# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>ii</td>
</tr>
<tr>
<td>Tables</td>
<td>iii</td>
</tr>
<tr>
<td>Figures</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Optrex LCD Module</td>
<td>5</td>
</tr>
<tr>
<td>Interfacing the Optrex LCD Module</td>
<td>6</td>
</tr>
<tr>
<td>LCD Addressing</td>
<td>7</td>
</tr>
<tr>
<td>Grayhill Series 86 20-key Keypad and Decoding Logic</td>
<td>8</td>
</tr>
<tr>
<td>P&amp;E Micro BDM-MULTILINK</td>
<td>10</td>
</tr>
<tr>
<td>Technological Arts Adapt812DXLT 68HC812A4 Module</td>
<td>11</td>
</tr>
<tr>
<td>User Headers</td>
<td>16</td>
</tr>
<tr>
<td>Terminal Blocks</td>
<td>21</td>
</tr>
<tr>
<td>Jumper Blocks</td>
<td>22</td>
</tr>
<tr>
<td>Reserved Headers</td>
<td>23</td>
</tr>
<tr>
<td>Prototyping Area</td>
<td>26</td>
</tr>
<tr>
<td>Bench Top Unit External Connections</td>
<td>27</td>
</tr>
<tr>
<td>LCD Module</td>
<td>29</td>
</tr>
<tr>
<td>KEYPAD</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td>Appendix A:</td>
<td>33</td>
</tr>
<tr>
<td>ME405 Bench Top unit Diagram</td>
<td>33</td>
</tr>
<tr>
<td>ME405 Breakout Board Schematic</td>
<td>33</td>
</tr>
<tr>
<td>Appendix B:</td>
<td>39</td>
</tr>
<tr>
<td>Code</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Appendix C:</td>
<td>51</td>
</tr>
<tr>
<td>Electrical Specifications</td>
<td>51</td>
</tr>
</tbody>
</table>
Tables
Table 1: LCD Module Pin Assignments.................................................................5
Table 2: Header Serial........................................................................................16
Table 3: Header Timer.........................................................................................16
Table 4: Header J..................................................................................................17
Table 5: Header D................................................................................................17
Table 6: Header ADC..........................................................................................18
Table 7: Header C................................................................................................18
Table 8: Header H................................................................................................19
Table 9: Header Address Bus...............................................................................19
Table 10: Header Control...................................................................................20
Table 11: Terminal Blocks..................................................................................21
Table 12: Header BNC.......................................................................................23
Table 13: Header Keypad...................................................................................23
Table 14: Header LCD.......................................................................................24
Table 15: Header BKLT.......................................................................................24
Table 16: Power Input.........................................................................................25

Figures
Figure 1: Wiring Diagram between LCD and HC12..............................................6
Figure 2: LED Backlight......................................................................................7
Figure 3: LCD Addressing..................................................................................7
Figure 4: Generic Matrix Keypad.......................................................................8
Figure 5: Keypad-68HC812A4 Interface............................................................9
Figure 6: P&E Micro BDM-MULTILINK............................................................10
Figure 7: Adapt812DXLT..................................................................................11
Figure 8: Adapt812DXLT, H1 and H2 Pinouts....................................................12
Figure 9: Motorola 68HC812A4 Memory Map..................................................13
Figure 10: ME405 Board Schematic.................................................................14
Figure 11: ME405 Board Layout......................................................................15
Figure 12: Proto-Board......................................................................................26
Figure 13: Main Power.......................................................................................27
Figure 14: Signal and Optional Power...............................................................28
Figure 15: ME405 Bench Top Unit.................................................................34
Abstract:

The ME405 bench top unit is a complete Motorola 68HC12 development system. The ME405 bench top unit is comprised of six major components: an Optrex 2x40 LCD module, a Grayhill series 86 20 character keypad, a Technological Arts Adapt812DXLT 68HC812A4 module, a Pemicro Cable-12 BDM, the ME405 breakout board and a small prototyping area.
Optrex LCD Module

The Optrex DMC40202 module is a 40 character by 2-line LCD module with a yellow backlight. The Optrex module utilizes an 8-bit data bus and three control lines. Table 1 below lists the Optrex LCD module pins and their function.

Table 1: LCD Module Pin Assignments

<table>
<thead>
<tr>
<th>No.</th>
<th>Symbol</th>
<th>Level</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vss</td>
<td></td>
<td>Power Supply (0V, GND)</td>
</tr>
<tr>
<td>2</td>
<td>Vdd</td>
<td></td>
<td>Power Supply for Logic (5V, DC)</td>
</tr>
<tr>
<td>3</td>
<td>Vee</td>
<td></td>
<td>Power Supply for LCD Drive (~700mV, DC)</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>H / L</td>
<td>Register Select Signal</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>H / L</td>
<td>Read/Write Select Signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>H : Read, L : Write</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>H / L</td>
<td>Enable Signal (No pull-up Resister)</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
<td>H / L</td>
<td>Data Bus Line / Non-connection at 4-bit operation</td>
</tr>
<tr>
<td>8</td>
<td>DB1</td>
<td>H / L</td>
<td>Data Bus Line / Non-connection at 4-bit operation</td>
</tr>
<tr>
<td>9</td>
<td>DB2</td>
<td>H / L</td>
<td>Data Bus Line / Non-connection at 4-bit operation</td>
</tr>
<tr>
<td>10</td>
<td>DB3</td>
<td>H / L</td>
<td>Data Bus Line / Non-connection at 4-bit operation</td>
</tr>
<tr>
<td>11</td>
<td>DB4</td>
<td>H / L</td>
<td>Data Bus Line</td>
</tr>
<tr>
<td>12</td>
<td>DB5</td>
<td>H / L</td>
<td>Data Bus Line</td>
</tr>
<tr>
<td>13</td>
<td>DB6</td>
<td>H / L</td>
<td>Data Bus Line</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
<td>H / L</td>
<td>Data Bus Line</td>
</tr>
<tr>
<td>15</td>
<td>LED CATHODE</td>
<td></td>
<td>LED Cathode Terminal</td>
</tr>
<tr>
<td>16</td>
<td>LED ANODE</td>
<td></td>
<td>LED Anode Terminal</td>
</tr>
</tbody>
</table>
Interfacing the Optrex LCD Module

The external data bus on the 68HC12 operates much faster than the Optrex LCD module. The reaction times for all I/O lines on the Optrex module are relatively slow, approximately 160ns. In order to communicate with the LCD module port H is used to send and receive all data. Port H is designated a slow data bus. Figure 1 below is the connection diagram between the 68HC812A4 and the LCD module.

![Wiring Diagram between LCD and HC12](image)

**Figure 1: Wiring Diagram between LCD and HC12**

Notice, Port H is used as a data bus, and Port J pin 0-2 are used as the control lines for the LCD module.
The wiring diagram for the back light LED’s is shown in Figure 2 below.

![Figure 2: LED Backlight](image)

The current limiting resistor, $R_L$, for the LED backlight anode is found using the formula displayed below. The DMC40202 manual (40202are.pdf) suggests $V_s = 5$ Volts, $V_f = 4.0$ Volts, and $I_{\text{nom}} = 250\text{mA}$.

$$R_L = \frac{(V_s - V_f)}{I_{\text{nom}}}$$

$$R_L = \frac{(5 - 4)}{(.250)} \text{ A}$$

$R_L = 4\text{ohms}$

**LCD Addressing**

Each character of the LCD screen has its own individual address. The LCD has the following character addressing.

![Figure 3: LCD Addressing](chart)

Line 1 has addresses $00$-$27$ and Line 2 has addresses $40$-$67$, making up a total of eighty characters.
**Grayhill Series 86 20-key Keypad and Decoding Logic**

To provide keypad input to the ME405 bench top unit a Grayhill series 86, 20-key keypad was selected. This keypad is of the matrix type and must be decoded to provide useful information to the microcontroller. A Fairchild MM74C923 20-key keypad decoder was selected to provide decoding.

In a matrix or array keypad all keys share a common grid. When a key is depresses the grid shorts identify the key depressed. Refer to Figure 4 for a representation of a generic matrix keypad.

![Figure 4: Generic Matrix Keypad](image)

To capture a key depress, the Fairchild MM74C923 keypad decoder constantly scans the Grayhill keypad matrix. When a key is depressed the MM74C923 decodes the matrix value and pulls down the Data Available (DA) bit. The decoded key value is not presented on the data bus until the Output Enable (OE) bit is pulled low. On the ME405 bench top unit the DA bit is wired to a port J key wakeup line. When DA goes low, a key wakeup interrupt is generated. The 68HC812A4 services the key wakeup interrupt by pulling the OE bit low and reading the data bus for the decoded key value. Refer to the wiring diagram on the next page for a detail of the 68HC812A4-Keypad interface.
Figure 5: Keypad-68HC812A4 Interface

Note:  
C3 = .01uF  
C2 = 1.0uF
**P&E Micro BDM-MULTILINK**

To provide background-debugging support a P&E Micro BDM-MULTILINK is provided. BDM stands for Background Debugging Module; the BDM-MULTILINK provides complete access all internals of the Motorola 68HC12 family of microcontrollers. All registers, RAM and FLASH ROM can be edited real-time. This BDM supports breakpoint and code monitoring as well. Figure 6 is a picture of a BDM-MULTILINK mounted to the ME405 bench top unit.

![Figure 6: P&E Micro BDM-MULTILINK](image)

To connect the BDM it the host computer use a DB25 Male-to-Female 1284 IEEE parallel cable. Note: This is NOT a standard printer parallel cable. A DB25 Male-to-Female 1284 IEEE parallel cable is available at most computer stores.

P&E Micro has a complete development environment in conjunction with their BDM. Two applications in their kit are of interest to us are PROG12Z and ICD12Z. Use PROG12Z to program internal EEPROM and FLASH ROM. ICD12Z is an in circuit debugger, and provides access to the debugging features of the BDM.
The Motorola 68HC812A4 microcontroller is added to the ME405 bench top unit via a Technological Arts Adapt812DXLT development board. This board features a Motorola 68HC812A4 microcontroller, 32Kb of external data RAM, 128Kb of external FLASH EERPOM, two RS232 interfaces and an on board 5V 500mA regulator. Refer to Figure 7 for a general layout of the Adapt812DXLT development board.

**Figure 7: Adapt812DXLT**
Most connections to the Adapt812DXLT are handled via two 2x25 male headers, H1 and H2. These headers are mounted on the underside of the development board. Refer to Figure 8 for a pinout of these headers.

Figure 8: Adapt812DXLT, H1 and H2 Pinouts
For reference a 68HC812A4 memory map is included in Figure 9.

**Figure 9: Motorola 68HC812A4 Memory Map**
The ME405 breakout board provides key features to the ME405 bench top unit. It was designed here at Cal Poly for this course. There are two 50-pin headers on the Adapt812 board. These headers are very dense; packing many pins in a small area, and are not suitable for student laboratory experiments. The breakout board groups these pins logically, by 68HC12 port or peripheral. The ME405 breakout board also supports all accessory components. Figures 10 and 11 are a representation of the ME405 board schematic and PCB layout respectively. Refer to appendix A for a larger representation of the schematic.

Figure 10: ME405 Board Schematic
The headers on the ME405 breakout board can be broken down into four categories: user headers, terminal blocks, jumper blocks and reserved headers. Of the four only the reserved headers are unavailable to the user. Following is a breakdown of all headers on the ME405 breakout board.

**Figure 11: ME405 Board Layout.**
User Headers

User headers come in two types, male and female. The female headers are designed to accept single wire connections. Input/Output (I/O) generally only requires one or two bits. Of course the female headers will accept a full bus connection if it is necessary. The male headers are associated with buses (data bus, address bus, etc.) and generally have to be brought down to the prototyping area as a full bus. A standard 2x5 ribbon cable female connector can be used to bring the buses down. A special adaptor is available from your professor to terminate the ribbon cable at the breadboard. Below is a description all of the user headers. The description starts with the name of the header as it appears on the board, followed by the full name of the header, and then the port function.

**HD_S, Header Serial, (Serial, I/O):** is a 2x4 pin female header. All lines associated with both serial ports available on the 68HC812A4 are present at this header. Header Serial is also available for I/O.

<table>
<thead>
<tr>
<th>Table 2: Header Serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin #</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

**HD_T, Header Timer, (Timer, I/O):** is a 2x4 female header. All lines associated with 68HC812A4 timer port are available at this header. Header Timer is available for I/O.

<table>
<thead>
<tr>
<th>Table 3: Header Timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin #</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>
**HD_J, Header J, (Slow Address Bus, Key Wakeup, I/O):** is a 2x4 female header. This header is associated with port J on the 68HC812A4. Pins 1-4 are reserved for the LCD and keypad addressing. These reserved pins are available at this header for debugging proposes only. Pins 5-8 are free and are available for general I/O. All free bits of port J can be used to generate a key wakeup interrupt.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Bit</th>
<th>Function</th>
<th>Alt. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>LCD RS</td>
<td>--</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>LCD R/W</td>
<td>--</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>LCD E</td>
<td>--</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>KP DR</td>
<td>--</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
</tbody>
</table>

**HD_D, Header D, (I/O, Key Wakeup):** is a 2x4 female header. This header is associated with port D. All pins are available for general I/O. All bits of port D are key wakeup interrupt capable.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Bit</th>
<th>Function</th>
<th>Alt. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>I/O</td>
<td>Key Wakeup</td>
</tr>
</tbody>
</table>
**HD_ADC, Header ADC, (Analog to Digital, Input):** is a 2x5 female header. This header is associated with port ADC on the 68HC812A4 microcontroller; it also includes supporting analog reference inputs $V_{RL}$ and $V_{RH}$. When not needed for analog to digital conversion, pins 1-8 are available as digital inputs.

Note: When using the analog to digital converter DO NOT let the reference voltages $V_{RL}$ and $V_{RH}$ float, they must be tied to appropriate references for proper ADC operation.

<table>
<thead>
<tr>
<th>Table 6: Header ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD_ADC - PORTADC (Analog to Digital)</td>
</tr>
<tr>
<td>Pin #</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

**HD_C, Header C, (Data Bus):** is a 2x5 male header. This header is associated with the 68HC812A4 external 8-bit data bus. Pins 9 and 10 have no contact and should be left floating.

Note: When using the H_C, realize that the user must share the data bus; the external FLASH EEPROM and the keypad decoder are both on the bus.

<table>
<thead>
<tr>
<th>Table 7: Header C</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD_C - Data Bus</td>
</tr>
<tr>
<td>Pin #</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>
**HD_H, Header H, (Slow Data Bus):** is a 2x5 male header. This header is associated with the 68HC812A4 port H. Port H has been designated a slow data bus and is available to the user.

Note: When using the HD_H, realize that the user must share the data bus; the LCD module is on the slow data bus.

<table>
<thead>
<tr>
<th>HD_H - PORTH (Slow Data Bus)</th>
<th>Pin #</th>
<th>Bit</th>
<th>Function</th>
<th>Alt. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>LCD Data 0</td>
<td>Slow Data 0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>LCD Data 1</td>
<td>Slow Data 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>LCD Data 2</td>
<td>Slow Data 2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>LCD Data 3</td>
<td>Slow Data 3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>LCD Data 4</td>
<td>Slow Data 4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>LCD Data 5</td>
<td>Slow Data 5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>LCD Data 6</td>
<td>Slow Data 6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>LCD Data 7</td>
<td>Slow Data 7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>

**HD_ADDR, Header Address Bus, (Address Bus):** is a 2x5 male header. HD_ADDR is associated with the first 10 bits (0-9) of the 68HC812A4 external address bus.

<table>
<thead>
<tr>
<th>HD_ADDR - Addr. Bus[0..9]</th>
<th>Pin #</th>
<th>Bit</th>
<th>Function</th>
<th>Alt. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Addr. 0</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Addr. 1</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Addr. 2</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Addr. 3</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Addr. 4</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Addr. 5</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Addr. 6</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Addr. 7</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>Addr. 8</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Addr. 9</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>
**HD_CTRL, Header Control, (Control, I/O):** is a 2x5 male header. This header contains all bus control lines (chip selects, ECLK, etc.). HD_CTRL also includes external interrupts IRQ, XIRQ and RESET. If chip select 1-3 are not necessary, pins 2-4 are available for general I/O.

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Bit</th>
<th>HD_CTRL (Control) Function</th>
<th>Alt. Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>KP Chip Select</td>
<td>--</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Chip Select 1</td>
<td>I/O</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Chip Select 2</td>
<td>I/O</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Chip Select 3</td>
<td>I/O</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>ECLK</td>
<td>--</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>R/W</td>
<td>--</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>IRQ</td>
<td>--</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>XIRQ</td>
<td>--</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>RESET</td>
<td>--</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
Terminal Blocks
Along the bottom of the ME405 breakout board is a row of screwless terminal blocks. The terminal blocks provide connections between the BNC and power connectors. These terminal blocks provide a convenient connection point between the ME405 breakout board and the prototyping area. Table 10 is a breakdown of the terminal blocks.

<table>
<thead>
<tr>
<th>Terminal Blocks</th>
<th>Block Name</th>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR_1</td>
<td>1</td>
<td>+5 Volts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>PWR_2</td>
<td>1</td>
<td>+10 Volts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Power_1</td>
<td></td>
</tr>
<tr>
<td>PWR_3</td>
<td>1</td>
<td>Power_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Power_3</td>
<td></td>
</tr>
<tr>
<td>BNC_1</td>
<td>1</td>
<td>SIG_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>REF_1</td>
<td></td>
</tr>
<tr>
<td>BNC_2</td>
<td>1</td>
<td>SIG_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>REF_2</td>
<td></td>
</tr>
<tr>
<td>BNC_3</td>
<td>1</td>
<td>SIG_3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>REF_3</td>
<td></td>
</tr>
</tbody>
</table>
**Jumper Blocks**

There are four jumper blocks on the ME405 breakout board. Jumpers JP1 through JP3 connect the reference side of BNC_1 through BNC_3 to ground. In most cases it is desirable to keep the reference side of the BNC cable at ground, but if differential signaling is desired, this jumper can be removed and the reference can be connected to the appropriate source. Jumper JP4 controls the LCD backlight. JP4 is a simple switch, when the jumper is closed the backlight is on, when the jumper is open the LCD backlight is off.
Reserved Headers

In conjunction with the user headers and terminal blocks there are five reserved headers. The reserved headers are along the right side of the ME405 breakout board and connect to components such as the LCD module, keypad, etc. Following is breakdown of the reserved headers.

**HD_BNC, Header BNC:** Header BNC is a 1x6 pin male right angle header. All BNC terminals connect to the breakout board at this point. Care has been exercised routing these signal both on and off the board to avoid analog noise issues.

<table>
<thead>
<tr>
<th>HD_BNC</th>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>SIG_1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>REF_1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SIG_2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>REF_2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>SIG_3</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>REF_3</td>
</tr>
</tbody>
</table>

**HD_KP, Header Keypad:** Header KP is a 2x5 pin male header. All connections to the keypad are made through this header.

<table>
<thead>
<tr>
<th>HD_KP</th>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Y1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>X2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>X3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Y3</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>X4</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>X1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Y4</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Y5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>NC</td>
</tr>
</tbody>
</table>

**HD_LCD, Header LCD:** Header LCD is a 2x7 pin male header. LCD signaling and primary LCD power is routed through the LCD Header.
Table 14: Header LCD

<table>
<thead>
<tr>
<th>HD_LCD</th>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>GND</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>+5V</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>VC</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>RS</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>D1</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>D2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>D3</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>D4</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>D5</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>D7</td>
</tr>
</tbody>
</table>

**HD_BKLT, Header Back Light:** Header Back Light is 1x2 right angle male header. Power is routed though this header for the LCD module’s black light.

Table 15: Header BKLT

<table>
<thead>
<tr>
<th>HD_BKLT</th>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>LCD Back Light Anode</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>LCD Back Light Cathode</td>
</tr>
</tbody>
</table>
**POWER_IN, Power Input**: is a 1x5 right angle male header. All power input is routed through this header.

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+10V</td>
</tr>
<tr>
<td>2</td>
<td>GND_IN</td>
</tr>
<tr>
<td>3</td>
<td>POWER_1</td>
</tr>
<tr>
<td>4</td>
<td>POWER_2</td>
</tr>
<tr>
<td>5</td>
<td>POWER_3</td>
</tr>
</tbody>
</table>

Table 16: Power Input
Prototyping Area

There is a small prototyping area on the ME405 bench top unit. All student projects should be completed using this prototyping space. The proto-board is removable, should a project take multiple days a proto-board will be check out to you for use at any ME405 bench top unit. Figure 12 depicts the proto-board. The red lines in Figure 12 are the direction of common connection on the proto-board.

Figure 12: Proto-Board
Bench Top Unit External Connections

There are three external connection points on the ME405 bench top unit: main power, signal and optional power. Main power and optional power are standard banana jacks; these banana jacks accept banana plugs and are wire crimp capable. The signal connection points are standard BNC jacks. Following is a description of the external connections.

Main Power: is the point at which power is brought to the ME405 bench top unit. The red jack is positive and black is ground. For optimal operation supply the red jack with 10 volts DC. The ME405 bench top unit will operate over the following range 8-35 volts DC. ALWAYS attach the black ground jack before supplying any power to the red jack. Refer to Figure 13.

![Figure 13: Main Power](image)

Optional Power: can be used to bring any other power not already available. The optional power jacks can also be used to strain relive any external I/O. No signal on any optional jack should exceed 40 volts or 1.5 amps. Refer to Figure 14.

Signal: The signal BNC jacks are available for bringing analog signals on or off the ME405 bench top unit. Care has been taken to insure that analog signal run thought these connectors are free from noise. Refer to Figure 14.
Figure 14: Signal and Optional Power
ME405 I/O Libraries

To facilitate student laboratory experiments libraries are provided to help interfacing with the LCD and keypad. These libraries were written here at Cal Poly for the ME405 bench top unit. Refer to appendix B for a source listing of the ME405 libraries. Following is a summary of the I/O functionality provided in the ME 405 library. These commands are in the form of compiled subroutines that have been placed in the ME 405 library file. To make use of these commands that provide interface with the LCD module and the keypad, you must include the ME 405 library in your ICC12 project. To include this library, enter M:/Icc/lib/ME405 in the Additional Library box of the Project>Compiler Options>Target window. Note: in the filename of the library, there is no space between ME and 405.

LCD Module

INITLCD
The LCD module must be initialized before the LCD screen is used. Initialization of the LCD module properly configures pins PJ2-PJ0 of Port J and all of Port H, which are used for communication with the LCD module. Executing INITLCD performs initialization of the LCD module. After successful initialization of the module, a splash screen is displayed for about two seconds. The following instruction executes the INITLCD command:

```
jsr INITLCD ; initialize the LCD
```

OUTSTRING
The OUTSTRING command displays a string of characters on the LCD screen beginning at the LCD address provided. Note that the character string must be terminated by the ASCII null character, $00, which is not displayed. The OUTSTRING command inputs are as follows: accumulator A contains the LCD address and index register X contains the address of the starting character of the string to be displayed.

The following segment of code provides an example in which a string stored beginning at address MESSAGE is displayed:

```
lda #LCDaddr ; LCD address range is $00 - $27 and $40 - $67
ldx #MESSAGE ; starting address of string to be displayed
jsr OUTSTRING
```
The string to be displayed can be generated and stored in memory using assembler directive. ASCII or. ASCIZ if the message is known at assembly time. These assembler directives differ in that ASCIZ automatically appends an ASCII null character to the end of the given string.

```
MESSAGE: .ascii ‘your text to be displayed’
.byte $00 ; append an ASCII null to the end of your message
```

**OUTCHAR_AT and OUTCHAR**

There are two commands for displaying ASCII characters one at a time on the LCD screen. The OUTCHAR_AT command displays a character on the LCD screen at the LCD address provided. The inputs for the OUTCHAR_AT command are as follows: accumulator A contains the LCD address where the character is to be displayed and accumulator B contains the ASCII code for the character to be displayed. The OUTCHAR command displays a character on the LCD screen at the current LCD cursor location, that is, at the current LCD address. The OUTCHAR command input is as follows: accumulator A contains the ASCII code for the character to be displayed. The following segments of code provide an example for using each command.

To output a character at a specific address use:

```
lda #LCDaddr ; LCD address range is from $00 - $27 and $40 - $67
ldab #character ; character is in 8-Bit ASCII
jsr OUTCHAR_AT
```

To output a character at the current LCD address use:

```
lda #character ; character is in 8-Bit ASCII
jsr OUTCHAR
```

**CLRSCREEN**

The clear screen command clears the LCD screen and returns the LCD cursor to the home position, which is LCD address $00. The following instruction executes the CLRSCREEN command:

```
jsr CLRSCREEN
```

**SETADDR**

The set address command sets the cursor to a particular LCD address. The SETADDR command input is as follows: accumulator A contains the address to which the LCD cursor will be set. The following instructions set the cursor to the LCD address LCDaddr:

```
lda #LCDaddr ; LCD address range is from $00 - $27 and $40 - $67
jsr SETADDR
```
**LOAD_ADDR**
The load address command returns the current LCD address. The **LOAD_ADDR** command returns the current LCD address in accumulator A. The following instruction executes the **LOAD_ADDR** command:

```
jsr LOAD_ADDR ; the current LCD address is in accumulator A
```

**CURSOR**
The show cursor command displays a line under the LCD screen location currently addressed by the LCD cursor, which is the location of the next character to be displayed. In order to turn the cursor display off after the **CURSOR** command has been executed, the LCD module must be reinitialized. The following instruction executes the **CURSOR** command:

```
jsr CURSOR
```

**DISPOFF**
The display off command turns the LCD screen off after it has been initialized. To turn the LCD module back on, simply reinitialize the LCD module. The following instruction executes the **DISPOFF** command:

```
jsr DISPOFF
```

**PUTCHAR**
The **PUTCHAR** command provides character output functionality for the high level language C. The **PUTCHAR** command is provided primarily for use in conjunction with the C print function `printf`. 
KEYPAD

INITKEY
The keypad must be initialized before it will function properly. Initialization of the keypad properly configures pin PJ3 of Port J and pin PF0 of Port F, which are used for communication with the keypad. The keypad is interrupt driven, using the key-wakeup interrupt feature of Port J. Upon completion of the INITKEY command, the keypad is in the active state, that is, the keypad is ready to use. The following instruction executes the INITKEY command:

```
jsr INITKEY ; initialize the keypad
```

KP_INACTIVE
This command masks the keypad key-wakeup interrupt, effectively turning off the keypad. The following instruction executes the KP_INACTIVE command:

```
jsr KP_INACTIVE ; turn the keypad off
```

KP_ACTIVE
This command unmasks the keypad key-wakeup interrupt, effectively turning on the keypad. The following instruction executes the KP_ACTIVE command:

```
jsr KP_ACTIVE ; turn the keypad on
```

GETCHAR
The command GETCHAR accepts a single 8-bit ASCII character from the keypad. When the command GETCHAR is executed, the input buffer is checked for a received character. If no character is present, subroutine GETCHAR waits until a character is entered. When a character is received, the 8-bit ASCII code for that character is placed in accumulator B. These non-numeric keys are associated with the following 8-bit ASCII character codes:

- F1 = $F1
- F2 = $F2
- BS = $08
- ENT = $0A

The following instruction executes the GETCHAR command:

```
jsr GETCHAR ; ASCII character is placed in accumulator B
```
Appendix A:

ME405 Bench Top unit Diagram

ME405 Breakout Board Schematic
Figure 15: ME405 Bench Top Unit
Appendix B:

Code
I/O Drivers for Me405 Bench top Module

Date: 11/4/2002

Use: This file contains drivers for 40x2 LCD module and the keypad found on the ME405 Bench top Module

File: ME405_A4_IO.s

Compiler: ImageCraft ICC12 V6

NOTICE: THIS CODE COMES WITHOUT WARRANTY OF ANY KIND EITHER EXPRESSED OR IMPLIED. USE THIS CODE AT YOUR OWN RISK! I WILL NOT BE HELD RESPONSIBLE FOR ANY DAMAGES, DIRECT OR CONSEQUENTIAL THAT YOU MAY EXPERIENCE BY USING IT.

40x2 Character LCD Drivers

Original Author: Terry Cooke

Modified by: Geoff Nichols

Date: 6/9/02

Compiler: ImageCraft ICC12 V6

LCD_Utilites provides these functions in C:

initlcd() --&gt; Initialize the LCD

putchar('character') --&gt; Put character to LCD

clrscreen() --&gt; Clears the LCD

backspc() --&gt; Moves the cursor back one space

setaddr(addr.) --&gt; Sets the LCD addr.

All other functions found in this file can be accessed at the assembly level

Original Section by: Terry Cook

Pin Description:

LCD Address:

Port J (DDRT $00AF)

(PortJ.2) - E, (PortJ.1) - R/W, (PortJ.0) - RS

LCD Data

Port H (DDRH $0025)

(PortH.0 - PortH.7) - DB0-DB7

://: SYSTEM DEFINITIONS AND EQUATES 

;\; Internal Register Definitions

PORTJ  -  $0028

DDRJ  -  $0029

PORTH  -  $0024

DDRH  -  $0025

;\; Application Specific Definitions

LCD_DATA  -  PORTH

LCD_DATA_DIR  -  DDRH

LCD_CTRL  -  PORTJ

LCD_CTRL_DIR  -  DDRJ

E  -  0b000000100 ;Enable Line, bit 2

RW  -  0b00000010 ;Read Write, bit 1

RS  -  0b000000001 ;Register Select, bit 0
Adr_mask        = 0b00000111 ;LCD address bits

.area bss(con)
;;;; RAM VARIABLES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIME:  .blkb 1 ;used for delay time
TEMP:  .blkb 1 ;Temporary Temp Register

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; MAIN ROUTINE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

.area IO_LIB(abs)
.org $FC00

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; INITIALIZE THE LCD ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
_initlcd::
INITLCD:: ;;; Initialize Ports
bclr LCD_CTRL, Adr_mask ;clear LCD_CTRL
clr LCD_DATA ;clear LCD_DATA
movb #$FF, LCD_DATA_DIR ;LCD_DATA output
bset LCD_CTRL_DIR, Adr_mask ;LCD_CTRL output

;; Bits 6,5,4
;; Wait for 15ms
movb #150, TIME ;set delay time
jsr VAR_DELAY ;sub for 0.1ms delay

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

;; Wait for 4.1ms
movb #50, TIME ;set delay time
jsr VAR_DELAY ;sub for 0.1ms delay

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

;; Wait for 100 us
movb #2, TIME ;set delay time
jsr VAR_DELAY ;sub for 0.1ms delay

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

;; Wait for 100 us
movb #2, TIME ;set delay time
jsr VAR_DELAY ;sub for 0.1ms delay

;; 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 Off Command
ldaa #$10 ;function set command
jsr LCD_WRITE ;write data to LCD

;; Send Clear Display
ldaa #$01 ;function set command
jsr LCD_WRITE ;write data to LCD

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

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

;;; SEND INITIALIZATION MESSAGES
;;; Messages have address and content predefined
jsr MESSAGE1 ;send Message1

ldy #25 ;Approximate 2sec Delay to Display
OUTR: ldx #$FFFF ;Driver Info
INNR: dex
lbne INNR
dey
lbne OUTR
JSR CLRSCREEN ;Clear Screen
rts ;Return From Subroutine

; OutString Subroutine (Need LCD addr in ACCA, Text addr in IDX)
OUTSTRING:
adda #$80 ;addr command = $XX+8
staa TEMP ;store LCD address to temp register
clr LCD_CTRL
jsr LCD_ADDR ;send addr to LCD
Loop6: ldxa $00,X ;load AccA w/char from msg
cmpa #00 ;Check for 'BS'
bnez OUTMSG3
jsr LCD_WRITE ;write data to LCD
inx ;increment X
bra Loop6 ;loop to finish msg
OUTMSG3: ldaa TEMP ;Load LCD address from temp register
rts

; OutChar_AT Subroutine outs character at a address (Need LCD addr in ACCA, Text
; in ACCB)
OUTCHAR_AT:: cmpb #$08 ;Check for 'BS'
lbeq BS_HANDLE
adda #$80 ;convert adress value to a adress command
JSR LCD_ADDR ;send addr to LCD
tba ;transfer ACCB -> ACCA
jsr LCD_WRITE ;write data to LCD
rts

; OutChar Subroutine (Need Text in ACCA)
OUTCHAR:: cmpa #$08 ;Check for 'BS'
lbeq BS_HANDLE
clr LCD_CTRL ;clear LCD_CTRL lines
bset LCD_CTRL,R5 ;set the register select bit
jsr LCD_WRITE ;write data to LCD
rts
; Clear Screen and Return Home Subroutine
ClrScreen::
    lda #501 ; load the clearscrean command
    clr LCD_CTRL ; clear LCD_CTRL lines
    jsr LCD_WRITE ; write data to LCD
    rts

; SetAddr Subroutine (Addr in ACCA)
SetAddr::
    tba ; transfer B to A
    adda #$80 ; convert address value to a address command
    clr LCD_CTRL ; clear LCD_CTRL lines
    jsr LCD_ADDR ; send addr to LCD
    rts

; BackSpace Subroutine
backspc::
    jsr LOAD_ADDR ; Load LCD addr
    deca
    adda #$80 ; convert address value to address command
    staa TEMP ; store LCD address to temp register
    jsr LCD_ADDR ; send addr to LCD
    lda #$20 ; load the space ASCII character
    jsr LCD_WRITE ; write blank space to LCD
    lda TEMP ; load the temporarily stored address
    jsr LCD_ADDR ; send addr to LCD
    jsr LOAD_ADDR ; if on line two delete character from
    cmpa #$40 ; line two buffer.
    lbs del_bfr
    exit_bs: rts

; Cursor Subroutine Shows Cursor
Cursor::
    clr LCD_CTRL ; clear LCD_CTRL lines
    lda #50E ; load show cursor command
    jsr LCD_WRITE ; write data to LCD
    rts

; DispOff Subroutine Turns Display Off
DispOff::
    clr LCD_CTRL ; clear LCD_CTRL lines
    lda #$08 ; load ACCA with display off command
    jsr LCD_WRITE ; write data to LCD
    rts

; Routine creates a delay according to the formula
; TIME;~100ms using an 8MHz internal bus
; Cycle count per instruction shown
VAR_DELAY:
    ldab #100 ; 1
Loop1:  nop ; 1
        nop
        nop
        dbne B,Loop1 ; 3
dec TIME ; 4
bne VAR_DELAY ; 3
;;; Routine sends LCD Data

LCD_WRITE:

sta LCD_DATA
bset LCD_CTRL,E ;clock in data
bclr LCD_CTRL,E

;;; Test Busy Flag

ldaa LCD_CTRL
psha

bclr LCD_CTRL,Adr_mask ;set LCD_DATA to input
bset LCD_CTRL,RW

BFTEST:

bset LCD_CTRL,E

ldaa LCD_DATA ;load LCD_DATA to ACCA
bclr LCD_CTRL,E

anda #$80 ;And ACCA with $10000000
bne BFTEST ;Branch Not Equal to Zero

bclr LCD_CTRL,RW

movb #$FF,LCD_DATA_DIR ;set LCD_DATA to output

pula

sta LCD_CTRL
rts

;;; Routine sends LCD Address

LCD_ADDR:

bclr LCD_CTRL,RS ;LCD in command mode

sta LCD_DATA
bset LCD_CTRL,E ;clock in data
bclr LCD_CTRL,E

bset LCD_CTRL,RS ;LCD in data mode

;;; Test Busy Flag

ldaa LCD_CTRL
psha

bclr LCD_CTRL,Adr_mask ;set LCD_DATA to input
bset LCD_CTRL,RW

BFTEST2:

bset LCD_CTRL,E

ldaa LCD_DATA ;load LCD_DATA to ACCA
bclr LCD_CTRL,E

anda #$80 ;And ACCA with $10000000
bne BFTEST2 ;Branch Not Equal to Zero

bclr LCD_CTRL,RW

movb #$FF,LCD_DATA_DIR ;set LCD_DATA to output

pula

sta LCD_CTRL
rts

;;; Routine Loads Last Written Character's Address
;Result is placed in B for 'C' compability

_loadaddr:

clr LCD_CTRL

movb #$500,LCD_DATA_DIR ;set LCD_DATA to input

bset LCD_CTRL,RW

bset LCD_CTRL,E ;Set Enable

ldaa LCD_DATA ;load LCD_DATA to ACCA

bclr LCD_CTRL,E ;Clear Enable

movb #$FF,LCD_DATA_DIR ;set LCD_DATA to output

clr LCD_CTRL ;Clear the Ctrl Register

tab ;transfer A to B

rts
;;; Routine Loads Last Written Character's Address
;Result is placed in A
LOAD_ADDR::
  clr       LCD_CTRL
  movb      #$00,LCD_DATA_DIR   ;set LCD_DATA to input
  bset      LCD_CTRL,RW         ;Set Enable
  ldaa      LCD_DATA            ;load LCD_DATA to ACCA
  bclr      LCD_CTRL,E          ;Clear Enable
  movb      #$FF,LCD_DATA_DIR   ;set LCD_DATA to output
  clr       LCD_CTRL            ;Clear the Ctrl Register
  rts

BS_HANDLE:
  jsr        LOAD_ADDR           ;Load current LCD addr.
  deca
  jsr        SETADDR             ;Set LCD addr.
  rts                            ;Return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Message Routines ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MESSAGE1: ldaa      #$82                ;addr = $02
          jsr       LCD_ADDR            ;send addr to LCD
          ldx       #0
Loop3:   ldaa      MSG1,X              ;load AccA w/char from msg
          beq       OUTMSG1             ;end of msg?
          jsr       LCD_WRITE           ;write data to LCD
          inx                           ;increment X
          bra       Loop3               ;loop to finish msg
OUTMSG1: ldaa      #$C2                ;addr = $42
          jsr       LCD_ADDR            ;send addr to LCD
          ldx       #0
Loop5:   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       Loop5               ;loop to finish msg
OUTMSG2: rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; MESSAGE STORAGE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MSG1:    .asciz      "40x2 LCD Drivers        Date:6/09/02"
MSG2:    .asciz     "By: Terry Cooke and Geoff Nichols"

;end first section
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; putchar addition by: Geoff Nichols
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; RAM VARIABLES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
.line_bfr: .blkb     42
 .bufr_ptr: .blkb     2
 .clear_flg: .blkb     1

;;; Start of Program;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
jsr LOAD_ADDR ;Load the LCD character address and check
cmpa #$40 ;to see if it is >= 40, if so jump to
bhs line2_bufr ;line2_bufr

send_char:
tba ;Transfer acc. B to acc. A. C passes
;the character in acc. B, _OUTCHAR requires
;the character in acc. A

cmpa #$5A ;Check for the linefeed character and
beq linefeed ;branch to service it.
cmpa #00 ;Check for the brake character and
beq exit ;branch to service it.
jsr OUTCHAR ;Branch to OUTCHAR; put character to LCD

exit: rts

linefeed:
jsr LOAD_ADDR ;Load the LCD character address and check
cmpa #$40 ;to see if it is >= 40, if so jump to
bhs linefeed2 ;linefeed2
ldaa #$40 ;If LCD character address is less than 40
jsr SETADDR ;set it to 40(second line) and exit
bra exit

linefeed2:
pshx ;Store modified register
jar CLRSCREEN ;Clear LCD screen
ldaa #$50 ;Address fist location of LCD screen
ldx #line_bfr
jar OUTSTRING ;Put buffered second line on first line
ldaa #$40 ;Address start of second line
jsr SETADDR
ldaa #$FF ;Set clear buffer flag
staa clear_flg
pulx ;Restore modified buffer and exit
bra exit

;Line2 of the LCD must be buffered to allow for LCD scrolling when "newline"
;is encountered.

line2_bufr:
pshx ;Save modified registers
pshd
pshb
ldaa #$fd ;Check to see if buffer has cleared
cmpa clear_flg ;Branch to clear_bfr if clear_flg |= $fd
bne set_ptr
st_char:
ldx bufr_ptr ;Load the buffer pointer into X
pulb
cmpb #$A
beq skip2
stab 0,X ;Store character to line buffer
inx
stx bufr_ptr ;Store the next free buffer location to buffer
;pointer

skip2: puld pulx ;Restore modified
set_ptr:
  ldaa #$fd
  staa clear_flg ;Set clear buffer flag
  ldd #line_bfr
  std bufr_ptr ;Set bufr_ptr to start of line buffer
  ldaa #$00 ;Set line_bfr to all $00
  ldx #line_bfr
loop20:
  staa 0,x
  inx
  cmpx #line_bfr + 41 ;;check for run over/under
  bne loop20
  bra st_char

del_bfr:
  pshx
  ldd bufr_ptr
  decb
  std bufr_ptr
  ldx bufr_ptr
  ldaa #00
  staa 0,x
  pulx
  lbra exit bs

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;4X5 Keypad Driver
;By: Greg Macdonald 6/1/02
;Modified By: Geoff Nichols 6/24/2002
;Use: To be used as an include file in an assembly program or as a C
;     header file.
;File: KEYPADV3_a4_clean.s
;Compiler/Assembler: ImageCraft ICC12 V12
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Pine Description;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;KP_DATA: Data Bus CS0 ($0200 - $027F)
;          PORTD.0 - PORTD.4) -> (KP_DATA.A -KP_DATA.E)
;
; Keypad enable:
; CS0 -> OUTPUT ENABLE
;
;KP_IRQ: PORTJ ($0028)
;         PORTJ.3 -> DATA AVAILABLE [Key wakeup, raising edge]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
;;;;Application Specific Definitions
KP_DATA         =  $0200
KP_IRQ          =  PORTJ
KP_IRQ_DIR      =  DDRJ
IRQ_ACK         =  KWIFJ

KP_DATA_MSK     =  0b00011111
KP_IRQ_MSK      =  0b00001000

;;;;;;;;;;This section will have to be edited depending on what type IRQ used;;;;;;;;;
;;;; Interrupt Vector Location
KP_VECTOR       =  $FFD0                ;PORTJ key wakeup interrupt vector

;;;; Interrupt Vector Setup
.area interrupt_vectors(abs)
  .org   KP_VECTOR
  .word  KEYPAD

;;;; Interrupt Initialization
.area IO_LIB
IRQ_INT:
  BCLR    KP_IRQ,KP_IRQ_MSK
            ;SET KP_IRQ TO INPUT
  BSET    KWIEJ,KP_IRQ_MSK        ;Enable Key Wakeup
  BSET    KWIFJ,KP_IRQ_MSK        ;Set Key Wakeup to Rising
                      ;edge
  RTS

;;;;;;;;;;;;;;;;;;;;;;;;;Lookup Table;;;;;;;;;;;;;;;;;;;;;;;;;

айте 0b00000000
 .byte   $43
 .byte   $38
 .byte   $34
 .byte   $30
 .byte   $44
 .byte   $39
 .byte   $35
 .byte   $31
 .byte   $45
 .byte   $41
 .byte   $36
 .byte   $32
 .byte   $46
 .byte   $42
 .byte   $37
 .byte   $33
 .byte   $0A
 .byte   $01

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

48
.byte $08 ;Backspace 'BS'
.byte 0x00010010
.byte $F2 ;Function 2 'F2'
.byte 0b00010011
.byte $F1 ;Function 1 'F1'
.byte 0b11100000 ;Special table overflow byte
.byte $15 ;ASCII NAK(negative acknowledge) 'error'

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

.area bss

;;;; RAM VARIABLES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHARACTER: .blkb 1 ;Character captured from KP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Main ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

.area IO_LIB

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Initialization Subroutine
_initkey::

INITKEY::

JSR IRQ_INT ;Interrupt Initialization
CLI ;Clear Interrupt Mask
RTS

;;;; Keypad active
_kp_active::

KP_ACTIVE::

BSET KWIEJ,KP_IRQ_MSK ;Enable KP IRQ
RTS

;;;; Keypad inactive
_kp_inactive::

KP_INACTIVE::

BCLR KWIEJ,KP_IRQ_MSK ;Disable KP IRQ
RTS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;INTERRRUPT SERVICE ROUTINE;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

KEYPAD:: ;Get Keypad value

LDAA KP_DATA ;Get Data
ANDA #KP_DATA_MSK ;CLEARS 3 MSBITS LEAVING ONLY VAILD

stab CHARACTER ;Store captured character

49
;Clear IRQ and wait until key is released

SPIN:
  LDA A KP_IRQ
  ANDA #KP_IRQ_MSK
  CMP A #KP_IRQ_MSK
  BEQ SPIN

  BSET IRQ_ACK,KP_IRQ_MSK
  RTI

;---------------------------------------------

;-- getchar() --
;By: Geoff Nichols 7/01/2002
;
;File: getchar.s
;Compiler/Assembler: ImageCraft ICC12 V12
;Use: To be used as an include file in an assembly program or C. "getchar";
; waits for a character to appear in the char_brif. When a new character is
; encountered it is placed it in acc. D with the first 8 bits (acc. A)
; padded with 0's. To call from C "getchar();" to call from assembly
; "getchar"
;---------------------------------------------

.area IO_LIB
_getchar::
GETCHAR::
  LDA B CHARACTER
  CMP B #$FF
  BEQ GETCHAR

  STA A #$00

  MOV B #$FF,CHARACTER
  RTS

;Echo character
_echo::
ECHO::
  CMP B #$0A
  BEQ NO_ECHO

  CMP B #$08
  BEQ NO_ECHO

  CMP B #$F1
  BEQ NO_ECHO

  CMP B #$F2
  BEQ NO_ECHO

  JMP PUTCHAR

NO_ECHO:
  RTS
Appendix C:

Electrical Specifications