LCD

Liquid crystal display
LCD (Liquid crystal display)

- An electrical field is applied to align the crystals in the liquid (a low frequency bipolar signal between 30 Hz to 1KHz)
  - When crystals are aligned, light can pass through
  - The point looks dark because behind the crystals is a black backing which absorbs light

- Advantages over seven segment displays:
  - Can display many characters in a compact footprint
  - Can display graphics
  - Low power consumption

Figure 7.26 A liquid crystal display (LCD)
Interfacing to an LCD

- An LCD is a complex device and there are many parameters to deal with.
- Example
  - What character to write
  - The bit pattern to use to display the character
  - Where on the screen the character is written to
  - Control functions such as clearing the display
  - Options such as whether the cursor is visible, or should blink
- One option would be to have many pins, each of which controls some function
  - However, this is not desirable for small embedded systems
  - MCUs are usually very pin-limited
- Instead, the LCD has its own controller that the MCU talks to
  - Commands and data are sent using relatively few pins (good)
  - Need to define the command interface and protocol (difficult)
- This approach is fairly typical of embedded systems
  - Allows the MCU to control some fairly complex devices
LCD controller

- LCDs are often sold in a module with a controller unit built in.
- The Hitachi HD44780 is the most popular character-based LCD controller.
- The controller defines a 14-pin control interface, along with the protocol for passing commands and data.
- Can handle different matrix sizes (e.g., 1x8, 2x16, 4x20, ...).
Hitachi 44870 LCD controller

- The controller has display data RAM to store the characters
- Each character is 8 bits, so the RAM for an 80 character display has 80x8 bits

DDRAM address:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>0A</td>
<td>0B</td>
<td>0C</td>
<td>0D</td>
<td>0E</td>
<td>0F</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>4A</td>
<td>4B</td>
<td>4C</td>
<td>4D</td>
<td>4E</td>
<td>4F</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>1A</td>
<td>1B</td>
<td>1C</td>
<td>1D</td>
<td>1E</td>
<td>1F</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>5A</td>
<td>5B</td>
<td>5C</td>
<td>5D</td>
<td>5E</td>
<td>5F</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
</tr>
</tbody>
</table>

Display position

Layout of a 4x20 character LCD display
**LCD controller**

- The controller has a ROM to generate the dot patterns for each character (5x8 or 5x10)
  - It is preloaded with the dot patterns for all the ASCII characters
  - You can change and add your own dot patterns

The ASCII code for the letter “A” is 0x41, or 0100 0001, or LHLL LLLH
Control Interface

• Inside the controller there is:
  – An 8-bit instruction register
  – An 8-bit data register

• You write to these registers to control the LCD
  – Instructions are commands like ‘clear the display’
  – Data values are the ASCII codes for the characters to be displayed
  – The signal that determines whether the code is a command or an instruction is the RS pin (pin 4)
    • RS = 0 means that the code is an instruction
    • RS = 1 means that the code is a data value
  – The 8 bit code can be sent as
    • 8 bits in parallel, using 8 pins, or
    • Two sets of 4 bits (nibbles) – the ‘most significant nibble’ is sent first
      – This is slower but saves on pin count ... need only 4 pins instead of 8
Pin assignment

- Vcc, Vss are power and ground
- Vee is a voltage input to control the contrast
  - Typical value is around 0.5V
  - You can use a little potentiometer ("trimpot") to adjust it to get maximum contrast

We only need a minimum of 6 pins to control the LCD:
- RS (tells LCD if we are sending instruction or data)
- E (enable, used to clock in instruction or data)
- DB4:DB7 (4 bit nibble)
- Just tie R/W = 0 because we will only write to the LCD

Table 7.3 Pin assignment for displays with less than 80 characters

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>symbol</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VSS</td>
<td>-</td>
<td>Power supply (GND)</td>
</tr>
<tr>
<td>2</td>
<td>VCC</td>
<td>-</td>
<td>Power supply (+5 V)</td>
</tr>
<tr>
<td>3</td>
<td>VEE</td>
<td>-</td>
<td>Contrast adjust</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>I</td>
<td>0 = instruction input, 1 = data input</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>I</td>
<td>0 = write to LCD, 1 = read from LCD</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>I</td>
<td>Enable signal</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
<td>I/O</td>
<td>Data bus line 0</td>
</tr>
<tr>
<td>8</td>
<td>DB1</td>
<td>I/O</td>
<td>Data bus line 1</td>
</tr>
<tr>
<td>9</td>
<td>DB2</td>
<td>I/O</td>
<td>Data bus line 2</td>
</tr>
<tr>
<td>10</td>
<td>DB3</td>
<td>I/O</td>
<td>Data bus line 3</td>
</tr>
<tr>
<td>11</td>
<td>DB4</td>
<td>I/O</td>
<td>Data bus line 4</td>
</tr>
<tr>
<td>12</td>
<td>DB5</td>
<td>I/O</td>
<td>Data bus line 5</td>
</tr>
<tr>
<td>13</td>
<td>DB6</td>
<td>I/O</td>
<td>Data bus line 6</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
<td>I/O</td>
<td>Data bus line 7</td>
</tr>
</tbody>
</table>
Timing

- Nibbles are clocked into the LCD with a falling edge on E (pin 6)
- Nibble data must be present on DB4:DB7 (pins 11:14)

Table 7.3 Pin assignment for displays with less than 80 characters

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>symbol</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V\textsubscript{SS}</td>
<td>-</td>
<td>Power supply (GND)</td>
</tr>
<tr>
<td>2</td>
<td>V\textsubscript{CC}</td>
<td>-</td>
<td>Power supply (+5 V)</td>
</tr>
<tr>
<td>3</td>
<td>V\textsubscript{EE}</td>
<td>-</td>
<td>Contrast adjust</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>I</td>
<td>0 = instruction input, 1 = data input</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>I</td>
<td>0 = write to LCD, 1 = read from LCD</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>I</td>
<td>Enable signal</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
<td>I/O</td>
<td>Data bus line 0</td>
</tr>
<tr>
<td>8</td>
<td>DB1</td>
<td>I/O</td>
<td>Data bus line 1</td>
</tr>
<tr>
<td>9</td>
<td>DB2</td>
<td>I/O</td>
<td>Data bus line 2</td>
</tr>
<tr>
<td>10</td>
<td>DB3</td>
<td>I/O</td>
<td>Data bus line 3</td>
</tr>
<tr>
<td>11</td>
<td>DB4</td>
<td>I/O</td>
<td>Data bus line 4</td>
</tr>
<tr>
<td>12</td>
<td>DB5</td>
<td>I/O</td>
<td>Data bus line 5</td>
</tr>
<tr>
<td>13</td>
<td>DB6</td>
<td>I/O</td>
<td>Data bus line 6</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
<td>I/O</td>
<td>Data bus line 7</td>
</tr>
</tbody>
</table>

- E pulse width must be at least 230 ns
- Must have cycle time (time between writes) no shorter than 500 ns

Figure 7.30 HD44780U LCD controller write timing diagram
Procedure to write a nibble to the IR register:
- Put the nibble on pins DB4:DB7, make RS low to signify an instruction
- Pull the E signal high
- Wait at least 230 ns
- Pull E low to clock in the nibble
- Wait at least 270 ns before sending any more nibbles
  - An alternative to waiting is to test the busy flag (DB7)
  - But this requires an additional pin (R/W) to specify a read vs write

Table 7.11 HD44780U bus timing parameters (2 MHz operation)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Min</th>
<th>Typ</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCYCLE</td>
<td>Enable cycle time</td>
<td>500</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>PW_{EH}</td>
<td>Enable pulse width (high level)</td>
<td>230</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>t_{Er}, t_{Ef}</td>
<td>Enable rise and decay time</td>
<td>-</td>
<td>-</td>
<td>20</td>
<td>ns</td>
</tr>
<tr>
<td>tAS</td>
<td>Address setup time, RS, R/\overline{W}, E</td>
<td>40</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>tDDR</td>
<td>Data delay time</td>
<td>-</td>
<td>-</td>
<td>160</td>
<td>ns</td>
</tr>
<tr>
<td>tDSW</td>
<td>Data setup time</td>
<td>80</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>tH</td>
<td>Data hold time (write)</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>tDHR</td>
<td>Data hold time (read)</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>tAH</td>
<td>Address hold time</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
</tbody>
</table>
Connections - Example

• Use Port T to connect to LCD

• Need to set up PT0:PT5 for output

• Other LCD connections (not shown):
  – LCD pin 1:  ground
  – LCD pin 2:  +5 V
  – LCD pin 3:  contrast adjust (~0.5 volts)
  – LCD pin 5:  R/W = 0 (tie to ground), specifies write
  – LCD pins 7:10:  DB0:DB3, not used
A program module to send a nibble

• Let input parameters be
  – “n”: the nibble, in the lower 4 bits of an 8-bit byte
  – “rs”: the indicator whether nibble is instruction (0) or data (1)

\[
\begin{array}{cccccc}
\text{n} & \text{rs} \\
- & - & - & - & n_3 & n_2 & n_1 & n_0 \\
- & - & - & - & - & - & - & rs \\
\end{array}
\]

• We need to put these into Port T

\[
\begin{array}{cccccc}
\text{PT} & \text{T} \\
- & - & rs & E & n_3 & n_2 & n_1 & n_0 \\
\end{array}
\]

• First make E=1, then make E=0 to clock in nibble
A program module to send a nibble

Start – input parameters are n, rs, t

Shift rs left 5 places to put rs into bit 5 position

“OR” the shifted rs with n; also “OR” with E=1 in the bit 4 position

Write this to PTT

Wait a short time (>230 ns)

Clear bit 4 of PTT to make E=0; this clocks in data

Wait t microseconds

return

n
\[ \begin{array}{ccccccc}
  & & & & & & n_0 \\
  & & & & & n_3 & n_2 \\
  & & & & n_1 & & \\
  & & & n_0 & & & \\
\end{array} \]

rs
\[ \begin{array}{cccccc}
  & & & & & rs \\
  & & & & & \\
  & & & & & \\
  & & & & & \\
\end{array} \]

PTT
\[ \begin{array}{cccccccc}
  & & & & & rs & E & n_3 & n_2 & n_1 & n_0 \\
  & & & & & & & & & & \\
  & & & & & & & & & & \\
\end{array} \]

- The minimum wait time is 270 ns; however some commands take a long time to execute
- So it will be helpful to have a general module that waits a specified amount of time
C function to write a nibble

/* This function writes a nibble to the LCD. Input parameters:
   n contains the nibble to be written (in the lower 4 bits)
   rs indicates instruction or data (rs=0 for inst, rs=1 for data)
   t is the time to delay after sending (units of 1 us)
Assumes these connections:
PT0:PT3 - connect to LCD pins 8:14 (DB4:DB7)
PT4 - connect to LCD pin 6 (E)
PT5 - connect to LCD pin 4 (RS)
*/
void writeNibbleToLCD(char n, char rs, int t) {
    rs <<= 5;                  // get rs bit into the bit 5 position
    PTT = rs|0x10|(0x0f & n);  // output the nibble with E=1
    DelayuSec(1);              // keep E pulse high a little while
    PTT &= ~0x10;              // make E go to 0
    DelayuSec(t);
}

• This calls a delay function that delays for the specified number of microseconds
## Instructions

1. Most instructions are just performed once when we initialize the LCD.

2. After setting up, the only one we need is “Clear display”.

---

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
<th>Code</th>
<th>Description</th>
<th>Execution Time (max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear display</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 0 0 0 0 0 0 0 0 0 0</td>
<td>Clears entire display and sets DDRAM address 0 in address counter.</td>
<td>1.52 ms</td>
</tr>
<tr>
<td>Return home</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>—</td>
<td>0 0 0 0 0 0 0 0 0 1</td>
<td>Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.</td>
<td>1.52 ms</td>
</tr>
<tr>
<td>Entry mode set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I/D</td>
<td>S</td>
<td>0 0 0 0 0 0 0 0 1</td>
<td>Sets cursor move direction and specifies display shift. These operations are performed during data write and read.</td>
<td>37 μs</td>
</tr>
<tr>
<td>Display on/off control</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>0 0 0 0 0 0 0 1 D C B</td>
<td>Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).</td>
<td>37 μs</td>
</tr>
<tr>
<td>Cursor or display shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>S/C</td>
<td>R/L</td>
<td>—</td>
<td>—</td>
<td>0 0 0 0 0 0 0 1 S/C R/L</td>
<td>Moves cursor and shifts display without changing DDRAM contents.</td>
<td>37 μs</td>
</tr>
<tr>
<td>Function set</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DL</td>
<td>N</td>
<td>F</td>
<td>—</td>
<td>—</td>
<td>0 0 0 0 1 DL N F —</td>
<td>Sets interface data length (DL), number of display lines (N), and character font (F).</td>
<td>37 μs</td>
</tr>
<tr>
<td>Set CGRAM address</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>0 0 0 1 ACG ACG ACG ACG ACG</td>
<td>Sets CGRAM address. CGRAM data is sent and received after this setting.</td>
<td>37 μs</td>
</tr>
<tr>
<td>Set DDRAM address</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>0 0 1 ADD ADD ADD ADD ADD ADD</td>
<td>Sets DDRAM address. DDRAM data is sent and received after this setting.</td>
<td>37 μs</td>
</tr>
<tr>
<td>Read busy flag &amp; address</td>
<td>0</td>
<td>1</td>
<td>BF</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>0 1 BF AC AC AC AC AC AC</td>
<td>Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.</td>
<td>0 μs</td>
</tr>
</tbody>
</table>

---

*from HD44780U datasheet*
Instructions (continued)

<table>
<thead>
<tr>
<th>I/D</th>
<th>I/D = 1: Increment</th>
<th>I/D = 0: Decrement</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S = 1: Accompanies display shift</td>
<td>S/C = 1: Display shift</td>
</tr>
<tr>
<td>S/C</td>
<td>S/C = 0: Cursor move</td>
<td></td>
</tr>
<tr>
<td>R/L</td>
<td>R/L = 1: Shift to the right</td>
<td>R/L = 0: Shift to the left</td>
</tr>
<tr>
<td>DL</td>
<td>DL = 1: 8 bits, DL = 0: 4 bits</td>
<td>N = 1: 2 lines, N = 0: 1 line</td>
</tr>
<tr>
<td>N</td>
<td>F = 1: 5 x 10 dots, F = 0: 5 x 8 dots</td>
<td>BF = 1: Internally operating</td>
</tr>
<tr>
<td>BF</td>
<td>BF = 0: Instructions acceptable</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**From HD44780U datasheet**

- DDRAM: Display data RAM
- CGRAM: Character generator RAM
- ACG: CGRAM address
- ADD: DDRAM address (corresponds to cursor address)
- AC: Address counter used for both DD and CGRAM addresses
- Execution time changes when frequency changes

Example:

- When $f_{op}$ or $f_{cde}$ is 250 kHz,
- $37 \mu s \times \frac{270}{250} = 40 \mu s$
Initialization

• We’ll set up the LCD to act like a dumb terminal
  – Namely, after receiving and displaying a character, the cursor shifts to the right one place
  – Some other options we will choose:
    • Choose a 4-bit (nibble) interface, where data and instructions are sent as a sequence of two nibbles
    • Don’t display the cursor

• We’ll write a C function called “InitializeLCD()” that will perform all initialization
  – Note: some instructions are pretty slow, like “clear display” (1.5 ms)
  – We’ll use a delay function where we pass in the number of microseconds to delay
### Initialization Sequence

1. **Power on**
   - Wait for more than 15 ms after $V_{CC}$ rises to 4.5 V

2. **RS R/W DB7 DB6 DB5 DB4**
   - 0 0 0 0 1 1

3. **Wait for more than 4.1 ms**

4. **RS R/W DB7 DB6 DB5 DB4**
   - 0 0 0 0 1 1

5. **Wait for more than 100 $\mu$s**

6. **RS R/W DB7 DB6 DB5 DB4**
   - 0 0 0 0 1 1

7. **RS R/W DB7 DB6 DB5 DB4**
   - 0 0 0 0 1 0
   - 0 0 0 0 1 0
   - 0 0 N F + +
   - 0 0 0 0 0 0
   - 0 1 0 0 0 0
   - 0 0 0 0 0 0
   - 0 0 0 0 0 1
   - 0 0 0 0 0 0
   - 0 0 0 1 I/D S

   **Initialization ends**

**From HD44780U datasheet**

---

**Figure 26 4-Bit Interface**
Initialization Sequence

• Wait 50 ms for LCD to power up

• Send the instruction nibbles 0x03, 0x03, 0x03, 0x02
  – Wait 5 ms after each nibble
  – Not sure why we have to send this many, but it’s called for in the datasheet

• Send the following instruction byte: 0x28
  – Each byte is sent as two nibbles (high nibble first)
  – Wait 50 microseconds to allow instruction to execute

<table>
<thead>
<tr>
<th>0x28</th>
<th>0 0 1 0 1 0 0 0 0</th>
</tr>
</thead>
</table>

  0 0 1 DL N F - -

| DL = 1: 8 bit mode |
| DL = 0: 4 bit mode |

| N = 1: 2 lines |
| N = 0: 1 line |

| F = 1: 5x10 dots |
| F = 0: 5x8 dots |

• So this sets up the LCD for 4 bit communications (i.e., nibbles), 2 lines, and 5x8 dots
Initialization Sequence (continued)

• Send the following additional instruction bytes: 0x0c, 0x14
  – Each byte is sent as two nibbles (high nibble first)
  – Wait 50 microseconds to allow each instruction to execute
  – What do these instructions do?

• Finally, set the instruction byte 0x01
  – This clears the display and sends the cursor home
  – This is a slow instruction, so wait 2 ms to allow it to execute
C function to Initialize LCD

```c
void InitializeLCD(void) {
    int i;

    for (i=0; i<100; i++) // delay 100ms to allow LCD powerup
        DelayuSec(1000);

    // The first parameter in each call is the nibble to be sent,
    // the second parameter is the rs value (rs=0 indicates an instruction),
    // and the third parameter is the time to delay after sending (in units of us).
    writeNibbleToLCD(0x03, 0, 5000);   // delay at least 4 ms = 4000 us
    writeNibbleToLCD(0x03, 0, 5000);
    writeNibbleToLCD(0x03, 0, 5000);
    writeNibbleToLCD(0x02, 0, 5000);

    // The first parameter in each call is the byte to be sent (as two nibbles),
    // the second parameter is the rs value (rs=0 indicates an instruction),
    // and the 3\textsuperscript{rd} parameter is the time to delay after sending both nibbles (usec).
    // These commands are all fast (~40 us) except for "clear display" (2 ms)
    writeByteToLCD(0x28, 0, 50);    // 2 lines, 5x8 dots
    writeByteToLCD(0x0c, 0, 50);    // display on, no cursor, no blinking
    writeByteToLCD(0x14, 0, 50);    // shift cursor right
    writeByteToLCD(0x01, 0, 2000);  // clear display and cursor home
}
```

- **writeByteToLCD** – this function must call **writeNibbleToLCD** twice, first for the high nibble, then for low nibble, then wait the desired delay time
Sending Data to LCD

- To send data, make RS=1
- Note that a write (or read) takes ~40 us to execute

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
<th>Description</th>
<th>Execution Time (max) (when $f_{cp}$ or $f_{osc}$ is 270 kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write data to CG or DDRAM</td>
<td>1 0</td>
<td>Write data</td>
<td>37 $\mu$s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$t_{ADD} = 4 \mu$s*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Writes data into DDRAM or CGRAM.</td>
<td></td>
</tr>
<tr>
<td>Read data from CG or DDRAM</td>
<td>1 1</td>
<td>Read data</td>
<td>37 $\mu$s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$t_{ADD} = 4 \mu$s*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reads data from DDRAM or CGRAM.</td>
<td></td>
</tr>
</tbody>
</table>

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.
Sending a character string to the LCD

- We will need a C function that will send a character string to the LCD
  - The LCD will display a maximum of 80 characters

- Recall that a string is just an array of “char”
  - C automatically terminates the string with a NUL character (i.e., the value 0)

- Should delay ~50 us after each character

```c
function printLCD(char mystr[ ])
{
    send instruction to clear display

    i = 0

    if i < 80
    {
        ch = mystr[i]
        if ch = 0
        {
            send data byte ch to LCD
            i = i + 1
        }
    }
    return
}
```
Summary / Questions

• What are the advantages and disadvantages of using an LCD display versus say, a seven segment display?

• The LCD controller has a complicated interface, in terms of the protocol for sending commands and data. What’s the benefit of doing this versus having a simpler interface where each pin controls a function?