Alphanumeric LCD Displays

55:036
Embedded Systems and Systems Software

The HD44780 LCD Controller
• Most low cost Character-based LCD modules use the Hitachi HD44780 controller chip
  – Typically 8, 16, 20, 24 or 40 characters/line
  – 1, 2, or 4 lines
  – Handles up to $2^7 = 128$ total characters/display
• Standard 14-pin interface

Alphanumeric LCDs

CHARACTER LCD Screens

Alphanumeric LCDs

LCD Pinouts

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Symbol</th>
<th>Level</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vss</td>
<td>-</td>
<td>I/O</td>
<td>Power supply (GND)</td>
</tr>
<tr>
<td>2</td>
<td>Vcc</td>
<td>-</td>
<td>I/O</td>
<td>Power supply (+5V)</td>
</tr>
<tr>
<td>3</td>
<td>R/W</td>
<td>-</td>
<td>I/O</td>
<td>Instruction input</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>0/1</td>
<td>I/O</td>
<td>Data input</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>0/1</td>
<td>I/O</td>
<td>Write to LCD module</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>0/0</td>
<td>I/O</td>
<td>Read from LCD module</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 0 (LSB)</td>
</tr>
<tr>
<td>8</td>
<td>DB1</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 1</td>
</tr>
<tr>
<td>9</td>
<td>DB2</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 2</td>
</tr>
<tr>
<td>10</td>
<td>DB3</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 3</td>
</tr>
<tr>
<td>11</td>
<td>DB4</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 4</td>
</tr>
<tr>
<td>12</td>
<td>DB5</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 5</td>
</tr>
<tr>
<td>13</td>
<td>DB6</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 6</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
<td>0/1</td>
<td>I/O</td>
<td>Data bus line 7 (MSB)</td>
</tr>
</tbody>
</table>
LCD Interface Modes

- **8 bit mode**
  - Uses all 8 data lines DB0-DB7
  - Data transferred to LCD in byte units
  - Interface requires 10 (sometimes 11) I/O pins of microcontroller (DB0-DB7, RS, E) (sometimes R/W)

- **4-bit mode**
  - 4-bit (nibble) data transfer
  - Doesn’t use DB0-DB3
  - Each byte transfer is done in two steps: high order nibble, then low order nibble
  - Interface requires only 6 (sometimes 7) I/O pins of microcontroller (DD4-DB7, RS, E) (sometimes R/W)

QwikFlash uses 4-bit interface mode

QwikFlash LCD Configuration

LCD Control: RS, E, R/W

- **RS (Register Select)**
  - When low: data transferred to/from device is treated as commands (status)
  - When high: data transferred to/from device is characters.

- **R/W (Read/Write)**
  - Controls data transfer direction
    - low to write to LCD
    - high to read from LCD
  - On the QwikFlash, this pin is wired to ground—i.e. can’t read from LCD

- **E (Enable) Input**
  - Initiates data transfer
  - For write, data transferred to LCD on high to low transition
  - For read, data available following low to high transition
At least 40 nsec

At least 230 nsec

At least 10 nsec

At least 80 nsec.
### LCD Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Cycle Time</td>
<td>400</td>
<td>200</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>Enable Rise/Fall Time</td>
<td>400</td>
<td>200</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>Enable Pulse Width (High)</td>
<td>400</td>
<td>200</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>Enable Pulse Width (Low)</td>
<td>400</td>
<td>200</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>Address Cycle Time</td>
<td>400</td>
<td>200</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>Address Hold Time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>ns</td>
</tr>
<tr>
<td>Data Delay Time</td>
<td>100</td>
<td>80</td>
<td>80</td>
<td>ns</td>
</tr>
<tr>
<td>Data Hold Time</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>ns</td>
</tr>
</tbody>
</table>

Unit: Max (1) Typ (1) Min (1)

### LCD Commands

#### LCD Command Execution Times

- Most HD44780 commands take 40 microseconds to execute.
- Clear Display and Cursor Home commands can take much longer (as much at several milliseconds).
- Can’t issue another command until previous one has finished.

#### Command Execution Times—Continued

- Two options:
  - Busy-wait:
    - After issuing a command, continuously monitor HD44780 status until device is not busy.
    - Can’t do this with QwikFlash, since we can’t read from the HD44780.
  - Insert a 40 microsecond (or, in some cases, much longer) delay between commands.
Command Execution Times--Continued

- Most HD44780 commands take 40 microseconds to execute
- Clear Display and Cursor Home commands can take much longer (as much as several milliseconds)
- Can’t issue another command until previous one has finished
- Two options
  - Busy-wait:
    - After issuing a command, continuously monitor HD44780 status until device is not busy
    - Can’t do this with QwikFlash, since we can’t read from the HD44780
  - Insert a 40 microsecond (or, in some cases, much longer) delay between commands

More about Timing

- Timing for writing to the LCD is not critical (as long as setup and hold times are observed):
  - Drive E high (pin RE1)
  - Send upper “nibble” of data to Port D (RD7-RD4)
  - Drive E low (pin RE1)
  - Drive E high (pin RE1)
  - Send lower nibble of data to Port D (RD7-RD4)
  - Drive E low (pin RE1)
- Note: Throughout this process RS must be properly set (low for commands; high for characters)

More about Timing

- Timing for writing to the LCD is not critical (as long as setup and hold times are observed):
  - Drive E high (pin RE1)
  - Send upper “nibble” of data to Port D (RD7-RD4)
  - Drive E low (pin RE1)
  - Drive E high (pin RE1)
  - Send lower nibble of data to Port D (RD7-RD4)
  - Drive E low (pin RE1)
- Note: Throughout this process RS must be properly set (low for commands; high for characters)
Writing to the LCD--Example

;This example code writes the byte stored in SFR TABLAT to the LCD

bsf PORTE,RE1              ;Drive E pin high
movff TABLAT,PORTD   ;Send upper nibble
bcf PORTE,RE1              ;Drive E pin low so LCD will
;accept nibble
bsf PORTE,RE1              ;Drive E pin high again
swapf TABLAT,W           ;Swap nibbles
movwf PORTD               ;Write lower nibble
bcf PORTE,RE1             ;Drive E pin low so LCD will
;process byte

The Cursor Position Map for the QwikFlash LCD

<table>
<thead>
<tr>
<th>0x0</th>
<th>0x1</th>
<th>0x2</th>
<th>0x3</th>
<th>0x4</th>
<th>0x5</th>
<th>0x6</th>
<th>0x7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40</td>
<td>0x41</td>
<td>0x42</td>
<td>0x43</td>
<td>0x44</td>
<td>0x45</td>
<td>0x46</td>
<td>0x47</td>
</tr>
</tbody>
</table>

HD44780 Character Codes

```
<table>
<thead>
<tr>
<th>0x00</th>
<th>0x01</th>
<th>0x02</th>
<th>0x03</th>
<th>0x04</th>
<th>0x05</th>
<th>0x06</th>
<th>0x07</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

To “write” a string of characters to the LCD

- Drive RS low (command mode)
- Send a **Set Display Address** command to the LCD to establish initial display position
- Drive RS high (character mode)
- Send first character to LCD
- Send second character to LCD
- etc.

"Degrees" symbol
To “write” a string of characters to the LCD

- Drive RS low (command mode)
- Send a Set Display Address command to the LCD to establish initial display position
- Drive RS high (character mode)
- Send first character to LCD
- Send second character to LCD
- etc.

The display position will automatically increment (or decrement) depending upon how you configured the LCD with the Character Entry Command.

Note: Need to wait 40 microseconds between each character.
Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte

Sending Characters to the LCD

- `bfr PORTE,RE0` : Drive RS pin low for cursor positioning command
- `bsf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `bsf PORTE,RE1` : Drive E pin high again
- `movlf 0x00,PORTD` : Send lower nibble of set address command
- `bcf PORTE,RE1` : Drive E pin low so LCD will process byte
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE0` : Drive RS pin high for displayable characters
- `bf PORTE,RE1` : Drive E pin high
- `movlf 0x80,PORTD` : Send upper nibble of Character ‘H’
- `bf PORTE,RE1` : Drive E pin low so LCD will accept nibble
- `movlf 0x00,PORTD` : Send lower nibble of Character ‘H’
- `rcall T40` : Wait 40 usec
- `bf PORTE,RE1` : Drive E pin low so LCD will process byte
Sending Characters to the LCD

\begin{verbatim}
bcf PORTE,RE0        ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x80,PORTD     ;Send upper nibble of set address command
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x00, PORTD    ;Send lower nibble of set address command
rcall T40                     ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin high for displayable characters
bsf PORTE,RE1       ;Drive E pin high
MOVLF 0x40,PORTD ;Send upper nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x80, PORTD ;Send lower nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                    ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x60,PORTD ;Send upper nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x90, PORTD ;Send lower nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                     ;Wait 40 usec
\end{verbatim}

Sending Characters to the LCD

\begin{verbatim}
bcf PORTE,RE0        ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x80,PORTD     ;Send upper nibble of set address command
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x00, PORTD    ;Send lower nibble of set address command
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                     ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin high for displayable characters
bsf PORTE,RE1       ;Drive E pin high
MOVLF 0x40,PORTD ;Send upper nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x80, PORTD ;Send lower nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                    ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x60,PORTD ;Send upper nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x90, PORTD ;Send lower nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                     ;Wait 40 usec
\end{verbatim}

Sending Characters to the LCD

\begin{verbatim}
bcf PORTE,RE0        ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x80,PORTD     ;Send upper nibble of set address command
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x00, PORTD    ;Send lower nibble of set address command
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                     ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin high for displayable characters
bsf PORTE,RE1       ;Drive E pin high
MOVLF 0x40,PORTD ;Send upper nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x80, PORTD ;Send lower nibble of Character 'H'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                    ;Wait 40 usec
bcf PORTE,RE0     ;Drive RS pin low for cursor positioning command
bsf PORTE,RE1        ;Drive E pin high
MOVLF 0x60,PORTD ;Send upper nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1       ;Drive E pin high again
MOVLF 0x90, PORTD ;Send lower nibble of character 'i'
bcf PORTE,RE1       ;Drive E pin low so LCD will process byte
rcall T40                     ;Wait 40 usec
\end{verbatim}

The DisplayC Subroutine

- Displays a constant character string (stored in program memory)
  - First byte of string contains the cursor-positioning command
  - Last byte of string is 0x00
  - Intervening bytes contain codes for characters to be displayed
- E.g.:
  \begin{verbatim}
  MYSTR db "\x80Hello\x00"
  or, equivalently
  MYSTR db 0x80,0x48, 0x65,0x6c,0x6f,0x00
  \end{verbatim}
Using TBLPTR to Access a Constant String in Program Memory

- Must make TBLPTRH:TBLPTRL “point at” the string:

```assembly
MOVLF high MYSTR, TBLPTRH
MOVLF low MYSTR, TBLPTRL
```

Now can read bytes from the string:

- `tblrd*`: reads byte pointed to by TBLPTR into TABLAT
- `tblrd++`: increments TBLPTR and reads byte pointed to by TBLPTR into TABLAT

Note: can also do: `tblptr*+, tblptr*-`

The DisplayC Subroutine

Before subroutine is called, TBLPTRH:TBLPTRL must contain the address of the first byte of the string.

```assembly
bcf PORTE,RE0       ;Drive RS pin low for cursor-positioning command
tblrd*                          ;Get byte from string into TABLAT
movf TABLAT,F        ;Check for leading zero byte
IF_  .Z.              ;If zero, get next byte
ENDIF_                
REPEAT_        ;Loop until 0x00 byte is encountered
bsf PORTE,RE1          ;Drive E pin high
movff TABLAT,PORTD  ;Send upper nibble
bcf PORTE,RE1          ;Drive E pin low so LCD will accept nibble
swapf TABLAT,W        ;Swap nibbles
movff PORTD,PORTD     ;Write lower nibble
rcall T40             ;Wait 40 usec
bsf PORTE,RE0         ;Drive RS pin high for displayable characters
tblrd+*                             ;Increment pointer, then get next byte
movf TABLAT,F             ;Is it zero?
UNTIL_  .Z.               ;Return
```

The DisplayC Subroutine

```assembly
bcf PORTE,RE0          ;Drive RS pin low for cursor-positioning command
movf TABLAT,F        ;Check for leading zero byte
IF_  .Z.              ;If zero, get next byte
ENDIF_                
REPEAT_        ;Loop until 0x00 byte is encountered
bcf PORTE,RE1          ;Drive E pin high
movff TABLAT,PORTD  ;Send upper nibble
bcf PORTE,RE1          ;Drive E pin low so LCD will accept nibble
swapf TABLAT,W        ;Swap nibbles
movff PORTD,PORTD     ;Write lower nibble
rcall T40             ;Wait 40 usec
bsf PORTE,RE0         ;Drive RS pin high for displayable characters
tblrd+*                             ;Increment pointer, then get next byte
movf TABLAT,F             ;Is it zero?
UNTIL_  .Z.               ;Return
```
The DisplayV Subroutine

- Displays a variable string stored in data memory
- String format is same as for DisplayC
  - First byte of string contains the cursor-positioning command
  - Last byte of string is 0x00
  - Intervening bytes contain codes for characters to be displayed
This subroutine is called with FSR0 containing the address of a variable display string. It sends the bytes of the string to the LCD. The first byte sets the cursor position. The remaining bytes are displayed, beginning at that position.

DisplayV

bcf PORTE,RE0 ;Drive RS pin low for cursor positioning code
REPEAT
bsf PORTE,RE1 ;Drive E pin high
movf INDF0,PORTD ;Send upper nibble
bsf PORTE,RE1 ;Drive E pin low so LCD will accept nibble
bsf PORTE,RE1 ;Drive E pin high again
swapf INDF0,W ;Swap nibbles
movf PORTD ;Write lower nibble
bsf PORTE,RE1 ;Drive E pin low so LCD will process byte
rcall T40 ;Wait 40 usec
bsf PORTE,RE0 ;Drive RS pin high for displayable characters
movf PREINC0,W ;Increment pointer, then get next byte
UNTIL .Z ;Is it zero?
return

Note the use of indirect addressing:
(FSR0 and INDF0)
User-defined characters

- HD44780 supports up to 8 user-defined characters
- Use character codes 0x01 – 0x00f
- Before using, must be defined by storing the pixel pattern in a character-generating RAM (CGRAM) on the controller chip

User-defined Characters

- Can write a user-defined character to the CGRAM using the DisplayC subroutine
- See example 7.5 in the text

CGRAM Address Map

Writing to the CGRAM is done using the Set CGRAM Address command

And finally, one last mystery

- The HD44780 has some initialization quirks.
- The recommended initialization sequence, following power-up is:
  - wait for 0.1 seconds
  - set the device to 8-bit mode three times
  - set device to 4-bit mode
  - complete additional device configuration
**Initialization, Continued**

- The QwikFlash LCD can be properly initialized by writing the following command string to the controller, **a nibble at a time** (Must be in Command mode (RS=0); also, must wait .01 seconds first):

  ```
  LCDstr db 0x33,0x32,0x28,0x01,0x0c,0x06,0x00
  ```

  **Function set 8 bit mode**
Initialization, Continued
• The QwikFlash LCD can be properly initialized by writing the following command string to the controller, **a nibble at a time** (Must be in Command mode (RS=0); also, must wait .01 seconds first):

```
LCDstr db 0x33,0x32,0x28,0x01,0x0c,0x06,0x00
```

- **Function set**
  - 4 bit mode,
  - two rows,
  - 5x7 characters

- **Clear display**

- **Display on**
- **Cursor underline off**
- **Cursor blink off**

- **Display shift off,**
  - **Address increment**

  This causes display address (cursor position) to be automatically incremented following each character write. Can also set controller to automatically decrement the address.