# Interfacing a Hitachi HD44780 to a Motorola 68HC11 or Motorola 68HC12

## Table of Contents

## Introduction:

This document is intended to explain the basics of interfacing a Hitachi HD44780 LCD Controller with the Motorola 68HC11 and Motorola 68HC12 microcontrollers and to provide sample code in the form of a C header file for the HC11 and HC12 and assembly code subroutines for the HC11. All code for this document was developed and tested with Introl C 4.0. All tables, diagrams, and charts from the Hitachi data sheets unless credited otherwise.

## Hardware Operation:

The hardware in the HD44780 is mostly transparent to the programmer. As a result many of the features do not need an in depth explanation. Those readers interested in more detailed information should refer to the Hitachi Data Sheet for the HD44780. For this project the Optrex DMC-16204 Display Module (DigiKey part number 73-1033ND) was used. This incorporates the HD44780 as the on board LCD Screen controller.

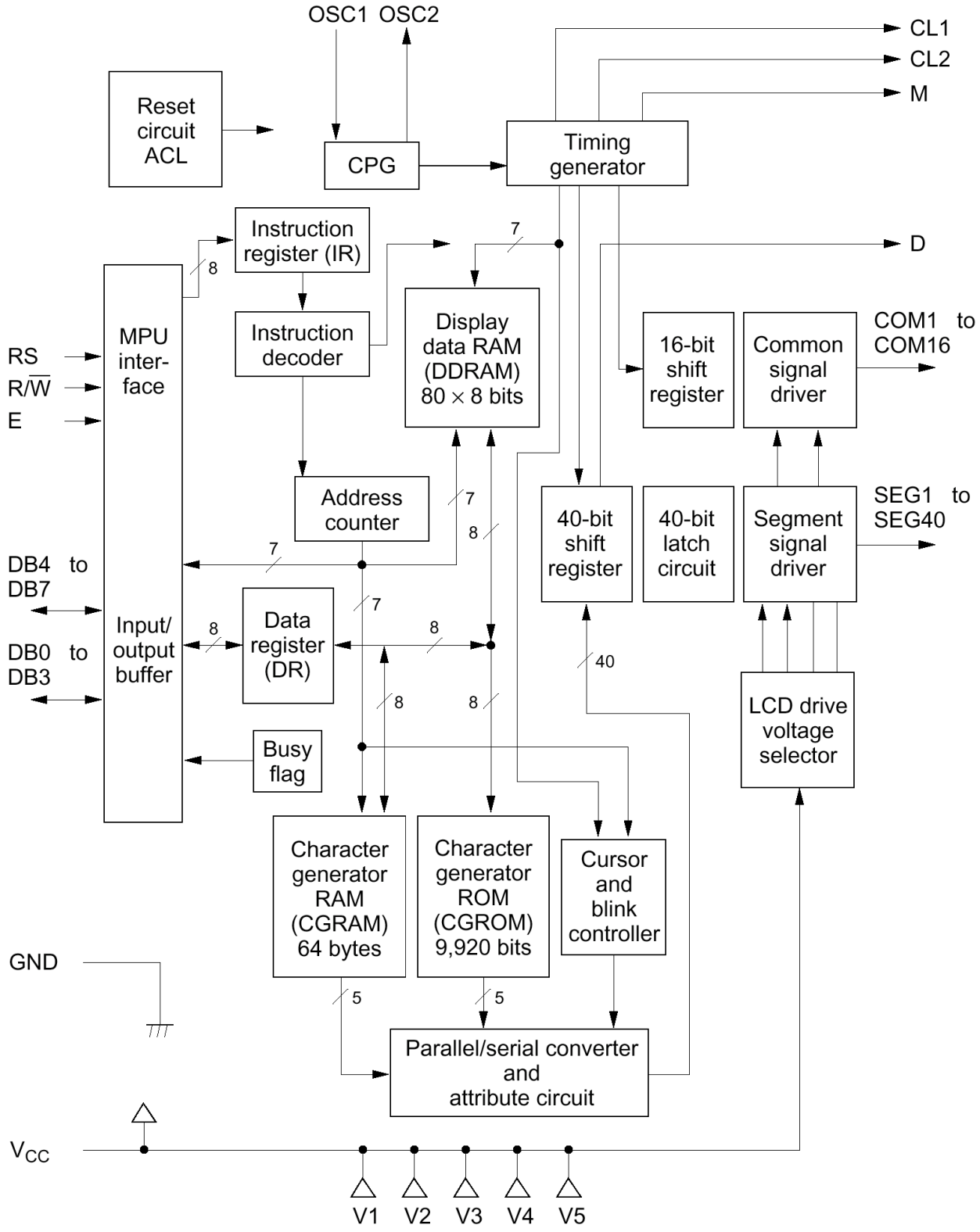
The Optrex DMC-16204 Display Module has 14 connections between itself and the microprocessor. On the Optrex DNC-16204, pin 2 provides power and pin 1 is connected to ground. Pin 3 controls the brightness level of the screen and is connected to the wiper of a 10 kOhm potentiometer. Pin 4 is the register select of the LCD screen. This is used to select between the instruction register or address counter of the HD44780. When the input to the pin is low the instruction register is active and the data register is active when the input is high. Pin 5 is the Read/Write select. When the input to the pin is high, the HD44780 is in read mode, when the input is low it is set up for a write. Pin 6 is the LCD enable. This is used to clock data and instructions into the HD44780. Pins 7 to 14 are the data pins. Pin 14 also doubles as the Busy Flag for the LCD screen. While many LCD screens use this order for the pins, the exact pin configuration may vary by part type and manufacturer. Be sure to refer to the LCD documentation before using this code to ensure that they are compatible. Schematics for wiring the LCD screen to the 6811 and 6812 are included in appendix A.

The basic operation of the screen is controlled by the state of the Register Select (RS) and the Read/Write (R/W) pins. These operations are summarized in Table 1.

Table 1: Register Selection

| RS | R/$\overline{W}$ | Operation |
|---|---|---|
| 0 | 0 | IR write as an internal operation (display clear, etc.) |
| 0 | 1 | Read busy flag (DB7) and address counter (DB0 to DB6) |
| 1 | 0 | DR write as an internal operation (DR to DDRAM or CGRAM) |
| 1 | 1 | DR read as an internal operation (DDRAM or CGRAM to DR) |

Figure 1: HD44780U Block Diagram



**Memory:**
The HD44780 provides an 80x8 bit Display Data RAM (DDRAM). This is used to store the data that is being displayed on the screen. This allows the HD44780 to store up to 40 characters per line. It is important to note that the DMC-16204 will display only 16 characters per line. The extra memory here can be used to store characters that may then be shifted onto the screen. All data to be displayed must be stored in the form of an 8 bit ASCII code character.

Figure 2: 1 Line Display

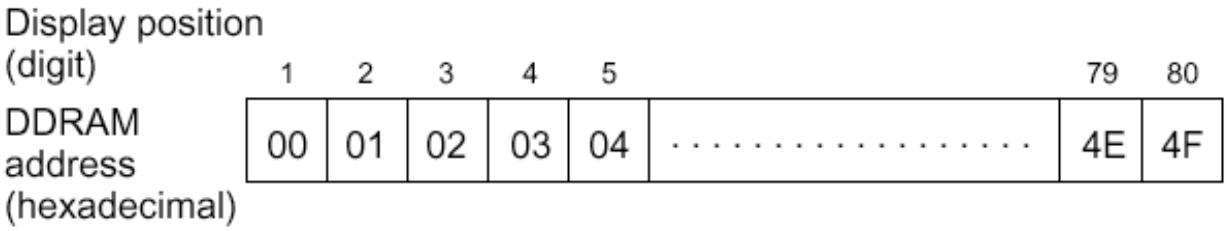
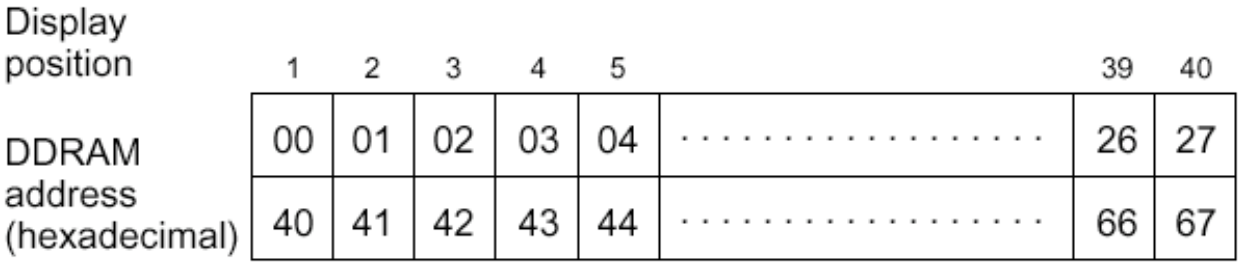



Figure 3: 2 Line Display



**Instructions:**
The HD44780 has a number of different instructions that it can execute. These instructions are listed in the following table:

Table 2: Instructions

| Instruction | Code RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | Description | Execution Time (max) (when $f_{cp}$ or $f_{OSC}$ is 270 kHz) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clear display | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Clears entire display and sets DDRAM address 0 in address counter. | |
| Return home | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | — | Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged. | 1.52 ms |
| Entry mode set | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | Sets cursor move direction and specifies display shift. These operations are performed during data write and read. | 37 μs |
| Display on/off control | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B). | 37 μs |
| Cursor or display shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | — | — | Moves cursor and shifts display without changing DDRAM contents. | 37 μs |
| Function set | 0 | 0 | 0 | 0 | 1 | DL | N | F | — | — | Sets interface data length (DL), number of display lines (N), and character font (F). | 37 μs |
| Set CGRAM address | 0 | 0 | 0 | 1 | ACG | ACG | ACG | ACG | ACG | ACG | Sets CGRAM address. CGRAM data is sent and received after this setting. | 37 μs |
| Set DDRAM address | 0 | 0 | 1 | ADD | ADD | ADD | ADD | ADD | ADD | ADD | Sets DDRAM address. DDRAM data is sent and received after this setting. | 37 μs |
| Read busy flag & address | 0 | 1 | BF | AC | AC | AC | AC | AC | AC | AC | Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents. | 0 μs |
| Write data to CG or DDRAM | 1 | 0 | Write data | | | | | | | | Writes data into DDRAM or CGRAM. | 37 μs<br>$t_{ADD}$ = 4 μs* |
| Read data from CG or DDRAM | 1 | 1 | Read data | | | | | | | | Reads data from DDRAM or CGRAM. | 37 μs<br>$t_{ADD}$ = 4 μs* |
| | I/D = 1: Increment<br>I/D = 0: Decrement<br>S = 1: Accompanies display shift<br>S/C = 1: Display shift<br>S/C = 0: Cursor move<br>R/L = 1: Shift to the right<br>R/L = 0: Shift to the left<br>DL = 1: 8 bits, DL = 0: 4 bits<br>N = 1: 2 lines, N = 0: 1 line<br>F = 1: 5 × 10 dots, F = 0: 5 × 8 dots<br>BF = 1: Internally operating<br>BF = 0: Instructions acceptable | | | | | | | | | | DDRAM: Display data RAM<br>CGRAM: Character generator RAM<br>ACG: CGRAM address<br>ADD: DDRAM address (corresponds to cursor address)<br>AC: Address counter used for both DD and CGRAM addresses | Execution time changes when frequency changes<br>Example:<br>When $f_{cp}$ or $f_{OSC}$ is 250 kHz,<br>$37\ \mu s \times \frac{270}{250} = 40\ \mu s$ |

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, $t_{ADD}$ is the time elapsed after the busy flag turns off until the address counter is updated.

It is important to note that the HD44780 can only execute one instruction at a time. Before sending an instruction to the display, the busy flag must to be read. If the busy flag is zero, then the instruction can be sent to the display, otherwise the instruction must be held by the microprocessor until the current instruction has completed execution and the busy flag is cleared.

**Instruction Descriptions:**

**Clear Display**
This instruction writes a 0x20 to all locations in the DDRAM. It also sets the DDRAM address to zero and unshifts the display, if it had been shifted. It also sets the display to increment mode.

**Return Home**
The DDRAM Address is set to zero and the display is unshifted, if it had been shifted.

**Entry Mode Set**
This instruction has two parameters, which it controls. The first is I/D. If this bit is high, the display increments the DDRAM address by one every time a character is written to the screen. If it is low, then the display address will be decremented by one every time a character is written. The second parameter is S. When S is high, the display shifts after a character is written to the screen. It will shift to the right if I/D = 0 or to the left if I/D=1. When S is low, the display does not shift when a character is written.

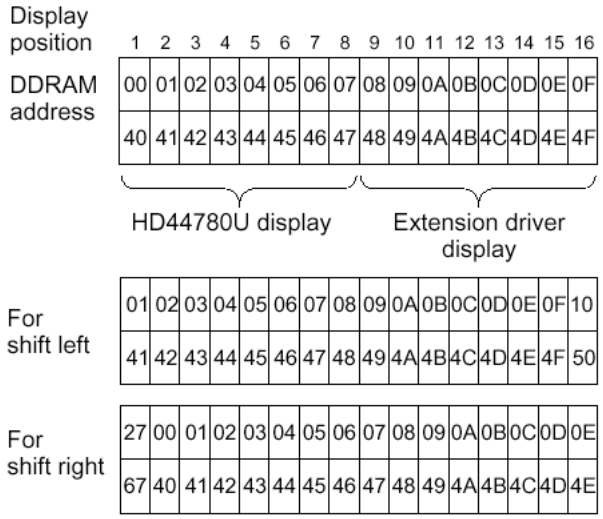
**Display Control On/Off**
This instruction has 3 parameters that the user can set. The first is D. This turns the display on when it is high and off when it is low. The second parameter is C. This displays the cursor when it is high and turns the cursor off when it is low. The last parameter is B. When this is high the character indicated by the cursor will blink. When it is low the display will not blink.

**Cursor or Display Shift**
This instruction shifts either the cursor or display by 1 character, without modifying the data stored in the DDRAM. The direction of the shift is determined by the value in the R/L bit. Both lines shift simultaneously. The shifting type and direction are summarized in the following table:

Table 3: Shift Functions

| S/C | R/L | |
|---|---|---|
| 0 | 0 | Shifts the cursor position to the left. (AC is decremented by one.) |
| 0 | 1 | Shifts the cursor position to the right. (AC is incremented by one.) |
| 1 | 0 | Shifts the entire display to the left. The cursor follows the display shift. |
| 1 | 1 | Shifts the entire display to the right. The cursor follows the display shift. |

Figure 4: 2 Line by 16 Character Display




**Function Set**
This instruction is used to initialize the display and what format the display will be using. This is done only during the initialization process and it may not be changed later in the program. DL is the data length of the interface. For this program, DL is always high, since the only the 8 bit interface is used. N is the number of display lines and F is the font size.

Table 4: Function Set

| N | F | No. of Display Lines | Character Font | Duty Factor | Remarks |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 5 × 8 dots | 1/8 | |
| 0 | 1 | 1 | 5 × 10 dots | 1/11 | |
| 1 | * | 2 | 5 × 8 dots | 1/16 | Cannot display two lines for 5 × 10 dot character font |

Note: * Indicates don't care.

**Set DDRAM Address**
This sets the DDRAM to the address included in the instruction. When the display is in single line mode the addresses range from 0x00 to 0x4F. In 2 line mode, the instructions range from 0x00 to 0x27 for the first line and from 0x40 to 0x67 for the second line.

**Read Busy Flag**
This instruction sends the state of the Busy Flag to the microcontroller. This appears on bit 7 and is used to determine if the LCD screen controller is still executing an instruction. If the bit is high, then there is an instruction executing that must be completed before another instruction can be written to the LCD screen controller

**Write Data to DDRAM**
This instruction writes an 8-bit pattern to the DDRAM.

## Software Operation:

The code to control the LCD screen was developed as both a C header files for the 68HC11 and 68HC12 and as several assembly language subroutines for the 68HC11. The C header files were written to provide an easy interface to the LCD screen. The two files are similar with the only differences being the ports used by the screen and the delay cycles used. The assembly code is much smaller then the C code and is also in many ways far more flexible. The code for these files is included in the Appendices.

**C Code:**
The C code was written as a header file that could be included in any program that interfaces with an LCD screen. For the 68HC11, the code uses Port C for writing data and Port A[3:5] for the control signals. For the 68HC12, the header file uses Port H for writing data and Port G[0:2] for the control signals.

The header file contains six functions to control the LCD screen. These are OpenXLCD, SetDDRamAddr, BusyXLCD, WriteCmdXLCD, WriteDataXLCD, and WriteBuffer. These functions provide all the basic features needed to display data on the screen and to position the cursor.

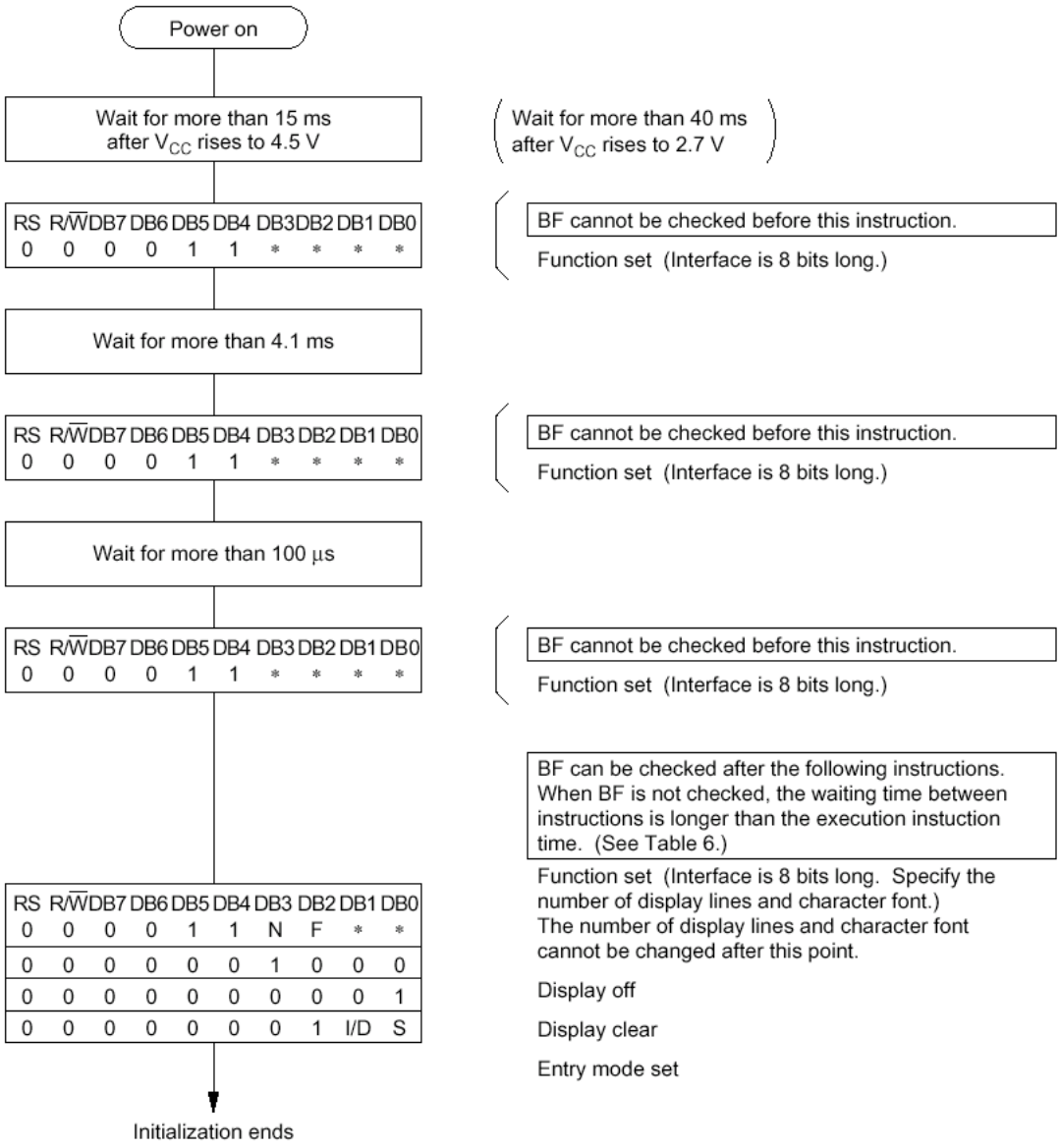
**OpenXLCD:**
This function executes all the initialization routines required by the HD44780 before it can be used. This routine sets the controller for 8-bit data entry and also initializes the number of display lines and the character font of the LCD screen. This is done by passing the desired type of display to the controller when OpenXLCD is called. The choices available are:

| **Screen Display** | **Value** |
|---|---|
| 5x8 single line | 0x30 |
| 5x8 double line | 0x3F |
| 5x10 single line | 0x34 |

These values may only be changed on startup. They can not be changed after the LCD screen has been initialized. The initialization routine ends by turning on the display and cursor, clearing the entire display and setting the DDRAM address to 0.

It is important to note that this function must be customized for the processor on which it is running. This is because there are several delay loops that are executed by this routine. These are all time dependant and were designed around the microprocessor's clock, 2 MHz for the 6811 and 8 MHz for the 6812. If this code is ported to other processors then these, the delay loops will need to be rewritten to take into account the clock frequency of the processor you are using.

Figure 5: 8 Bit Interface




**WriteCmdXLCD:**
This function is used to write commands to the LCD screen. The command to be issued is passed in as a parameter of the function.
There are several different commands, which can be issued to the LCD controller. These are for clearing the display, resetting the DDRAM address to 0, turning the display and cursor on or off, and shifting the display and cursor. The basic format of each of these is summarized in Table 2 above.

**WriteDataXLCD:**
This function is used to write data to the LCD screen. The data to be written is passed in as a parameter of this function. This is very similar in operation to the WriteCmdXLCD routine. The data to be displayed must be written to the display as an ASCII character. The character set stored in the controller is listed in Table 5 below.

Table 5: Character Set


Note: The user can specify any pattern for character-generator RAM.

**SetDDRamAddr:**
This function sets the address of the cursor. This is a special case of the WriteCmdXLCD function. The function is almost identical to WriteCmdXLCD, the only difference being that the value passed into the function is logically ORed with 0x80, which places a 1 in the leading location. This is used to signify that an address is being sent to the display as opposed to a character or other command.

**BusyXLCD:**
This function is used to check the busy flag of the LCD screen. When the busy flag is high, the LCD controller is still processing a previous command and cannot accept any new instructions. As a result, the busy flag must be checked before each attempt to write a command or data to the LCD screen. This step is incorporated in the WriteCmdXLCD, WriteDataXLCD and SetDDRamAddr functions, as they all call the BusyXLCD instruction as their first operation, and wait until the flag is cleared before proceeding to issue a new instruction.

**WriteBuffer:**
This command is used to write a string of characters, stored in a buffer, to the LCD screen. This function contains a while loop that simply makes repeated calls to the WriteDataXLCD function until the entire buffer has been transmitted. The only limitation on the use of the buffer is that only data can be sent to the screen. Instructions must be sent separately by calling WriteCmdXLCD.

**Test Code:**
There are two C programs included with this document. One is LCDtest11.c and the other is LCDtest12.c. Both programs execute the same set of operations and are used to exercise the LCD Screen to ensure that it is working properly by issuing a variety of instructions to the LCD screen.

The code initially turns the display off for a short time. It then prints:

```
 ____________________
|Good Morning        |
|Dave                |
```

After another short delay, the cursor moves to address 0xCA and prints "Hello" at this location.

```
 ____________________
|Good Morning        |
|Dave        Hello   |
```

Following yet another delay, "ABCDE" is written to Address 0x41 and the display is shifted left 5 times.

```
 ____________________
|Morning    ABCDE    |
|     Hello          |
```

The last instruction clears the display after another delay.

```
 ____________________
|                    |
|                    |
```

**Assembly Code:**
The assembly code for the LCD screen consists of two subroutines that can be included with any program to provide an interface to the LCD controller to the Motorola 68HC11. It is a good idea when using these programs to allow for a large stack, since all parameters are passed via the stack. The subroutines that are used are Writelcd, which writes the data to the LCD screen and Initlcd which executes all the screen initialization routines.

**Writelcd:**
This subroutine handles all data and instruction writes to the LCD screen. The data to be written is passed to the subroutine by the A register and the RS and R/W states are passed by the B register. In the subroutine, these values are then pushed onto the stack. The subroutine then checks the busy flag. Once the busy flag is clear the data is popped off the stack and written to Port C. Then the control pin states are then popped off the stack and written to Port A. Several clock cycles later the enable is pulsed and the data is written to the HD44780. The control pins are then cleared and the subroutine returns control to the calling function.

**Initlcd:**
This subroutine functions identically to the OpenXLCD function in the C code. The screen type is passed to the subroutine in the A register and is then pushed onto the stack. It is popped off the stack later in the subroutine so that it can be written to Port C. The user does not need to specify any other information when using this subroutine.

**Test Code:**
This is very simple code that shows how the Writelcd and Initlcd subroutines are used in a program. This is not, however, intended in any way as a diagnostic tool to check the functioning of the screen. If there is a question as to whether or not the screen is bad, it is recommended that the sample C code be run, as it provides a far more comprehensive test of the functions and features of the HD44780.

**Function differences between the C code and Assembly code routines**

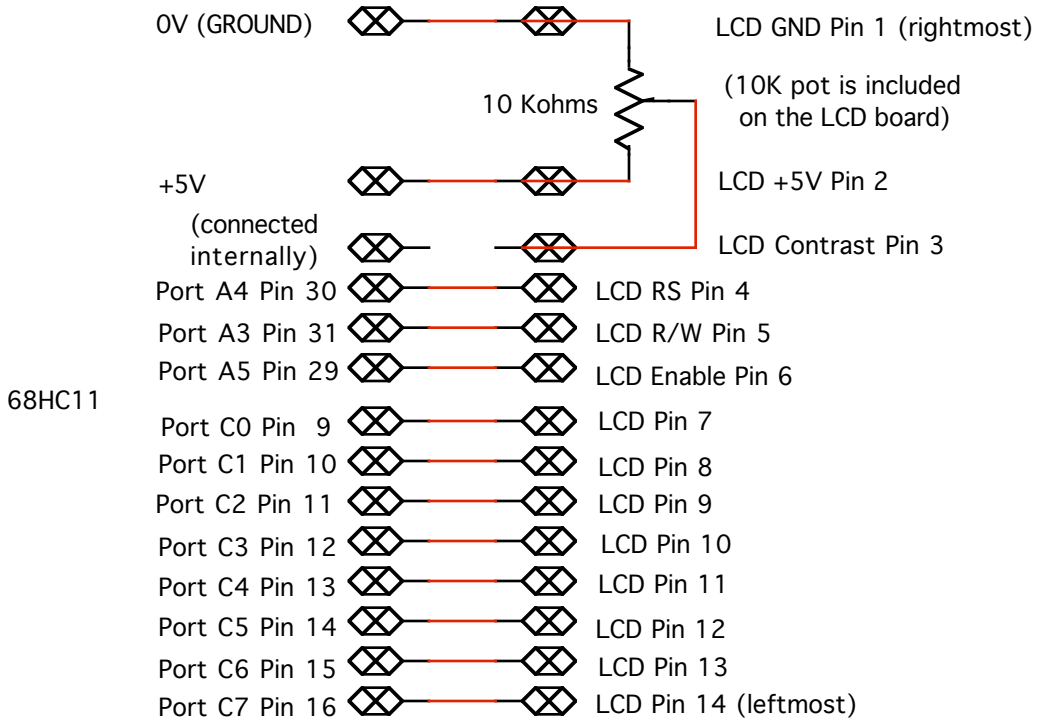
There are very few actual differences between the C header file and the assembly code subroutines. The differences between them are summarized briefly below.

1) The assembly code version reduces the number of different operations need by the programmer as it combines the WriteCmdXLCd, WriteDataXLCD, DDRamAddr, and BusyXLCD functions into a single subroutine. This can be done since the only difference between the first 3 instructions is the state of the RS and R/W pins. As the programmer must pass the state of the control pins to the subroutine from the main code, the functions can be combined into a single subroutine. In addition as there is only one subroutine, the BusyXLCD functions can be included in that subroutine since it is not called by any other subroutine.

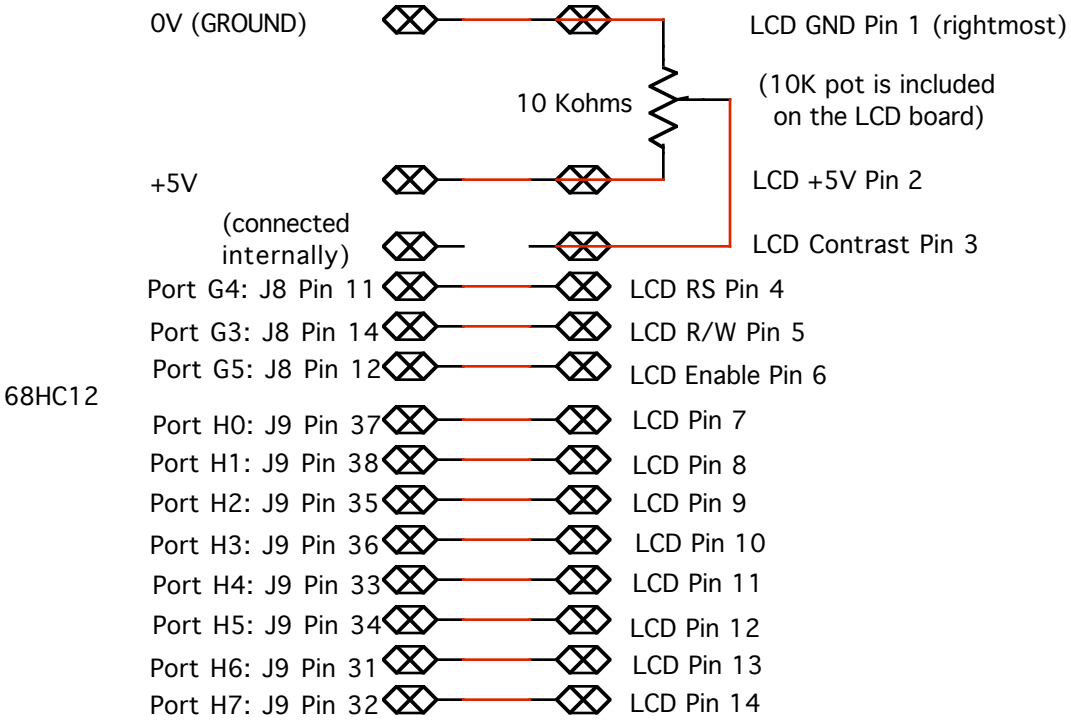
2) The C header file provides a function to write a string of ASCII characters to the screen. This function was omitted from the assembly code subroutine. This was left as an exercise for the students. To write a string of data or a sequence of data and instructions, the Writelcd subroutine must be slightly modified.

The programmer simply pushs the data or instructions, and the control signals for them, onto the stack in reverse execution order. The new Writelcd subroutine must recieve the number of instructions stored on the stack and continue processing them until all the instructions have been transmitted. This has an advantage over the C header WriteBuffer function in that it can handle both instructions and data since the control signals are passed along with the data to be written to the screen.

# Appendix A: Wiring Connections




68HC11 LCD Wiring Connections




68HC12 LCD Wiring Connections

## Appendix B: LCD11.h and LCDtest11.c

### LCD11.h

```
// LCD Screen routines for the Motorola 6811 using a Hitachi //
HD44780
// Written by Lee Rosenberg - rosenl@rpi.edu
// Developed for use with Introl C 4.0
// October 21, 1998

#include <HC11A1.h>

void OpenXLCD(char);      // configures I/O pins for external LCD
void SetDDRamAddr(char);  // sets display data address
char BusyXLCD(void);      // returns busy status of the LCD
void WriteCmdXLCD(char);  // write a command to the LCD
void WriteDataXLCD(char); // writes data byte to the LCD
void WriteBuffer(char *buffer); //Writes a string to the LCD

/*********************************************************

Write Buffer
     Function: Write a string of bytes to the HD44780
     Input Parameters: char *buffer
     Return Type: None

*********************************************************/

void WriteBuffer(char *buffer)
{
     while(*buffer)                 // while buffer not empty
     {
          while(BusyXLCD());        // check if screen busy
          WriteDataXLCD(*buffer);   // write a character
          buffer++;                 // increment pointer
     }
     return;
}

/*********************************************************

OpenXLCD

     Function: This configures the LCD screen.
     Input Parameters: char lcdtype
     Return Type: None
     Notes: This function must be run before the LCD screen
            can be used.

*********************************************************/

void OpenXLCD(char lcdtype)
{    int i;
     _H11PORTC = 0;       // initialize control port A and
     _H11DDRC = 0x00;     // Data port C
```

```
     _H11PORTA = 0x00;

// delay for 15ms.  This is customized for the HC11 and must //be
changed for other processors

     for(i=0; i<40,000; i++);

// set up interface to LCD
     _H11DDRC = 0xFF;
     _H11PORTC = 0x3F;    // Function set command (8 bit)

     _H11PORTA = 0x20;    // clock command in
     for(i=0; i<30;i++);  // delay for ~ 15 us
     _H11PORTA = 0x00;

// delay for at least 4.1 ms

     for(i=0;i<9000;i++);

// setup interface
     _H11PORTC=0x3F;      // Function set command (8 bit)
     _H11PORTA = 0x20;    // clock in command
     for(i=0;i<30;i++);   // delay for ~15 us
     _H11PORTA = 0x00;

// delay for at least 100us
     for(i=0;i<500;i++);

// set up interface
     _H11PORTC = 0x3F;    // function set command (8 bit)
     _H11PORTA = 0x20;
     for(i=0;i<30;i++);   // delay for ~15 us
     _H11PORTA = 0x00;

     WriteCmdXLCD(lcdtype); // function set 8 bit interface
     WriteCmdXLCD(0x0C);
     WriteCmdXLCD(0x01);    // turn off display
     return;
}


/*******************************************************

WriteCmdXLCD

     Function: Writes a command to the controller
     Input Parameter: char cmd
     Return Type: None
     Notes: Before writing the command the function checks
            that the display is not busy by calling
            BusyXLCD.
*******************************************************/

void WriteCmdXLCD(char cmd)
{
```

```
     int i;
     while(BusyXLCD()); // Check LCD is not in use
     _H11DDRC = 0xFF;
     _H11PORTC = cmd;     // write cmd to port
     _H11PORTA = 0x00;    // set control signals

     for(i=0; i<30; i++); // delay for ~15 us
     _H11PORTA=0x20;      // clock in the command
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11PORTA=0x00;

     for(i=0;i<30;i++);        // delay for ~15 us
     _H11DDRC=0x00;

     return;
}


/*********************************************************

SetDDRamAddr
     Function: Set the address of the LCD controller
     Input Parameter: char DDaddr
     Return Type: None
     Notes: This function sets the address of the LCD screen
            to the address that is passed in as a char.  The
            address is automatically modified to the correct
            format for the screen.

*********************************************************/

void SetDDRamAddr(char DDaddr)
{
     int i;
     while(BusyXLCD());        // check if screen is in use
     _H11DDRC=0xFF;
     _H11PORTC=(DDaddr | 0x80);// write cmd and addr to port
     _H11PORTA=0x00;
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11PORTA=0x20;           // clock in the command
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11PORTA=0x00;
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11DDRC =0x00;
     return;
}


/*********************************************************

BusyXLCD
     Function: This checks the busy status of the HD 44780
     Input Parameter: None
     Return Type: char
     Notes: This is necessary to ensure that the LCD screen
```

```
            is ready to recieve data.

*************************************************************/

char BusyXLCD(void)
{
     int i;
     _H11DDRC=0x00;
     _H11PORTA=0x08;      // set control bits
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11PORTA=0x28;      // clock them in
     for(i=0;i<30;i++);        // delay for ~15 us
     if(_H11PORTC & 0x80)  // read busy flag
     {
          _H11PORTA = 0x00;    // if set
          return 1;
     }
     else
     {
          _H11PORTA = 0x00;   // if clear
          return 0;
     }
}


/*************************************************************

WriteDataXLCD
     Function: Writes data to the LCD
     Input Parameter: char data
     Return Type: None
     Notes: This function takes ascii data and writes it to
            the LCD screen. All data is passed in as a char.

*************************************************************/

void WriteDataXLCD(char data)
{
     int i;
     while(BusyXLCD());   // check if screen is ready
     _H11DDRC = 0xFF;
     _H11PORTC = data;    // write data
     _H11PORTA = 0x10;
     for(i=0;i<30;i++);        // delay for ~15 us
     _H11PORTA=0x30;      // clock in data
     for(i=0;i<30;i++);        // dlay for ~ 15 us
     _H11PORTA=0x00;
     _H11DDRC= 0x00;
     return;
}
```

**LCDtest11.c**

```
/* Basic test program for the Hitachi HD 44780
This will test all the major functions and commands to ensure that
the screen is functioning correctly. */

// All necessary include statements

#include <HC11A1.h>       // register declarations
#include <introl.h>       // Introl functions
#include <stdio.h>        // I/O commands
#include <stdlib.h>       // Standard C functions
#include <lcd11.h>        // LCD functions

void main()
{
     int i, j;
     char buffer[]="hello";

     OpenXLCD(0x3F);      //intialize the screen

     WriteCmdXLCD(0x80);  // set address to 0

     WriteDataXLCD(0x47); // write "Good Morning Dave"
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x64);
     WriteDataXLCD(0x20);
     WriteDataXLCD(0x4D);
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x72);
     WriteDataXLCD(0x6E);
     WriteDataXLCD(0x69);
     WriteDataXLCD(0x6E);
     WriteDataXLCD(0x67);
     WriteCmdXLCD(0xC0);
     WriteDataXLCD(0x44);
     WriteDataXLCD(0x61);
     WriteDataXLCD(0x76);
     WriteDataXLCD(0x65);

     WriteCmdXLCD(0x08);  // turn off display

     for(i=0; i<10; i++)       // delay
          for(j=0; j<40000; j++);

     WriteCmdXLCD(0x0C);  // turn on display and cursor

     for(i=0; i<10; i++)       // delay
          for(j=0; j<40000; j++);
     SetDDRamAddr(0xCA);  // set cursor address to 4F

     WriteBuffer(&buffer);     //write buffer to screen
     for(i=0;i<10;i++)    // delay
          for(j=0;j<40000;j++);
```

```
      SetDDRamAddr(0x90);  // go to address 16
      WriteDataXLCD(0x41); // write ABCDE
      WriteDataXLCD(0x42);
      WriteDataXLCD(0x43);
      WriteDataXLCD(0x44);
      WriteDataXLCD(0x45);

      WriteCmdXLCD(0x18);  // shift display left 5 times
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);

      for(i=0;i<10;i++)    // delay
           for(j=0;j<40000;j++);

      WriteCmdXLCD(0x01);  // clear display

}
```

## Appendix C: LCD12.c and LCDtest12.c

### LCD12.c

```
// LCD Screen routines for the Motorola 6812 using a Hitachi
// HD44780
// Written by Lee Rosenberg - rosenl@rpi.edu
// Developed for use with Introl C 4.0
// October 21, 1998

#include <hc812a4.h>      // register declarations
#include <dbug12.h>       // D-Bug12 monitor

void OpenXLCD(char); // configures I/O pins for LCD
void SetDDRamAddr(char);  // sets display data address
char BusyXLCD(void); // returns busy status of the LCD
void WriteCmdXLCD(char);  // write a command to the LCD
void WriteDataXLCD(char); // writes data byte to the LCD
void WriteBuffer(char *buffer); // Writes a string to LCD


/*********************************************************

Write Buffer
     Function: Write a string of bytes to the HD44780
     Input Parameters: char *buffer
     Return Type: None

*********************************************************/

void WriteBuffer(char *buffer)
{
     while(*buffer)       // while buffer not empty
     {
          while(BusyXLCD());   // check if screen busy
          WriteDataXLCD(*buffer);   // write character
          buffer++;       // increment pointer
     }
     return;
}


/*********************************************************

OpenXLCD

     Function: This configures the LCD screen.
     Input Parameters: char lcdtype
     Return Type: None
     Notes: This function must be run before the LCD screen
            can be used.

*********************************************************/

void OpenXLCD(char lcdtype)
```

```
{    int i;
     _H12PORTH = 0;
     _H12DDRH = 0x00;
     _H12PORTG = 0x00;
     _H12DDRG= 0xFF;

// delay for 15ms.  This is customized for the HC12 and must // be
changed for other processors

     for(i=0; i<130,000; i++);

// set up interface to LCD
     _H12DDRH = 0xFF;
     _H12PORTH = 0x3F;    // Function set command (8 bit)
     _H12PORTG = 0x20;    // clock command in
     for(i=0; i<100;i++); // delay for ~12.5 us
     _H12PORTG = 0x00;

// delay for at least 4.1 ms

     for(i=0;i<40000;i++);

// setup interface
     _H12PORTH=0x3F;      // Function set command (8 bit)
     _H12PORTG = 0x20;    // clock in command
     for(i=0;i<100;i++);  // delay for ~12.5 us
     _H12PORTG = 0x00;

// delay for at least 100us

     for(i=0;i<1000;i++);

//   set up interface
     _H12PORTH = 0x3F;    // function set command (8 bit)
     _H12PORTG = 0x20;
     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12PORTG = 0x00;
     WriteCmdXLCD(lcdtype);    // function set command 8 bit
     WriteCmdXLCD(0x0C);
     WriteCmdXLCD(0x01);       // clear screen
     return;
}

/*******************************************************

WriteCmdXLCD

     Function: Writes a command to the controller
     Input Parameter: char cmd
     Return Type: None
     Notes: Before writing the command the function checks
            that the display is not busy by calling
            BusyXLCD.
*******************************************************/
void WriteCmdXLCD(char cmd)
```

```
{
     int i;
     while(BusyXLCD());   // check status of LCD
     _H12DDRG = 0xFF;
     _H12DDRH = 0xFF;
     _H12PORTG = 0x00;    // set control signals

     _H12PORTH = cmd;     // write cmd to port

     for(i=0; i<100; i++);     // delay for ~12.5 us
     _H12PORTG=0x20;      // clock the command in
     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12PORTG=0x00;

     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12DDRH=0x00;

     return;
}


/*********************************************************

SetDDRamAddr
     Function: Set the address of the LCD controller
     Input Parameter: char DDaddr
     Return Type: None
     Notes: This function sets the address of the LCD screen
            to the address that is passed in as a char.  The
            address is automatically modified to the correct
            format for the screen.

*********************************************************/

void SetDDRamAddr(char DDaddr)
{
     int i;
     while(BusyXLCD());        // check status of LCD
     _H12DDRG = 0xFF;
     _H12DDRH=0xFF;
     _H12PORTH=(DDaddr | 0x80);// write cmd and addr to port
     _H12PORTG=0x00;
     for(i=0;i<100;i++);            // delay for ~12.5 us
     _H12PORTG=0x20;           // clock command in
     for(i=0;i<100;i++);            // delay for ~12.5 us
     _H12PORTG=0x00;
     for(i=0;i<100;i++);            // delay for ~12.5 us
     _H12DDRH =0x00;
     return;
}


/*********************************************************

BusyXLCD
```

```
     Function: This checks the busy status of the HD 44780
     Input Parameter: None
     Return Type: char
     Notes: This is necessary to ensure that the LCD screen
            is ready to receive data.

*********************************************************/

char BusyXLCD(void)
{
     int i;
     _H12DDRG = 0xFF;
     _H12DDRH=0x00;
     _H12PORTG=0x08;      // set control bits
     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12PORTG=0x28;      // clock in the command
     for(i=0;i<100;i++);       // delay for ~12.5 us
     if(_H12PORTH & 0x80) // Read the busy flag
     {
          _H12PORTG = 0x00;    // if it is busy return 1
          return 1;
     }
     else
     {
          _H12PORTG = 0x00;    // if it is not busy return 0
          return 0;
     }
}

/*********************************************************

WriteDataXLCD
     Function: Writes data to the LCD
     Input Parameter: char data
     Return Type: None
     Notes: This function takes ASCII data and writes it to
            the LCD screen. All data is passed in as a char.

*********************************************************/

void WriteDataXLCD(char data)
{
     int i;
     while(BusyXLCD());   // check if the LCD is busy
     _H12DDRG = 0xFF;
     _H12DDRH = 0xFF;
     _H12PORTH = data;    // Write the data
     _H12PORTG = 0x10;
     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12PORTG=0x30;      // clock the data in
     for(i=0;i<100;i++);       // delay for ~12.5 us
     _H12PORTG=0x00;
     _H12DDRH= 0x00;
     return;
}
```

**LCDtest12.c**

```
/* basic test program for the Hitachi HD 44780
This will test all basic functions of the LCD screen to ensure it
is functioning properly.  */

#include <hc812a4.h>      // register declarations
#include <introl.h>       // Introl functions
#include <lcd12.h>        // LCD functions
#include <dbug12.h>       // D-Bug12 functions

void __main()
{
     int i, j;
     char buffer[]="hello";

     OpenXLCD(0x3F);      // initialize the screen

     WriteCmdXLCD(0x80);  // set address to 0

     WriteDataXLCD(0x47); // write "Good Morning Dave"
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x64);
     WriteDataXLCD(0x20);
     WriteDataXLCD(0x4D);
     WriteDataXLCD(0x6F);
     WriteDataXLCD(0x72);
     WriteDataXLCD(0x6E);
     WriteDataXLCD(0x69);
     WriteDataXLCD(0x6E);
     WriteDataXLCD(0x67);
     WriteCmdXLCD(0xC0);
     WriteDataXLCD(0x44);
     WriteDataXLCD(0x61);
     WriteDataXLCD(0x76);
     WriteDataXLCD(0x65);

     WriteCmdXLCD(0x08);  // turn off display

     for(i=0; i<10; i++)       // delay
          for(j=0; j<40000; j++);

     WriteCmdXLCD(0x0C);  // turn on display and cursor

     for(i=0; i<10; i++)
          for(j=0; j<40000; j++);
     SetDDRamAddr(0xCA);  // set cursor address to 4F

     WriteBuffer(&buffer);     //write buffer to screen
     for(i=0;i<10;i++)    // delay
          for(j=0;j<40000;j++);

     SetDDRamAddr(0x90);  // go to address 16
     WriteDataXLCD(0x41); // write ABCDE
```

```
      WriteDataXLCD(0x42);
      WriteDataXLCD(0x43);
      WriteDataXLCD(0x44);
      WriteDataXLCD(0x45);

      WriteCmdXLCD(0x18);  // shift display left 5 times
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);
      WriteCmdXLCD(0x18);

      for(i=0;i<10;i++)    // delay
            for(j=0;j<40000;j++);

      WriteCmdXLCD(0x01);  // clear display

}
```

## Appendix D: LCD.asm

```
* 6811 assembly code to interface with the Hitachi HD44780
* LCD Screen Controller.
* This code contains all the necessary subroutines to write
* to the screen.
* It also includes a simple main program that will execute
* the instructions.
* The subroutines are designed to be transferred to other
* programs and simply dropped in.
*
* Important Note: This program requires a significant amount
* of space on the stack.  Be sure to initialize the stack
* before beginning to run these routines.


* Equates
* buffalo operations

outa      equ  $ffb8  output the ASCII character in A
outstrg   equ  $ffca  output string at x
outcrlf   equ  $ffc4  output crlf
outlhlf   equ  $ffb2  output left nibble of a in ASCII
outrhlf   equ  $ffb5  output right nibble of a in ASCII
out2bsp   equ  $ffc1  output 2byte value at x in HEX
input     equ  $ffac  a=input() ; a=0 if no char entered
inchar    equ  $ffcd  a=input() ; loop till user enters char
upcase    equ  $ffa0  a=upcase(a)
wchekequ  $FFA3  z=1 if A={space,comma,tab}
dchekequ  $FFA6  z=1 if A={space,comma,tab,CR}

* Port Declarations
porta     equ  $1000
portc     equ  $1003
ddrc      equ  $1007



* This is the main program that calls the subroutines.

          org $c000
          jmp start
temp      rmb 1

* data to be displayed on the screen

test1fcc "point a"
          fcb $04
test2     fcc "point b"
          fcb $04

startlds #$DFFF; initialize the stack.

          ldaa #$3F      ; Load the screen type
          jsr initlcd    ; initialize the screen
```

```
ldaa #$80 ; set address to 0
ldab #$00 ; set control pins
jsr writelcd


ldaa #$47 ; write a character
ldab #$10
jsr writelcd


ldaa #$6F ; write a character
ldab #$10
jsr writelcd

ldaa #$6F ; write a character
ldab #$10
jsr writelcd

ldaa #$64 ; write a character
ldab #$10
jsr writelcd


ldaa #$20 ; write a character
ldab #$10
jsr writelcd

ldaa #$4d ; write a character
ldab #$10
jsr writelcd

ldaa #$6f ; write a character
ldab #$10
jsr writelcd

ldaa #$72 ; write a character
ldab #$10
jsr writelcd

ldaa #$6e ; write a character
ldab #$10
jsr writelcd

ldaa #$69 ; write a character
ldab #$10
jsr writelcd

ldaa #$6e ; write a character
ldab #$10
jsr writelcd

ldaa #$67 ; write a character
ldab #$10
jsr writelcd
```

```
          ldaa #$c0  ; write address
          ldab #$00
          jsr writelcd

          ldaa #$44  ; write a character
          ldab #$10
          jsr writelcd

          ldaa #$61  ; write a character
          ldab #$10
          jsr writelcd

          ldaa #$76  ; write a character
          ldab #$10
          jsr writelcd

          ldaa #$65  ; write a character
          ldab #$10
          jsr writelcd

          swi

*********************************************************
*    Initlcd
*
*    This subroutine initializes the LCD screen.  The LCD
*    screen format is passed in by the A register and is
*    stored on the stack.
*********************************************************

initlcd   psha       ; save the lcdtype
          ldaa #$00  ; clear ports A and C
          staa portc
          staa porta
          staa ddrc

          ldx #$9c40 ; wait for ~15ms
loop      dex
          cpx #$0000
          bne loop

          ldaa #$ff  ; set port C for output
          staa ddrc
          ldaa #$3f  ; write the function set command
          staa portc

          jsr delay  ; delay function

          ldaa #$20  ; pulse the enable bit
          staa porta
          jsr delay

          ldaa #$00  ; turn off enable
          staa porta
```

```
          staa portc

          ldx #$2328 ; wait ~4.1 ms
loop2 dex
          cpx #$00
          bne loop2

          ldaa #$3f  ; write the function set command
          staa portc
          jsr delay

          ldaa #$20  ; pulse the enable bit
          staa porta
          jsr delay

          ldaa #$00  ; turn off enable
          staa porta
          staa portc

          ldx #$1f4  ; wait ~100 us
loop3     dex
          cpx #$00
          bne loop3

          ldaa #$3f  ; write the function set command
          staa portc
          jsr delay

          ldaa #$20  ; pulse the enable bit
          staa porta
          jsr delay

          ldaa #$00  ; turn off enable
          staa porta
          pula       ; get the lcd type from the stack
          ldab #$00
          jsr writelcd    ; write the # of lines and font

          ldaa #$0C       ; clear screen
          ldab #$00
          jsr writelcd    ; write it to the screen

          ldaa #$01       ; set cursor to address 0.
          ldab #$00
          jsr writelcd

          rts

*****************************************************
* Writelcd
*
* This function checks the busy flag and then writes
* either data or instructions to the LCD screen.
* The data to be written is stored in register A
* and the control pin settings are stored in register
```

```
* B.  These are stored on the stack until they are
* needed.
************************************************

writelcd  pshb      ; store the rs and rw values
          psha      ; store the data

          ldaa #$00
          staa ddrc ; set port C for input

loop4     ldaa #$08 ; checking the busy flag
          staa porta

          jsr delay

          ldaa #$28 ; pulse the enable
          staa porta
          jsr delay

          ldaa portc ; read port C
          anda #$80  ; check the busy flag

          cmpa #$80  ; if flag set loop else
          beq loop4

          ldaa #$00 ; clear the enable
          staa porta
          ldaa #$ff ; set port C for output
          staa ddrc
          pula      ; write data to port C
          staa portc

          pulb      ; write control pins to port A
          stab porta
          stab temp

          jsr delay

          ldaa temp
          oraa #$20 ; set the enable and write it to port A
          staa porta
          jsr delay

          ldaa #$00 ; clear control pins
          staa porta

          rts


************************************************************
*    delay
*
* This function creates a delay to allow pins time to set up * and
stabilize.
```

```
*********************************************************

delay     ldab #$1E  ; wait 18 counts
waitloop  decb
          cmpb #$00
          bne waitloop

          rts
```