AN1745

Interfacing the HC705C8A to an LCD Module

By Mark Glenewinkel
Consumer Systems Group
Austin, Texas

Introduction

More and more applications are requiring liquid crystal displays (LCD) to effectively communicate to the outside world. This application note describes the hardware and software interface needed to display information from the MC68HC705C8A.

Some LCD suppliers provide only the LCD glass so that the waveforms needed to directly drive the LCD segments have to be generated by the microcontroller (MCU) or microprocessor (MPU). Other LCD suppliers provide an LCD module, which has all LCD glass and segment drivers provided in one small packaged circuit board.

This application note uses an LCD module from Optrex, part number DMC16207 (207). It utilizes a Hitachi LCD driver, HD44780, to provide the LCD segment waveforms and a simple parallel port interface that easily interfaces to an MCU or MPU bus.

Circuitry and example code are given to also demonstrate the ability of providing pre-defined messages from EPROM memory. The code can be easily modified to take serial peripheral interface (SPI) and serial communication interface (SCI) data and display it on the LCD module.

LCD Module Hardware Interface

Optrex has many LCD module configurations that have varying display lines and display line character lengths. The 207 module has a 2-line, 16-character/line display. Each character is displayed using a 5 x 7 pixel font matrix. The 207 module has a character generator ROM capable of displaying ASCII characters.

The parallel interface bus can work with either 4-bit or 8-bit buses. Once data is presented on the bus, it is latched by clocking the E pin on the device. Depending on the RS pin, the data will be used as an instruction or an ASCII character.

Pin Descriptions

Table 1 describes the interface pins found on the 207 module.

Table 1. 207 Module Pinout

Pin no.	Signal	I/O	Function
1	V_{SS}	Power	GND (ground)
2	V_{CC}	Power	2.7 V to 5.5 V
3	V_{EE}	Power	LCD drive voltage
4	RS	I	Selects registers 0: Instruction register (for write), address counter (for read) 1: Data register (for write and read)
5	$R/\overline{W}$	I	Selects read or write 0: Write 1: Read
6	E	I	Starts data read/write on falling edge
14–11	DB7–DB4	I/O	Four high-order bidirectional three-state data bus pins. Used for data transfer and receive between the MCU and the 207. DB7 can be used as a busy flag.
10–7	DB3–DB0	I/O	Four low-order bidirectional three-state data bus pins. Used for data transfer and receive between the MCU and the 207. These pins are not used during 4-bit operation.

Bus Timing

Table 2. Bus Timing Electricals

Characteristic	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{CYCE}	500	—	—	ns
Enable pulse width (high level)	PW_{EH}	230	—	—	ns
Enable rise and decay time	t_{Er}, t_{Ef}	—	—	20	ns
Address setup time, RS, R/$\overline{W}$, E	t_{AS}	40	—	—	ns
Data delay time	t_{DDR}	—	—	160	ns
Data setup time	t_{DSW}	80	—	—	ns
Data hold time (write)	t_H	10	—	—	ns
Data hold time (read)	t_{DHR}	5	—	—	ns
Address hold time	t_{AH}	10	—	—	ns

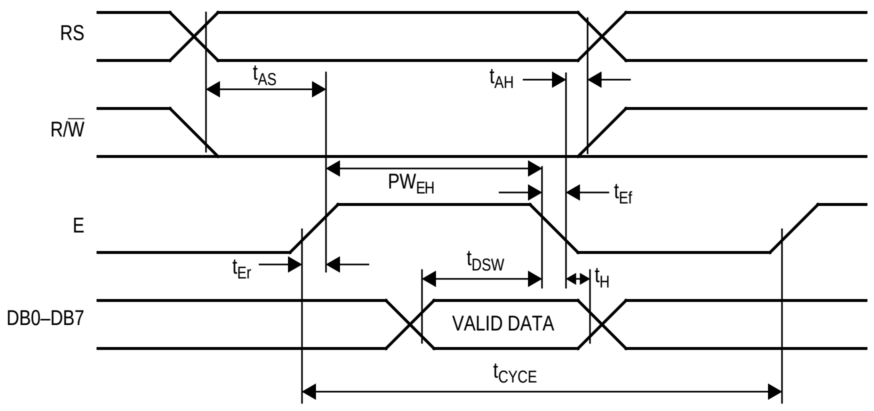


Figure 1. Write Timing Operation

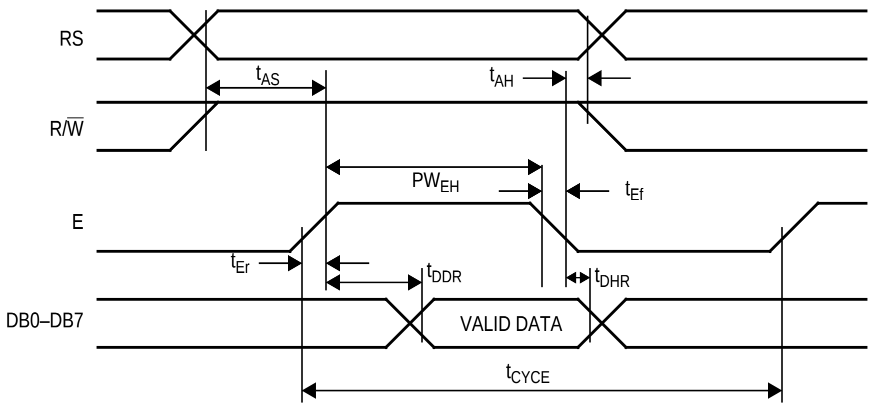


Figure 2. Read Timing Operation

Bus Interface

Figure 3 and Figure 4 show examples of 8-bit and 4-bit timing sequences, respectively. Note that a BF check is not needed if the maximum instruction execution time is respected before sending another instruction.

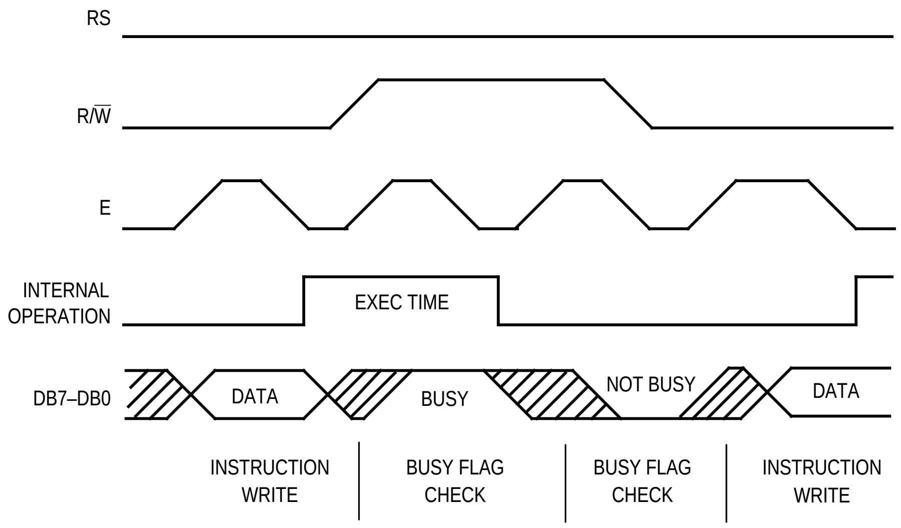


Figure 3. 8-Bit Bus Timing Sequence

For 4-bit interface data, only four bus lines (DB7–DB4) are used for transfer. Bus lines DB3–DB0 are disabled. The data transfer is completed after the 4-bit data has been transferred twice. The four high-order bits are transferred first (DB7–DB4), and then the low-order bits are transferred (DB3–DB0).

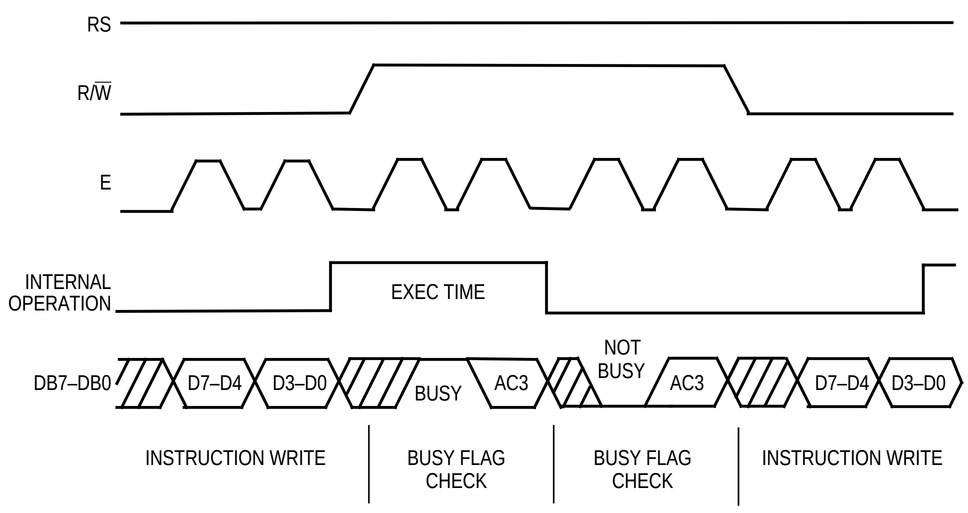


Figure 4. 4-Bit Bus Timing Sequence

LCD Module Software Interface

LCD Instruction Commands

The 207 module has many different configurations that can be easily implemented by sending the correct function command to the device. These commands are listed in **Table 3** followed by an explanation of each function they execute.

Table 3. 207 Module Instruction Code

Instruction	RS	$R\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Execution time (max)
Clear display	0	0	0	0	0	0	0	0	0	1	1.64 ms
Return cursor home	0	0	0	0	0	0	0	0	1	x	1.64 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40 µs
Display on/off ctrl	0	0	0	0	0	0	1	D	C	B	40 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	x	x	40 µs
Function set	0	0	0	0	1	DL	N	F	x	x	40 µs
Set CGRAM address	0	0	0	1	A_{CG}	A_{CG}	A_{CG}	A_{CG}	A_{CG}	A_{CG}	40 µs
Set DDRAM address	0	0	1	A_{DD}	A_{DD}	A_{DD}	A_{DD}	A_{DD}	A_{DD}	A_{DD}	40 µs
Read busy flag &addr	0	1	BF	A_C	A_C	A_C	A_C	A_C	A_C	A_C	0 µs
Write data to CG or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	40 µs
Read data from CG or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	40 µs

DDRAM: Display data RAM
CGRAM: Character generator RAM
A_{CG}: CGRAM address
A_{DD}: DDRAM address; corresponds to cursor address
A_C: Address counter used for both DDRAM and CGRAM addresses

Clear Display

Clear display writes space code $20 into all DDRAM addresses. It then sets DDRAM address 0 into the address counter and returns the display to its original status if it was shifted. In other words, the display disappears and the cursor or blinking goes to the left edge of the first line of the display. I/D of entry mode is set to 1 (increment mode). S of entry mode is left unchanged.

Return Cursor Home

Return cursor home sets the DDRAM address 0 into the address counter and returns the display to its original status if it was shifted. The DDRAM contents do not change.

The cursor or blinking goes to the left edge of the first line of the display.

Entry Mode Set

I/D — Increments (I/D = 1) or decrements (I/D = 0) the DDRAM address by 1 when a character code is written into or read from DDRAM. The cursor or blinking moves to the right when incremented by 1 and to the left when decremented by 1. The same applies to writing and reading of CGRAM.

S — Shifts the entire display either to the right (ID = 0) or to the left (I/D = 1) when S is 1. The display does not shift if S is 0. If S is 1, it will seem as if the cursor does not move but the display does. The display does not shift when reading from DDRAM. Also, writing into or reading out from CGRAM does not shift the display.

Display On/Off Control

D — The display is on when D = 1 and is off when D = 0. When off, the display data remains in DDRAM, but can be displayed instantly by setting D = 1.

C — The cursor is displayed when C = 1 and not displayed when C = 0. Even if the cursor disappears, the function of I/D or other specifications will not change during display data write. The cursor is displayed using five dots in the eighth line of the 5 x 8 dot character.

B — The character indicated by the cursor blinks when B = 1. The blinking is displayed as switching between all blank dots and displayed characters at a speed of 409.6-ms intervals when f_{OSC} (HD44780 operating frequency) is 250 kHz. The cursor and blinking can be set to display simultaneously. (The blinking frequency changes according to f_{OSC}. For example, when f_{OSC}, is 270 kHz, 409.6 x (250/270) = 379.2 ms.)

Cursor or Display Shift

Cursor or display shift shifts the cursor position or display to the right or left without writing or reading display data. See Table 4.This function is used to correct or search the display. In a 2-line display, the cursor moves to the second line when it passes the 40th digit of the first line. The first and second line displays will shift at the same time.

When the displayed data is shifted repeatedly, each line moves only horizontally. The second line display does not shift into the first line position.

The address counter (A_C) contents will not change if the only action performed is a display shift.

Table 4. Cursor and Display Shift Combination

S/C	R/L	Description
0	0	Shifts the cursor position to the left; A_C is decremented by 1
0	1	Shifts the cursor position to the right; A_C is incremented by 1
1	0	Shifts the entire display to the left; he cursor follows the display shift
1	1	Shifts the entire display to the right; he cursor follows the display shift

Function Set

DL — Sets the interface data length. Data is sent or received in 8-bit lengths (DB7 to DB0) when DL = 1, and in 4-bit lengths (DB7 to DB4) when DL = 0. When 4-bit length is selected, data must be sent or received twice.

N — Sets the number of display lines

F — Sets the character font

NOTE: *Perform the function set instruction at the beginning of the program before executing any instructions (except for the read busy flag and address instruction). From this point, the function set instruction cannot be executed unless the interface data length is changed.*

Set CGRAM Address

Set CGRAM address sets the CGRAM binary address $A_{CG}5$–$A_{CG}0$ into the address counter. Data is written to or read from the MCU for CGRAM.

Set DDRAM Address

Set DDRAM address sets the DDRAM binary address $A_{DD}6–A_{DD}0$ into the address counter. Data is written to or read from the MCU for DDRAM.

Read Busy Flag and Address

Read busy flag and address reads the busy flag (BF) indicating that the system is now internally operating on a previously received instruction. If BF = 1, the internal operation is in progress. The next instruction will not be accepted until BF is reset to 0. Check the BF status before the next write operation. At the same time, the value of the address counter in binary ($A_C6–A_C0$) is read out. This address counter is used by both CGRAM and DDRAM addresses, and its value is determined by the previous instruction. The address contents are the same as for instructions set CGRAM address and set DDRAM address.

Write Data to CGRAM or DDRAM

Write data to CGRAM or DDRAM writes 8-bit data to CGRAM or DDRAM. To write into CGRAM or DDRAM is determined by the previous specification of the CGRAM or DDRAM address setting. After a write, the address is incremented or decremented automatically by 1 according to the entry mode. The entry mode also determines the display shift.

Read Data from CGRAM or DDRAM

Read data from CGRAM or DDRAM reads 8-bit data from CGRAM or DDRAM. The previous designation determines whether CGRAM or DDRAM is to be read. Before entering this read instruction, either CGRAM or DDRAM address set instruction must be executed. If not executed, the first read data will be invalid. When serially executing read instructions, the next address data is normally read from the second read. The address set instructions need not be executed just before this read instruction when shifting the cursor by the cursor shift instruction (when reading out of DDRAM). The operation of the cursor shift instruction is the same as the set DDRAM address instruction. After a read, the entry mode automatically increases or decreases the address by 1. However, the display shift is not executed regardless of the entry mode.

Address Map

Table 5 shows the address map for the HD44780. The character positions of the LCD module are shown in the first row of the table with the addresses shown beneath them. The 207 uses only the first 16 addresses.

NOTE: *Note that the addresses are 7 bits wide and when writing to the DDRAM, the MSB (bit 7) is always a 1. Therefore, to write to address $02, the 8-bit data sent to the 207 will be $82 or binary 10000010%.*

Understand that when the display is shifted, the whole address map is used. In other words, when a shift right is executed the character at address $27 is moved to position 1 of the first line of the display.

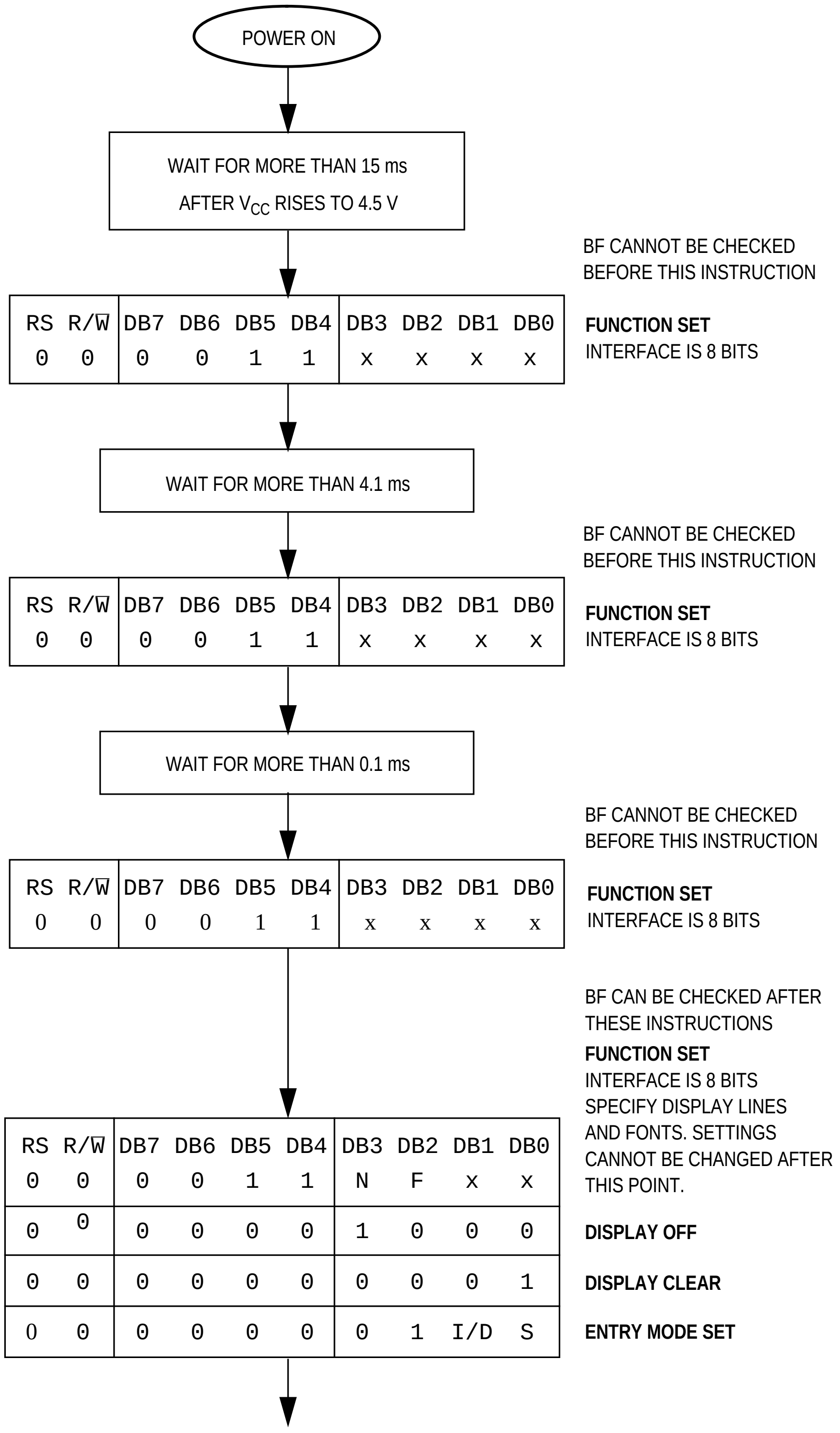
Table 5. LCD Address Map

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	...	Bit 16	...	Bit 39	Bit 40
$00	$01	$02	$03	$04	...	$0F	...	$26	$27
$40	$41	$42	$43	$44	...	$4F	...	$66	$67

Initialization Routines

To ensure proper initialization of the 207 module, a sequence of instruction codes must be executed. These instructions set the data bus width, font type, and number of display lines. In addition, the LCD is cleared, and the entry mode for data is set.

Figure 5 shows the power-on reset initialization for an 8-bit data bus, while Figure 6 shows the power-on reset initialization for a 4-bit data bus.


POWER ON
WAIT FOR MORE THAN 15 ms AFTER V_{CC} RISES TO 4.5 V
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
FUNCTION SET
INTERFACE IS 8 BITS
WAIT FOR MORE THAN 4.1 ms
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
FUNCTION SET
INTERFACE IS 8 BITS
WAIT FOR MORE THAN 0.1 ms
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
FUNCTION SET
INTERFACE IS 8 BITS
BF CAN BE CHECKED AFTER THESE INSTRUCTIONS

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	x	x	x	x	**FUNCTION SET** INTERFACE IS 8 BITS

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	x	x	x	x	**FUNCTION SET** INTERFACE IS 8 BITS

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	x	x	x	x	**FUNCTION SET** INTERFACE IS 8 BITS

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	N	F	x	x	**FUNCTION SET** INTERFACE IS 8 BITS SPECIFY DISPLAY LINES AND FONTS. SETTINGS CANNOT BE CHANGED AFTER THIS POINT.
0	0	0	0	0	0	1	0	0	0	**DISPLAY OFF**
0	0	0	0	0	0	0	0	0	1	**DISPLAY CLEAR**
0	0	0	0	0	0	0	1	I/D	S	**ENTRY MODE SET**

Figure 5. Power-On Reset 8-Bit Initialization

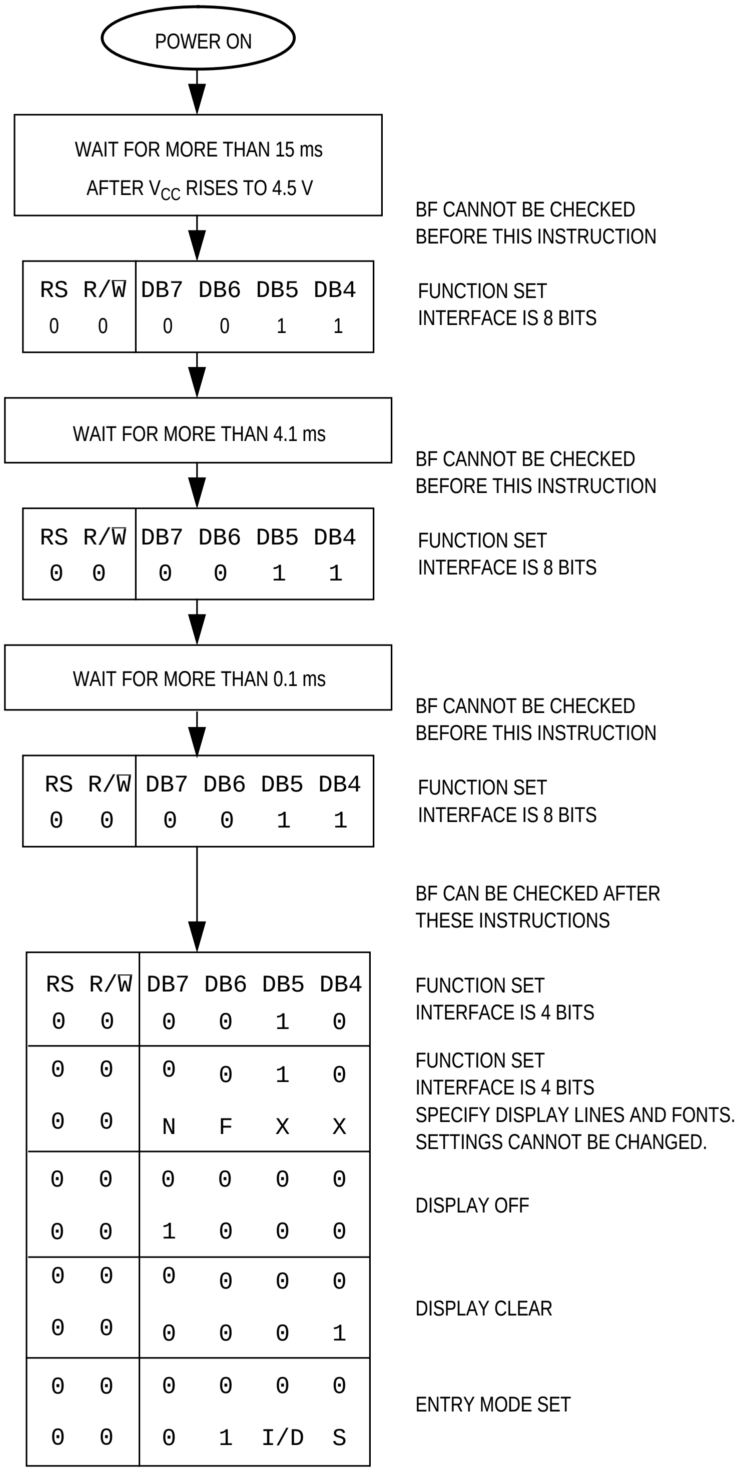

POWER ON
WAIT FOR MORE THAN 15 ms AFTER V_{CC} RISES TO 4.5 V
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
RS R/W̄ DB7 DB6 DB5 DB4
0 0 0 0 1 1
FUNCTION SET INTERFACE IS 8 BITS
WAIT FOR MORE THAN 4.1 ms
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
RS R/W̄ DB7 DB6 DB5 DB4
0 0 0 0 1 1
FUNCTION SET INTERFACE IS 8 BITS
WAIT FOR MORE THAN 0.1 ms
BF CANNOT BE CHECKED BEFORE THIS INSTRUCTION
RS R/W̄ DB7 DB6 DB5 DB4
0 0 0 0 1 1
FUNCTION SET INTERFACE IS 8 BITS
BF CAN BE CHECKED AFTER THESE INSTRUCTIONS
RS R/W̄ DB7 DB6 DB5 DB4
0 0 0 0 1 0
FUNCTION SET INTERFACE IS 4 BITS
0 0 0 0 1 0
0 0 N F X X
FUNCTION SET INTERFACE IS 4 BITS SPECIFY DISPLAY LINES AND FONTS. SETTINGS CANNOT BE CHANGED.
0 0 0 0 0 0
0 0 1 0 0 0
DISPLAY OFF
0 0 0 0 0 0
0 0 0 0 0 1
DISPLAY CLEAR
0 0 0 0 0 0
0 0 0 1 I/D S
ENTRY MODE SET

Figure 6. Power-On Reset 4-Bit Initialization

MC68HC705C8A Interface

Choosing between an 8- and a 4-bit data bus is usually defined by the I/O (input/output) and code space constraints of the application. To analyze both, two different test routines were written to demonstrate the 8-bit and 4-bit bus configurations. Also, the R/W pin of the 207 was grounded for write executions only. Since we cannot check the BF flag, the delay times stated in **Table 3** must be observed.

Although these routines were tested on an MC68HC705C8A device, any HC05 device with enough memory and I/O can execute these routines. A simple change in the memory map should allow the code to be ported to other HC05s.

Hardware

The code was tested on these development tools:

- M68MMPFB0508 — MMEVS platform board
- X68EM05C9A — C/D series emulation module
- M68CBL05B — Low noise flex cable
- M68TB05C9P40 — 40-pin PLCC target head adapter

The schematic shown in **Figure 7** shows a typical circuit used to interface the MC68HC705C8A to the 207.

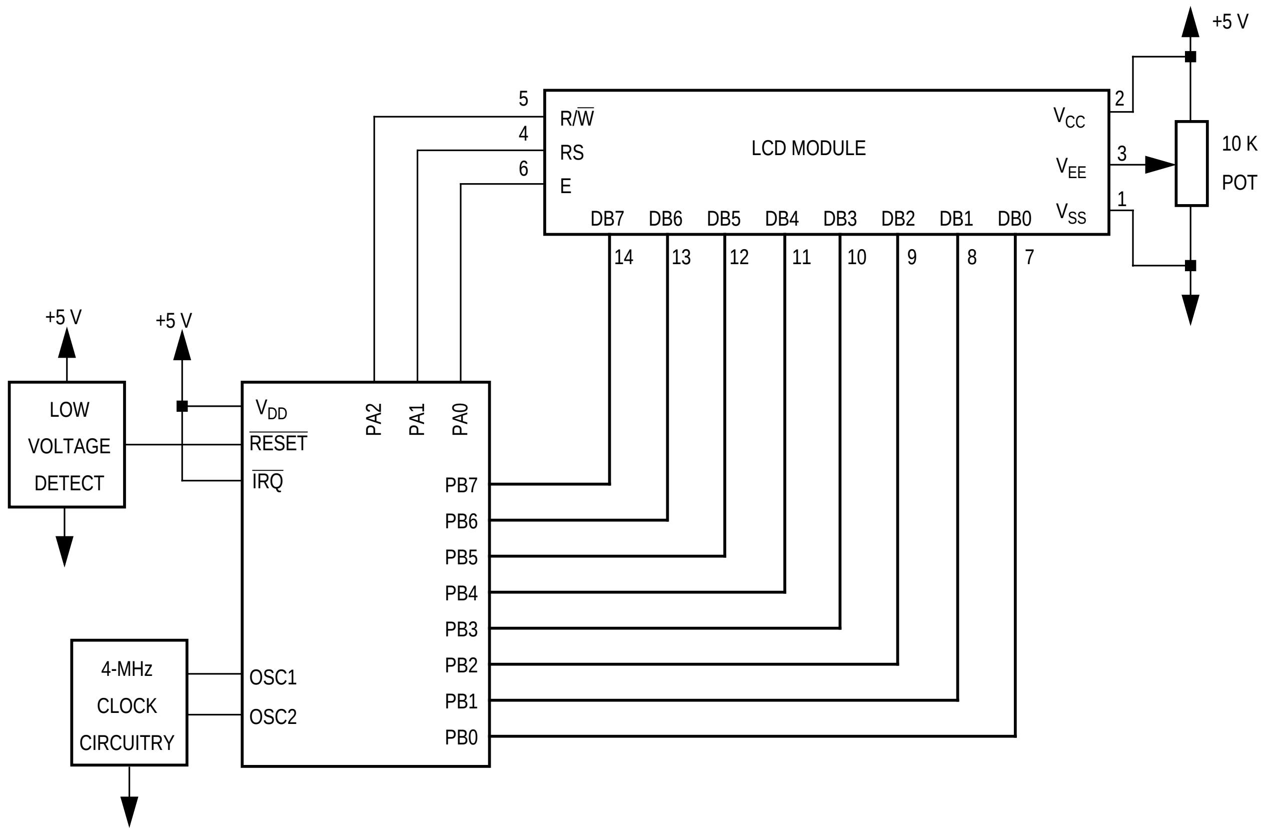


Figure 7. Typical C8A-to-207 Circuit

Software

The software written to demonstrate the MC68HC705C8A to LCD module interface is shown in the following appendices.

- **Appendix A — Flowcharts**
- **Appendix B — 8-Bit Bus Code**
- **Appendix C — 4-Bit Bus Code**

The flowchart roughly sketches out the routines. The code was written to take pre-defined messages in ROM and easily display them by calling a subroutine. If the MC68HC705C8A is receiving messages from the SPI or SCI, put the message in a temporary RAM buffer and change the message routine to start reading ASCII characters from the start of the buffer.

References

MC68HC705C8A Technical Data, Freescale order number MC68HC705C8A/D, Freescale, 1996.

M68HC05 Applications Guide, Freescale order number M68HC05AG/AD/D, Freescale, 1996.

DMC-16207 Digikey #73-1025-ND.

1997 Optrex LCD Databook Digikey #73-1001-ND.

Appendix A — Flowcharts

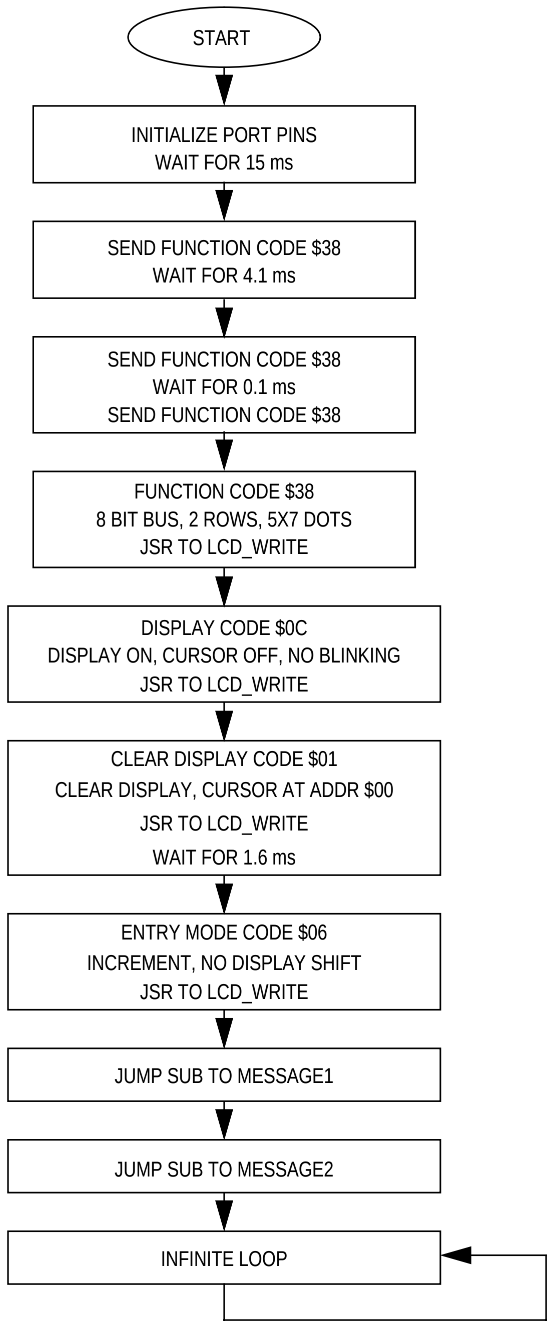


Figure 8. Main Flowchart

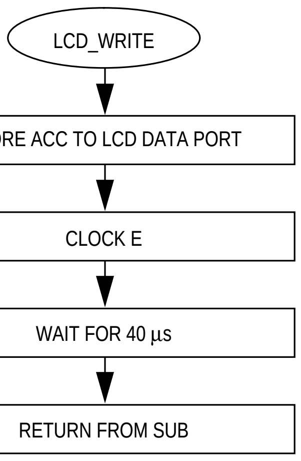


Figure 9. LCD_Write Subroutine Flowchart

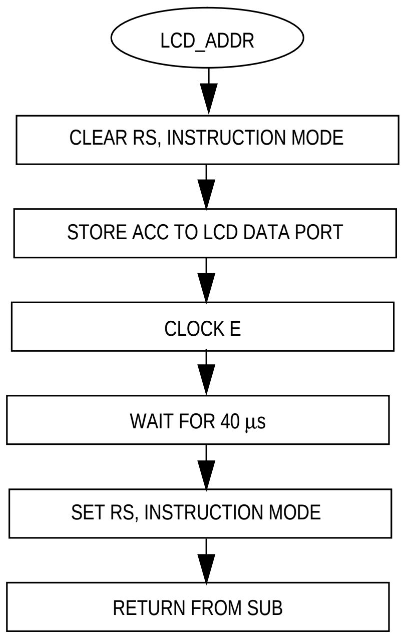


Figure 10. LCD_ADDR Subroutine Flowchart

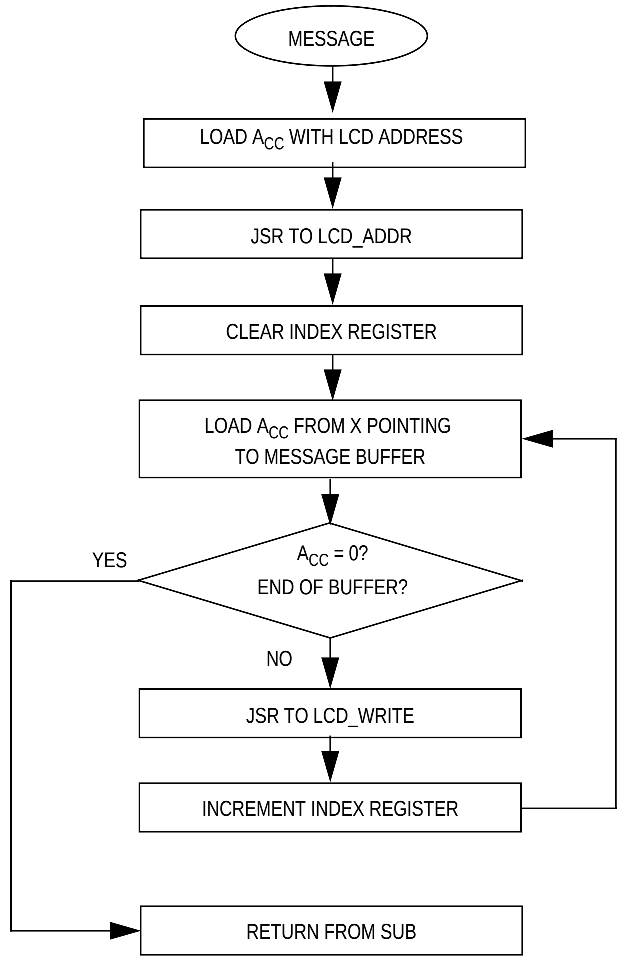

MESSAGE
LOAD A_{CC} WITH LCD ADDRESS
JSR TO LCD_ADDR
CLEAR INDEX REGISTER
LOAD A_{CC} FROM X POINTING TO MESSAGE BUFFER
A_{CC} = 0? END OF BUFFER?
YES
NO
JSR TO LCD_WRITE
INCREMENT INDEX REGISTER
RETURN FROM SUB

Figure 11. Message Subroutine Flowchart

Appendix B — 8-Bit Bus Code

```
*********************************************************************************
**
*
* File name: LCD_MOD8.ASM
* Example Code for LCD Module (HD44780) using 8-bit bus
* Ver: 1.0
* Date: April 10, 1998
* Author: Mark Glenewinkel
*         Field Applications
*         Consumer Systems Group
* Assembler: P&E IDE ver 1.02
*
* For code explanation and flowcharts, please consult Freescale Application Note
*    "Interfacing the HC705C8A to an LCD Module" Literature # AN1745/D
*
**********************************************************************************

*** SYSTEM DEFINITIONS AND EQUATES ***********************************************
*** Internal Register Definitions
PORTA          EQU     $00                      ;LCD control signals
PORTB          EQU     $01                      ;LCD data bus
DDRA           EQU     $04                      ;data direction for PortA
DDRB           EQU     $05                      ;data direction for PortB

*** Application Specific Definitions
LCD_CTRL       EQU     $00                      ;PORTA
LCD_DATA       EQU     $01                      ;PORTB
E              EQU     0T                       ;PORTA, bit 0
RW             EQU     2T                       ;PORTA, bit 2
RS             EQU     1T                       ;PORTA, bit 1

*** Memory Definitions
EPROM          EQU     $160                     ;start of EPROM mem
RAM            EQU     $50                      ;start of RAM mem
MSG_STORAGE    EQU     $500                     ;start of message block
RESET          EQU     $1FFE                    ;vector for reset


*** RAM VARIABLES
**********************************************************************************
               ORG     RAM
TIME           DB      1                        ;used for delay time
```

```
*** MAIN ROUTINE ******************************************************************
              ORG     EPROM                 ;start at beg of EPROM
*** Intialize Ports
START         clr     LCD_CTRL              ;clear LCD_CTRL
              clr     LCD_DATA              ;clear LCD_DATA
              lda     #$FF                  ;make ports outputs
              sta     DDRA                  ;PortA output
              sta     DDRB                  ;PortB output

*** INITIALIZE THE LCD
*** Wait for 15 ms
              lda     #150T
              sta     TIME                  ;set delay time
              jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
              lda     #$38                  ;LCD init command
              sta     LCD_DATA
              bset    E,LCD_CTRL            ;clock in data
              bclr    E,LCD_CTRL

*** Wait for 4.1 ms
              lda     #41T
              sta     TIME                  ;set delay time
              jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
              lda     #$38                  ;LCD init command
              sta     LCD_DATA
              bset    E,LCD_CTRL            ;clock in data
              bclr    E,LCD_CTRL

*** Wait for 100 µs
              lda     #1T
              sta     TIME                  ;set delay time
              jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
              lda     #$38                  ;LCD init command
              jsr     LCD_WRITE             ;write data to LCD

*** Send Function Set Command
*** 8-bit bus, 2 rows, 5x7 dots
              lda     #$38                  ;function set command
              jsr     LCD_WRITE             ;write data to LCD

*** Send Display Ctrl Command
*** display on, cursor off, no blinking
              lda     #$0C                  ;display ctrl command
              jsr     LCD_WRITE             ;write data to LCD
```

```
*** Send Clear Display Command
*** clear display, cursor addr=0
              lda     #$01                ;clear display command
              jsr     LCD_WRITE           ;write data to LCD
              lda     #16T
              sta     TIME                ;set delay time for 1.6 ms
              jsr     VAR_DELAY           ;sub for 0.1ms delay

*** Send Entry Mode Command
*** increment, no display shift
              lda     #$06                ;entry mode command
              jsr     LCD_WRITE           ;write data to LCD

*** SEND MESSAGES
*** Set the address, send data

              jsr     MESSAGE1            ;send Message1
              jsr     MESSAGE2            ;send Message2

DUMMY         bra     DUMMY               ;done with example


*** SUBROUTINES ***************************************************************
*** Routine creates a delay according to the formula
*** TIME*100 µs using a 2-MHz internal bus
*** Cycle count per instruction shown
VAR_DELAY     lda     #33T                ;2
L1            deca                        ;3
              bne     L1                  ;3
              dec     TIME                ;5
              bne     VAR_DELAY           ;3
              rts                         ;6

*** Routine sends LCD Data
LCD_WRITE     sta     LCD_DATA
              bset    E,LCD_CTRL          ;clock in data
              bclr    E,LCD_CTRL
              lda     #13T                ;2 40 µs delay for LCD
L2            deca                        ;3
              bne     L2                  ;3
              rts

*** Routine sends LCD Address
LCD_ADDR      bclr    RS,LCD_CTRL         ;LCD in command mode
              sta     LCD_DATA
              bset    E,LCD_CTRL          ;clock in data
              bclr    E,LCD_CTRL
              lda     #13T                ;2 40 µs delay
L4            deca                        ;3
              bne     L4                  ;3
              bset    RS,LCD_CTRL         ;LCD in data mode
              rts
```

```
MESSAGE1        lda    #$84                 ;addr = $04
               jsr    LCD_ADDR             ;send addr to LCD
                clrx
L3              lda    MSG1,X               ;load AccA w/char from msg
                beq    OUTMSG1              ;end of msg?
                jsr    LCD_WRITE            ;write data to LCD
                incx
                bra    L3                   ;loop to finish msg
OUTMSG1         rts

MESSAGE2        lda    #$C0                 ;addr = $40
                jsr    LCD_ADDR             ;send addr to LCD
                clrx
L5              lda    MSG2,X               ;load AccA w/char from msg
                beq    OUTMSG2              ;end of msg?
                jsr    LCD_WRITE            ;write data to LCD
                incx
                bra    L5                   ;loop to finish msg
OUTMSG2         rts


*** MESSAGE STORAGE ****************************************************************
              ORG      MSG_STORAGE
MSG1          db       'Freescale'
              db       0
MSG2          db       'Microcontrollers'
              db       0


*** VECTOR TABLE *******************************************************************
              ORG      RESET
              DW       START
```

Appendix C — 4-Bit Bus Code

```
*****************************************************************************
*
* File name: LCD_MOD4.ASM
* Example Code for LCD Module (HD44780) using 4-bit bus
* Ver: 1.0
* Date: April 10, 1998
* Author: Mark Glenewinkel
*         Freescale Field Applications
*         Consumer Systems Group
* Assembler: P&E IDE ver 1.02
*
* For code explanation and flowcharts, please consult Freescale Application Note
*    "Interfacing the HC705C8A to an LCD Module" Literature # AN1745/D
*
*****************************************************************************

*** SYSTEM DEFINITIONS AND EQUATES *********************************************
*** Internal Register Definitions
PORTA          EQU      $00                    ;LCD control signals
PORTB          EQU      $01                    ;LCD data bus
DDRA           EQU      $04                    ;data direction for PortA
DDRB           EQU      $05                    ;data direction for PortB

*** Application Specific Definitions
LCD_CTRL       EQU      $00                    ;PORTA
LCD_DATA       EQU      $01                    ;PORTB
E              EQU      0T                     ;PORTA, bit 0
RW             EQU      2T                     ;PORTA, bit 2
RS             EQU      1T                     ;PORTA, bit 1

*** Memory Definitions
EPROM          EQU      $160                   ;start of EPROM mem
RAM            EQU      $50                    ;start of RAM mem
MSG_STORAGE    EQU      $500                   ;start of message block
RESET          EQU      $1FFE                  ;vector for reset


*** RAM VARIABLES ***************************************************************
               ORG      RAM
TIME           DB       1                      ;used for delay time
```

```
*** MAIN ROUTINE ******************************************************************
               ORG     EPROM                 ;start at beg of EPROM
*** Intialize Ports
START          clr     LCD_CTRL              ;clear LCD_CTRL
               clr     LCD_DATA              ;clear LCD_DATA
               lda     #$FF                  ;make ports outputs
               sta     DDRA                  ;PortA output
               sta     DDRB                  ;PortB output

*** INITIALIZE THE LCD
*** Wait for 15 ms
               lda     #150T
               sta     TIME                  ;set delay time
               jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
               lda     #$30                  ;LCD init command
               sta     LCD_DATA
               bset    E,LCD_CTRL            ;clock in data
               bclr    E,LCD_CTRL

*** Wait for 4.1 ms
               lda     #41T
               sta     TIME                  ;set delay time
               jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
               lda     #$30                  ;LCD init command
               sta     LCD_DATA
               bset    E,LCD_CTRL            ;clock in data
               bclr    E,LCD_CTRL

*** Wait for 100 µs
               lda     #1T
               sta     TIME                  ;set delay time
               jsr     VAR_DELAY             ;sub for 0.1ms delay

*** Send Init Command
               lda     #$30                  ;LCD init command
               jsr     LCD_WRITE             ;write data to LCD

*** Send Function Set Command
*** 4-bit bus, 2 rows, 5x7 dots
               lda     #$20                  ;function set command
               jsr     LCD_WRITE             ;write data to LCD
               lda     #$20                  ;function set command
               jsr     LCD_WRITE             ;write data to LCD
               lda     #$80                  ;function set command
               jsr     LCD_WRITE             ;write data to LCD
```

```
*** Send Display Ctrl Command
*** display on, cursor off, no blinking
               lda      #$00                  ;display ctrl command MSB
               jsr      LCD_WRITE             ;write data to LCD
               lda      #$C0                  ;display ctrl command LSB
               jsr      LCD_WRITE             ;write data to LCD

*** Send Clear Display Command
*** clear display, cursor addr=0
               lda      #$00                  ;clear display command MSB
               jsr      LCD_WRITE             ;write data to LCD
               lda      #16T
               sta      TIME
               jsr      VAR_DELAY             ;delay for 1.6 ms
               lda      #$10                  ;clear display command LSB
               jsr      LCD_WRITE             ;write data to LCD
               lda      #16T
               sta      TIME
               jsr      VAR_DELAY             ;delay for 1.6 ms

*** Send Entry Mode Command
*** increment, no display shift
               lda      #$00                  ;entry mode command MSB
               jsr      LCD_WRITE             ;write data to LCD
               lda      #$60                  ;entry mode command LSB
               jsr      LCD_WRITE             ;write data to LCD

*** SEND MESSAGES
*** Set the address, send data

               jsr      MESSAGE1              ;send Message1
               jsr      MESSAGE2              ;send Message2


DUMMY          bra      DUMMY                 ;done with example


*** SUBROUTINES ********************************************************************
*** Routine creates a delay according to the formula
*** TIME*100 µs using a 2-MHz internal bus
*** Cycle count per instruction shown
VAR_DELAY      lda      #33T                  ;2
L1             deca                           ;3
               bne      L1                    ;3
               dec      TIME                  ;5
               bne      VAR_DELAY             ;3
               rts                            ;6
```

```
*** Routine sends LCD Data
LCD_WRITE       sta      LCD_DATA
                bset     E,LCD_CTRL            ;clock in data
                bclr     E,LCD_CTRL
                lda      #13T                  ;2 40 µs delay for LCD
L2              deca                           ;3
                bne      L2                    ;3
                rts

*** Routine sends LCD Address
LCD_ADDR        bclr     RS,LCD_CTRL           ;LCD in command mode
                sta      LCD_DATA
                bset     E,LCD_CTRL            ;clock in data
                bclr     E,LCD_CTRL
                lda      #13T                  ;2 40 µs delay
L4              deca                           ;3
                bne      L4                    ;3
                bset     RS,LCD_CTRL           ;LCD in data mode
                rts

MESSAGE1        lda      #$80                  ;addr = $04 MSB
                jsr      LCD_ADDR              ;send addr to LCD
                lda      #$40                  ;addr = $04 LSB
                jsr      LCD_ADDR              ;send addr to LCD
                clrx
L3              lda      MSG1,X                ;load AccA w/char from msg
                beq      OUTMSG1               ;end of msg?
                jsr      LCD_WRITE             ;write data to LCD
                lda      MSG1,X                ;load Acca w/char from msg
                asla                           ;shift LSB to MSB
                asla
                asla
                asla
                jsr      LCD_WRITE             ;write data to LCD
                incx
                bra      L3                    ;loop to finish msg
OUTMSG1         rts

MESSAGE2        lda      #$C0                  ;addr = $40 MSB
                jsr      LCD_ADDR              ;send addr to LCD
                lda      #$00                  ;addr = $40 LSB
                jsr      LCD_ADDR              ;send addr to LCD
                clrx
L5              lda      MSG2,X                ;load Acca w/char from msg
                beq      OUTMSG2               ;end of msg?
                jsr      LCD_WRITE             ;write data to LCD
                lda      MSG2,X                ;load AccA w/char from msg
                asla                           ;shift LSB to MSB
                asla
                asla
asla
```

```
               jsr      LCD_WRITE              ;write data to LCD
               incx
               bra      L5                     ;loop to finish msg
OUTMSG2        rts


*** MESSAGE STORAGE ******************************************************************
               ORG      MSG_STORAGE
MSG1           db       'Freescale'
               db       0
MSG2           db       'Microcontrollers'
               db       0


*** VECTOR TABLE *********************************************************************
               ORG      RESET
               DW       START
```

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

AN1745/D