Interfacing the MC68HC912B32 to an LCD Module

By Mark Glenewinkel
Field Applications Engineering
Austin, Texas

Introduction

More and more applications are requiring liquid crystal displays (LCD) to communicate effectively to the outside world. This application note describes the hardware and software interface needed to display information from the MC68HC912B32 (B32).

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 Corporation, 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 memory to the display. The code can be modified easily to take serial peripheral interface (SPI) and serial communication interface (SCI) data and display it on the LCD module.
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 per 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

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Signal</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$V_{SS}$</td>
<td>Power</td>
<td>GND (ground)</td>
</tr>
<tr>
<td>2</td>
<td>$V_{CC}$</td>
<td>Power</td>
<td>4.5 volts to 5.5 volts</td>
</tr>
<tr>
<td>3</td>
<td>$V_{EE}$</td>
<td>Power</td>
<td>LCD drive voltage</td>
</tr>
</tbody>
</table>
| 4          | RS     | I    | Selects registers  
|            |        |      | 0: Instruction register (for write), address counter (for read)  
|            |        |      | 1: Data register (for write and read) |
| 5          | R/\bar{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 3-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, 3-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

<table>
<thead>
<tr>
<th>Spec</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable cycle time</td>
<td>t_{CYCLE}</td>
<td>500</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Enable pulse width (high level)</td>
<td>PW_{EH}</td>
<td>230</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Enable rise and decay time</td>
<td>t_{ER}, t_{EF}</td>
<td>—</td>
<td>—</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>Address setup time, RS, R/W, E</td>
<td>t_{AS}</td>
<td>40</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Data delay time</td>
<td>t_{DDR}</td>
<td>—</td>
<td>—</td>
<td>160</td>
<td>ns</td>
</tr>
<tr>
<td>Data setup time</td>
<td>t_{DSW}</td>
<td>80</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Data hold time (write)</td>
<td>t_{H}</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Data hold time (read)</td>
<td>t_{DHR}</td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>Address hold time</td>
<td>t_{AH}</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>

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:** A BF (busy flag) check is not needed if the maximum instruction execution time is respected before sending another instruction.

*Figure 3. 8-Bit Bus Timing Sequence*

*Figure 4. 4-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).

### LCD Module Software Interface

#### LCD Instruction Commands

The 207 module has many different configurations that can be implemented easily 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**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
<th>Execution Time (max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear display</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1.64 ms</td>
</tr>
<tr>
<td>Return cursor home</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>Return cursor home</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1.64 ms</td>
</tr>
<tr>
<td>Entry mode set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D C B 40 µs</td>
</tr>
<tr>
<td>Display on/off control</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C B 40 µs</td>
</tr>
<tr>
<td>Cursor or display shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>S/C</td>
<td>R/L x x 40 µs</td>
</tr>
<tr>
<td>Function set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>N</td>
<td>F</td>
<td>x</td>
<td>x</td>
<td>40 µs</td>
</tr>
<tr>
<td>Set CGRAM address</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>40 µs</td>
</tr>
<tr>
<td>Set DDRAM address</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>40 µs</td>
</tr>
<tr>
<td>Read busy flag and address</td>
<td>0</td>
<td>1</td>
<td>BF</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>0 µs</td>
</tr>
<tr>
<td>Write data to CG or DDRAM</td>
<td>1</td>
<td>0</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
<td>40 µs</td>
</tr>
<tr>
<td>Write data from CG or DDRAM</td>
<td>1</td>
<td>1</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
<td>40 µs</td>
</tr>
</tbody>
</table>

DDRAM: Display data RAM
CGRAM: Character generator RAM
AC, ADD: Correspond to CGRAM address
AC: 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 (I/D = 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 it 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

<table>
<thead>
<tr>
<th>S/C</th>
<th>R/L</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Shifts the cursor position to the left; A_C is decremented by 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Shifts the cursor position to the right; A_C is incremented by 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Shifts the entire display to the left; the cursor follows the display shift</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Shifts the entire display to the right; the cursor follows the display shift</td>
</tr>
</tbody>
</table>

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 \( A_{CG5} - A_{CG0} \) 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 \( A_{DD6} - A_{DD0} \) 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 normally is 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:** The addresses are seven 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>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>...</th>
<th>Bit 16</th>
<th>...</th>
<th>Bit 39</th>
<th>Bit 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00$</td>
<td>$01$</td>
<td>$02$</td>
<td>$03$</td>
<td>$04$</td>
<td>...</td>
<td>$0F$</td>
<td>...</td>
<td>$26$</td>
<td>$27$</td>
</tr>
<tr>
<td>$40$</td>
<td>$41$</td>
<td>$42$</td>
<td>$43$</td>
<td>$44$</td>
<td>...</td>
<td>$4F$</td>
<td>...</td>
<td>$66$</td>
<td>$67$</td>
</tr>
</tbody>
</table>

**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.
Figure 5. Power-On Reset 8-Bit Initialization
Figure 6. Power-On Reset 4-Bit Initialization
MC68HC912B32 Hardware Interface

The B32 is a 16-bit MCU device with standard on-chip peripherals including:

- 32 Kbytes of FLASH EEPROM
- 1 Kbyte of RAM
- 768 bytes of EEPROM
- Asynchronous serial communications interface (SCI)
- Serial peripheral interface (SPI)
- 8-channel, 16-bit timer
- 8-channel, 8-bit analog-to-digital converter (ADC)
- 4-channel pulse-width modulator (PWM)
- J1850-compatible byte data link communications module (BDLC)

The B32 has a maximum of 63 I/O (input/output) pins in single-chip mode. These I/O pins share functionality with the on-chip peripheral modules. Rarely will a system have all of these I/O pins available. The LCD module works in either an 8-bit or 4-bit data bus. The data bus size should be defined from the I/O, peripheral, and code space usage of the application. Three I/O pins are also needed for bus control.

The schematic used for testing the B32-to-207 interface on the MC68HC912B32 evaluation board is shown in Figure 7. The test circuit was designed to use either a 4-bit or 8-bit databus. Although the R/W pin on the 207 is connected to the B32, it may be grounded if only writes to the LCD are executed. Since we cannot check the BF flag, the delay times stated in Table 3 must be observed.

Although these routines were tested on an MC68HC912B32 device, any HC12 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 HC12s.
The software written to demonstrate the MC68HC912B32-to-LCD module interface is shown in sections titled Flowcharts, 4-Bit Bus Code, and 8-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 B32 is receiving messages from the SPI or SCI, put the ASCII data in a temporary RAM buffer and change the message routines to start reading ASCII characters from the start of the buffer.
Development Tools

The interface was created and tested using these development tools:

- M68HC12B32EVB — Motorola’s MC68HC912B32 evaluation board
- WIN IDE— P&E Microcomputer Systems integrated development environment, version 1.02
- CASM12W — P&E Microcomputer Systems HC12 assembler, version 3.08
- ICD12W — P&E Microcomputer Systems HC12 in-circuit debugger, version 1.04 build B

References


DMC-16207 Digikey #73-1025-ND.

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

Motorola’s HC12 website:

START

INITIALIZE PORT PINS
WAIT FOR 15 MS

SEND FUNCTION CODE $38
WAIT FOR 4.1 MS

SEND FUNCTION CODE $38
WAIT FOR 0.1 MS
SEND FUNCTION CODE $38

FUNCTION CODE $38
8-BIT BUS, 2 ROWS, 5 X 7 DOTS
JSR TO LCD_WRITE

DISPLAY CODE $0C
DISPLAY ON, CURSOR OFF, NO BLINKING
JSR TO LCD_WRITE

CLEAR DISPLAY CODE $01
CLEAR DISPLAY, CURSOR AT ADDR $00
JSR TO LCD_WRITE
WAIT FOR 1.6 MS

ENTRY MODE CODE $06
INCREMENT, NO DISPLAY SHIFT
JSR TO LCD_WRITE

JUMP SUB TO MESSAGE1

JUMP SUB TO MESSAGE2

INFINITE LOOP

Figure 8. Main Flowchart
Figure 9. LCD_Write Subroutine Flowchart

LCD_WRITE

STORE ACCA TO LCD DATA PORT

CLOCK E

WAIT FOR 40 µs

RETURN FROM SUB

Figure 10. LCD_ADDR Subroutine Flowchart

LCD_ADDR

CLEAR RS, INSTRUCTION MODE

STORE ACCA TO LCD DATA PORT

CLOCK E

WAIT FOR 40 µs

SET RS, INSTRUCTION MODE

RETURN FROM SUB
Figure 11. Message Subroutine Flowchart
8-Bit Bus Code

************************************************************************************
* File name: H12_LCD8.ASM
* Example Code for LCD Module (DMC16207) using 8-bit bus
* interfacing with the MC68HC912B32
* Ver: 1.0
* Date: September 6, 1998
* Author: Mark Glenewinkel
* Motorola Field Applications
* Assembler: P&E CASM12W ver 3.08
*
*For code explanation and flowcharts, please consult Motorola Application Note
* "Interfacing the MC68HC912B32 to an LCD Module" Literature # AN1774/D
*
* Note: Code originates in RAM instead of FLASH
*
************************************************************************************

*** SYSTEM DEFINITIONS AND EQUATES ***********************************************
*** Internal Register Definitions
PORTA EQU $00 ;LCD data bus
PORTB EQU $01 ;LCD control signals
DDRA EQU $02 ;data direction for PortA
DDRB EQU $03 ;data direction for PortB

*** Application Specific Definitions
LCD_DATA EQU $00 ;PORTA
LCD_CTRL EQU $01 ;PORTB
E EQU 1T ;PORTB, bit 0
RW EQU 4T ;PORTB, bit 2
RS EQU 2T ;PORTB, bit 1

*** Memory Definitions
RAM_START EQU $0800 ;start of RAM mem
RAM_VAR EQU $0BF0 ;start of RAM variables
MSG_STORAGE EQU $0B00 ;start of message block

*** Vectors
RESET EQU $FFE ;vector for reset

*** RAM VARIABLES ***************************************************************

TIME DB 1 ;used for delay time

*** MAIN ROUTINE ***************************************************************

*** Initialize the Stack Pointer
    lds #$0BFF ;init SP, top or RAM
*** Initialize Ports
START clr LCD_CTRL ;clear LCD_CTRL
clr LCD_DATA ;clear LCD_DATA
movb #$FF,DDRA ;PortA output
movb #$FF,DDRB ;PortB output

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

*** Send Init Command
movb #$38,LCD_DATA ;LCD init command
bset LCD_CTRL,E ;clock in data
bc1r LCD_CTRL,E

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

*** Send Init Command
movb #$38,LCD_DATA ;LCD init command
bset LCD_CTRL,E ;clock in data
bc1r LCD_CTRL,E

*** Wait for 100 us
movb #1T,TIME ;set delay time
jsr VAR_DELAY ;sub for 0.1ms delay

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

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

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

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

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

*** SEND MESSAGES
*** Messages have address and content predefined
jsr MESSAGE1 ;send Message1
jsr MESSAGE2 ;send Message2

DUMMY bra DUMMY ;done with example
*** SUBROUTINES ********************************************************************
*** Routine creates a delay according to the formula
*** TIME*~100µs using an 8MHz internal bus
*** Cycle count per instruction shown
VAR_DELAY  ldab #199T ;1
L1    nop ;1
   dbne B,L1 ;3
dec TIME ;4
   bne VAR_DELAY ;3
   rts ;5

*** Routine sends LCD Data
LCD_WRITE staa LCD_DATA
   bset LCD_CTRL,E ;clock in data
   bclr LCD_CTRL,E
   ldaa #107T ;40µs delay for LCD
L2    dbne A,L2 ;3
   rts

*** Routine sends LCD Address
LCD_ADDR  bclr LCD_CTRL,RS ;LCD in command mode
   staa LCD_DATA
   bset LCD_CTRL,E ;clock in data
   bclr LCD_CTRL,E
   ldaa #107T ;40µs delay for LCD
L4    dbne A,L4 ;3
   bset LCD_CTRL,RS ;LCD in data mode
   rts

*** Message Routines
MESSAGE1  ldaa #$84 ;addr = $04
   jsr LCD_ADDR ;send addr to LCD
   ldx #0
L3    ldaa MSG1,X ;load AccA w/char from msg
   beq OUTMSG1 ;end of msg?
   jsr LCD_WRITE ;write data to LCD
   inx ;increment X
   bra L3 ;loop to finish msg
OUTMSG1  rts

MESSAGE2  ldaa #$C4 ;addr = $44
   jsr LCD_ADDR ;send addr to LCD
   ldaa MSG2,X ;load AccA w/char from msg
   beq OUTMSG2 ;end of msg?
   jsr LCD_WRITE ;write data to LCD
   inx ;increment X
   bra L5 ;loop to finish msg
OUTMSG2  rts

*** MESSAGE STORAGE ****************************************************************
ORG MSG_STORAGE
MSG1   db 'Motorola'
   db 0

MSG2   db 'HC12 MCU'
   db 0

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

AN1774
20 MOTOROLA
### 4-Bit Bus Code

****************************
* File name: H12_LCD4.ASM
* Example Code for LCD Module (DMC16207) using 4-bit bus
* interfacing with the MC68HC912B32
* Ver: 1.0
* Date: September 6, 1998
* Author: Mark Glenewinkel
*         Motorola Field Applications
* Assembler: P&E CASM12W ver 3.08
* For code explanation and flow charts, please consult Motorola Application Note
*   “Interfacing the MC68HC912B32 to an LCD Module” Literature # AN1774/D
* Note: Code originates in RAM instead of FLASH
*
************************************************************************************

### System Definitions and Equates

#### Internal Register Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>$00</td>
</tr>
<tr>
<td>PORTB</td>
<td>$01</td>
</tr>
<tr>
<td>DDRA</td>
<td>$02</td>
</tr>
<tr>
<td>DDRB</td>
<td>$03</td>
</tr>
</tbody>
</table>

#### Application Specific Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_DATA</td>
<td>$00</td>
</tr>
<tr>
<td>LCD_CTRL</td>
<td>$01</td>
</tr>
<tr>
<td>E</td>
<td>$1T</td>
</tr>
<tr>
<td>RW</td>
<td>$4T</td>
</tr>
<tr>
<td>RS</td>
<td>$2T</td>
</tr>
</tbody>
</table>

#### Memory Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM_START</td>
<td>$0800</td>
</tr>
<tr>
<td>RAM_VAR</td>
<td>$0BF0</td>
</tr>
<tr>
<td>MSG_STORAGE</td>
<td>$0B00</td>
</tr>
</tbody>
</table>

#### Vectors

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>$FFFF</td>
</tr>
</tbody>
</table>

### RAM Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>DB 1</td>
</tr>
</tbody>
</table>

### Main Routine

#### Initialize the Stack pointer

```asm
lds #$0BFF
```

#### Initialize Ports

```asm
START clr LCD_CTRL
        clr LCD_DATA
        movb #$FF,DDRA
        movb #$FF,DDRb
```

AN1774

MOTOROLA
**Application Note**

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

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

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

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

*** Wait for 100 us
    movb #1T,TIME ;set delay time
    jsr VAR_DELAY ;sub for 0.1ms delay

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

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

*** Send Display Ctrl Command
*** display on, cursor off, no blinking
    ldaa #$00 ;function set command
    jsr LCD_WRITE ;write data to LCD
    ldaa #$C0 ;display ctrl command
    jsr LCD_WRITE ;write data to LCD

*** Send Clear Display Command
*** clear display, cursor addr=0
    ldaa #$00 ;clear display command
    jsr LCD_WRITE ;write data to LCD
    movb #16T,TIME ;set delay time for 1.6ms
    jsr VAR_DELAY ;sub for 0.1ms delay
    ldaa #$10 ;clear display command
    jsr LCD_WRITE ;write data to LCD
    movb #16T,TIME ;set delay time for 1.6ms
    jsr VAR_DELAY ;sub for 0.1ms delay
```
*** Send Entry Mode Command
*** increment, no display shift
ldaa #$00       ;entry mode command
jsr LCD_WRITE   ;write data to LCD
ldaa #$60       ;entry mode command
jsr LCD_WRITE   ;write data to LCD

*** SEND MESSAGES
*** Messages have address and content predefined
jsr MESSAGE1     ;send Message1
jsr MESSAGE2     ;send Message2

DUMMY             ;done with example

*** SUBROUTINES *******************************************************
*** Routine creates a delay according to the formula
*** TIME*~100µs using an 8MHz internal bus
*** Cycle count per instruction shown
VAR_DELAY ldab #199T     ;1
L1     nop              ;1
dbne B,L1           ;3
dec TIME           ;4
bne VAR_DELAY       ;3
crts              ;5

*** Routine sends LCD Data
LCD_WRITE staa LCD_DATA ;clock in data
bset LCD_CTRL,E
bclr LCD_CTRL,E
ldaa #107T         ;40µs delay for LCD
L2     dbne A,L2      ;3
crts

*** Routine sends LCD Address
LCD_ADDR bclr LCD_CTRL,RS ;LCD in command mode
staa LCD_DATA
bset LCD_CTRL,E   ;clock in data
bclr LCD_CTRL,E
ldaa #107T         ;40µs delay for LCD
L4     dbne A,L4      ;3
bset LCD_CTRL,RS   ;LCD in data mode
crts

*** Message Routines
MESSAGE1 ldaa #$80     ;addr = $04 MSB
jsr LCD_ADDR      ;send addr to LCD
ldaa #$40         ;addr = $04 LSB
jsr LCD_ADDR      ;send addr to LCD
ldx #0
L3    ldaa MSG1,X     ;load AccA w/char from msg
beq OUTMSG1      ;end of msg?
jsr LCD_WRITE    ;write data to LCD
ldaa MSG1,X      ;load AccA w/char from msg
asla
asla
asla
asla

AN1774
MOTOROLA
Application Note

```assembly
; write data to LCD
jsr LCD_WRITE

; increment X
inx

; loop to finish msg
bra L3
```

```assembly
MESSAGE2
lda #0           ; addr = $44 MSB
jsr LCD_ADDR    ; send addr to LCD
lda #40          ; addr = $44 LSB
jsr LCD_ADDR    ; send addr to LCD
ldx #0
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
inx              ; increment X
bra L5           ; loop to finish msg
```

```assembly
*** MESSAGE STORAGE *********************************************
ORG MSG_STORAGE
MSG1        db 'Motorola'
db 0

MSG2        db 'HC12 MCU'
db 0

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

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola and are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217.
1-800-441-2447 or 1-303-675-2140. Customer Focus Center, 1-800-521-6274
JAPAN: Motorola Japan Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-Ku, Tokyo, Japan, 03-5487-8488
ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd., Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate,
Tai Po, New Territories, Hong Kong, 852-26629298
Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/;
TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848
HOME PAGE: http://motorola.com/sps/

Mfax is a trademark of Motorola, Inc.