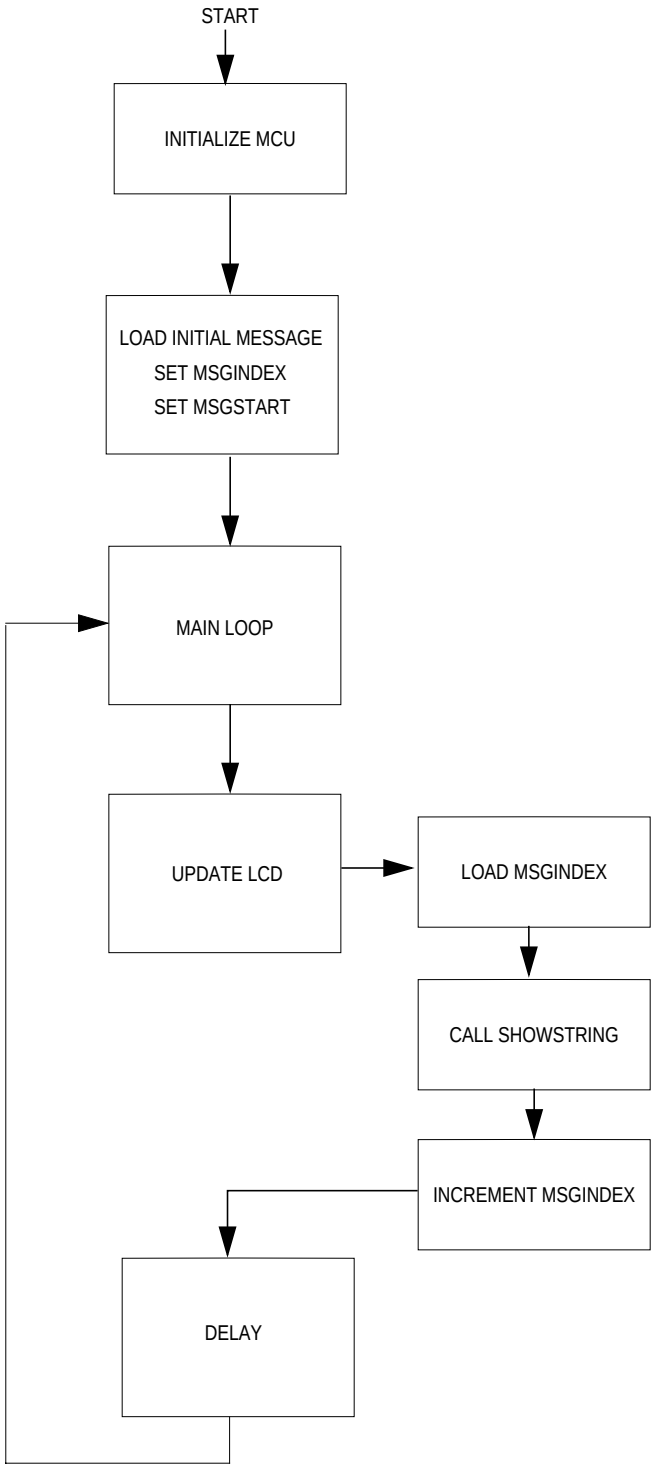

Figure 3. Main Program Flow

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

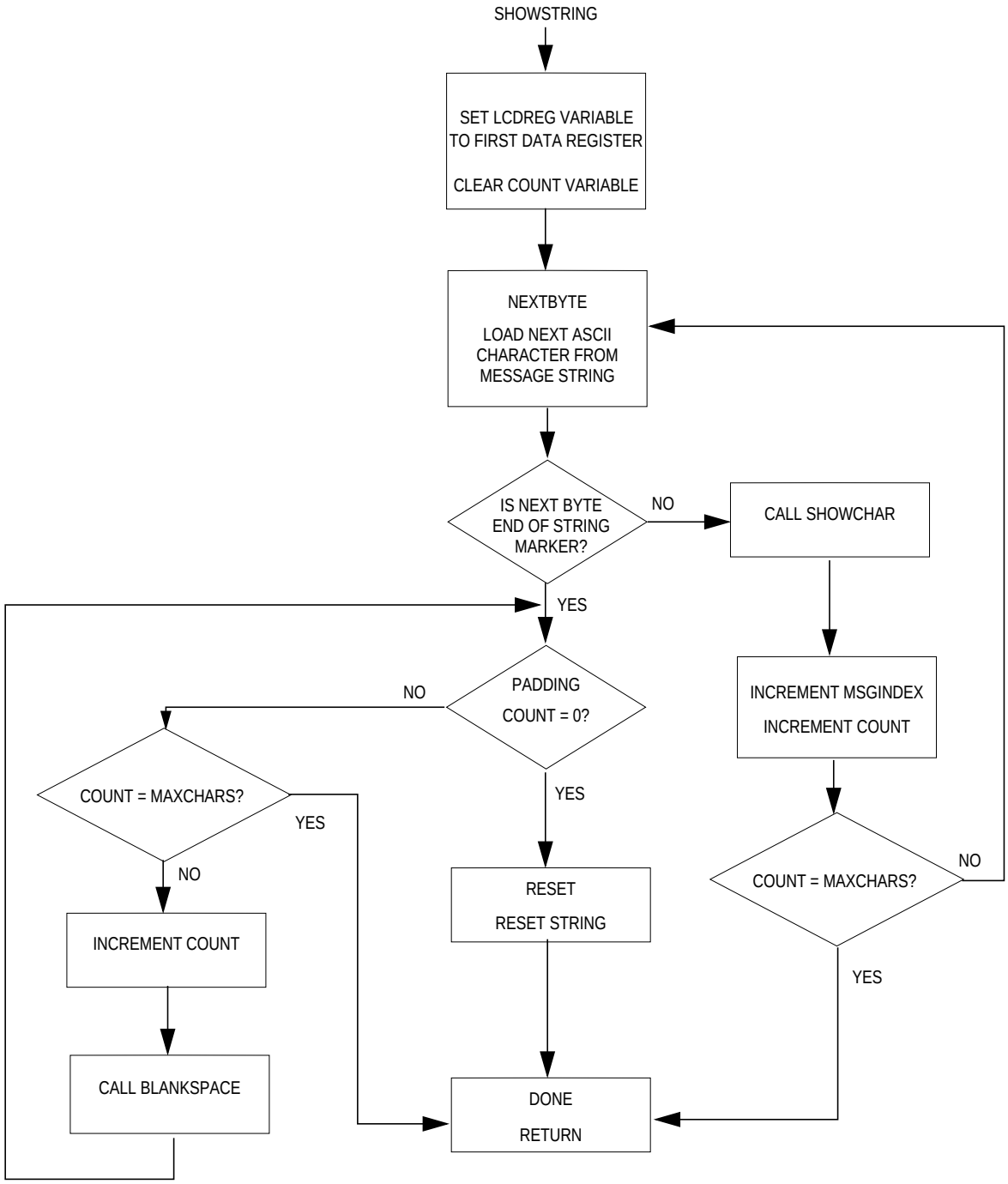

Figure 4. ShowString Subroutine

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

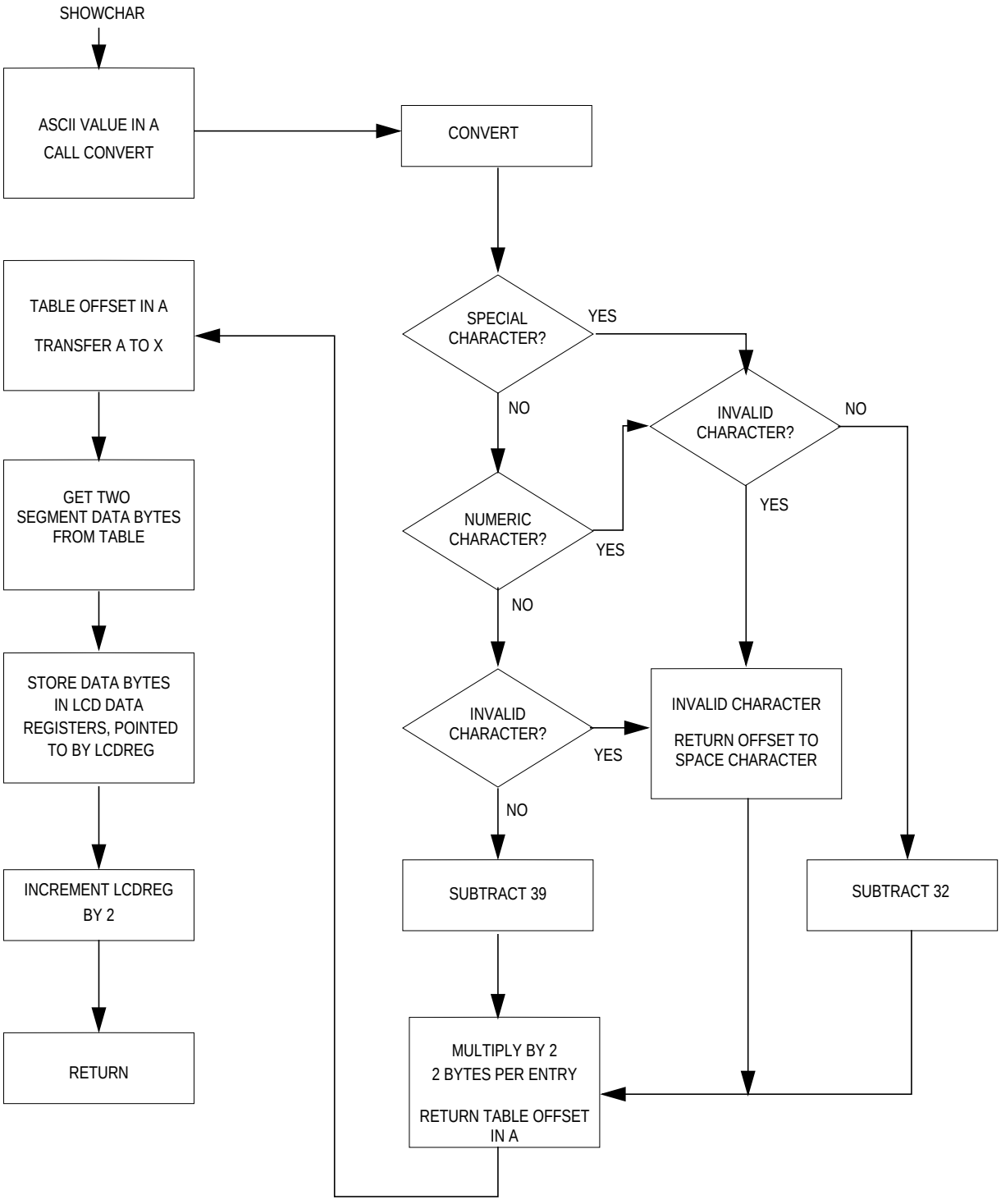

Figure 5. ShowChar Subroutine

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Code Listing

```
*******************************************************************************
* SCROLL.ASM
*******************************************************************************
* Brad Bierschenk, 03/23/98
* CSG Applications Engineering
* Freescale
*
* Software written to demonstrate scrolling long text messages across an LCD
* display.
*
* This is written for the MC68HC705L16, which provides built-in LCD drive
* capabilities.
* The LCD used is a Planar-Standish Model 4228 Multiplex 15-segment, 8-digit panel.
* (1/4 duty, 1/3 bias)
* An external interrupt provided by a button switch on IRQ1' selects the
* message to be displayed.
*
* Although this software was written for the 705L16 interface to raw LCD glass, it can
* easily modified for use with a smart LCD module and a serial interface with
* another MCU.
*
*------------------------------------------------------------------------------
$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
IRQVEC     EQU   $FFFA      ;IRQ' 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
INTCR      EQU   $08        ;Interrupt control register
INTSR      EQU   $09        ;Interrupt status register

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


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
IRQ1E       EQU   $07         ;IRQ1 enable bit in INTCR
IRQ1S       EQU   $03         ;IRQ1 sensitivity bit
IRQ1F       EQU   $07         ;IRQ1 flag bit in INTSR
RIRQ1       EQU   $03         ;Reset IRQ1 flag bit

*-----------------------------------------------------------------------------
* LCD Equates
*-----------------------------------------------------------------------------
MAXCHARS    EQU   $08         ;Maximum characters per line of LCD
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
MsgStart    RMB   1           ;Stores starting point of string

*-----------------------------------------------------------------------------
* 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
            BSET  IRQ1S,INTCR ;Set edge-level sensitivity
            BSET  IRQ1E,INTCR ;Enable IRQ1 interrupts
            BSET  RIRQ1,INTSR ;Clear IRQ1 flag
            CLI                ;Enable interrupts

*-----------------------------------------------------------------------------
* Initialize string to be initially displayed.
* When a new message is desired, the same LDA offset, JSR LoadMsg steps should
* be followed.
*-----------------------------------------------------------------------------
            LDA   #Msg1       ;Load offset of desired string
            JSR   LoadMsg     ;Initialize message variables

*-----------------------------------------------------------------------------
* Main loop
* UpdateLCD might be one of many tasks necessary in a paced-loop structure.
* If more tasks were implemented in the main loop, the delay would be adjusted
* (or eliminated) to provide the desired scroll rate.
*-----------------------------------------------------------------------------
```

Freescale Semiconductor, Inc.

```
MainLoop    JSR    UpdateLCD  ;Update the LCD display
            LDA    #!250
            JSR    Delay      ;Wait 250ms
            BRA    MainLoop   ;Repeat

*-----------------------------------------------------------------------------
* SUBROUTINES
*-----------------------------------------------------------------------------
* Initialize the message variables for the desired output string.
* Register A contains the offset of desired message.
*-----------------------------------------------------------------------------
LoadMsg     STA    MsgIndex   ;Setup the message index
            STA    MsgStart   ;Store the start of the message
            RTS               ;Return

*-----------------------------------------------------------------------------
* Update the LCD with current portion of string to be displayed.
*-----------------------------------------------------------------------------
UpdateLCD   LDX    MsgIndex   ;Start at current index into message
            JSR    ShowString ;Show current portion of string
            INC    MsgIndex   ;Increment the index
            RTS               ;Return

*-----------------------------------------------------------------------------
* Show the current string portion on the display.
* When called, the X register contains the index offset.
*-----------------------------------------------------------------------------
ShowString  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
            CMP    #EOT       ;Check for end of string
            BEQ    Padding    ;Last character reached
            JSR    ShowChar   ;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
Padding     LDA    Count      ;Pad the rest of the display with spaces
            CMP    #$00       ;See if string has scrolled off display
            BEQ    Reset      ;Need to reset string
            CMP    #MAXCHARS  ;Check for end of display
            BEQ    Done       ;Finished displaying padding spaces
            INC    Count      ;Increment counter
            JSR    BlankSpace ;Put space in current display position
            BRA    Padding    ;Repeat
Reset       JSR    BlankSpace ;Show a final space in first position
            LDX    MsgStart   ;Load start of message index
            DECX              ;Compensate for INCX in UpdateLCD after RTS
            STX    MsgIndex   ;Record new message index
Done        RTS               ;Return
```

Freescale Semiconductor, Inc.

```
*-------------------------------------------------------------------------------
* ShowChar 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
* offset into the LCD data table to get the 2 bytes for the LCD position and
* store them in the appropriate LCD data registers.
*-------------------------------------------------------------------------------
ShowChar    STX   TempX       ;Save X register
            JSR   Convert     ;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.
*-------------------------------------------------------------------------------
Convert     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)
```

Freescale Semiconductor, Inc.

```
*-----------------------------------------------------------------------------
* 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 ~Accumulator*1ms (fop = 1MHz)
* A contains the number of 1ms delays desired
*-----------------------------------------------------------------------------
Delay      CMP   #$00       ;Check for remaining delays
           BEQ   DDone      ;Done?
           BSR   MsDelay    ;Call 1ms delay routine
           DECA             ;Decrement count
           BRA   Delay      ;Repeat
DDone      RTS              ;Return

*-----------------------------------------------------------------------------
* Delay for ~1ms (fop = 1MHz)
*-----------------------------------------------------------------------------
MsDelay    STA   TempA
           LDA   #$5A
MsLoop     CMP   #$00
           BEQ   MsDone
           DECA
           BRA   MsLoop
MsDone     LDA    TEMPA
           RTS

*-----------------------------------------------------------------------------
* Interrupt service routine
* This allows a switch on IRQ1 to switch between message strings.
*-----------------------------------------------------------------------------
ISR        LDA   MsgStart   ;Start of current message
           CMP   #Msg1      ;Determine next message
           BEQ   Load2
           CMP   #Msg2
           BEQ   Load3
Load1      LDA   #Msg1      ;Load message 1
           BRA   Load
Load2      LDA   #Msg2      ;Load message 2
           BRA   Load
Load3      LDA   #Msg3      ;Load message 3
Load       JSR   LoadMsg
           BSET  RIRQ1,INTSR ;Clear IRQ1 flag
           RTI               ;Return
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
*-----------------------------------------------------------------------------
* ROM Constants
*-----------------------------------------------------------------------------
*-----------------------------------------------------------------------------
* "Canned" 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
* are defined in the table as valid.
*-----------------------------------------------------------------------------
Msgs       EQU   *          ;Base address of messages
;-----------------------------------------------------------------------------
Msg1       EQU   *-Msgs     ;First message offset
           FCB   "** Freescale MICROCONTROLLERS **"
           FCB   EOT        ;End of text (EOT) marker
;-----------------------------------------------------------------------------
Msg2       EQU   *-Msgs     ;Second message offset
           FCB   "SCROLLING MESSAGE DEMONSTRATION"
           FCB   EOT        ;End of text
;-----------------------------------------------------------------------------
Msg3       EQU   *-Msgs     ;Third message offset
           FCB   "705L16 LCD INTERFACE"
           FCB   EOT        ;End of text
;-----------------------------------------------------------------------------
EndMsgs    EQU   *-Msgs     ;End of messages label
```

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

```
*-------------------------------------------------------------------------------
* 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'
```

Freescale Semiconductor, Inc.

```
            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
           ORG    IRQVEC    ;IRQ vector
           FDB    ISR
```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Application Note

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

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 application, 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 Semiconductor, Inc.