Recap

I/O synchronization
LCD interface
Implementing local variables with a stack frame
Parameter passing

Overview

LCD programming
Fixed-point numbers

LCD programming

<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>
</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>
</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>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Entry mode set

<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>
</tr>
</thead>
<tbody>
<tr>
<td>Display on/off control</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>Cursor or display shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>S/C</td>
<td>R/L</td>
<td>—</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>
</tr>
<tr>
<td>Set CGRAM address</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
<td>ACG</td>
</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>
</tr>
</tbody>
</table>

I/D = 1: Increment
I/D = 0: Decrement
S  = 1: Accompanies display shift
S/C= 1: Display shift
S/C= 0: Cursor move
R/L = 1: Shift to the right
R/L = 0: Shift to the left
DL = 1: 8 bits, DL = 0: 4 bits
N  = 1: 2 lines, N = 0: 1 line
F  = 1: 5 × 10 dots, F = 0: 5 × 8 dots
BF = 1: Internally operating
BF = 0: Instructions acceptable
- 2-line display (N = 1) (Figure 4)
  
  - Case 1: When the number of display characters is less than \(40 \times 2\) lines, the two lines are displayed from the head. Note that the first line end address and the second line start address are not consecutive. For example, when just the HD44780 is used, 8 characters \(\times 2\) lines are displayed. See Figure 5.

  When display shift operation is performed, the DDRAM address shifts. See Figure 5.

### Figure 4 2-Line Display

<table>
<thead>
<tr>
<th>Display position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDRAM address</td>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
</tr>
</tbody>
</table>

### Figure 5 2-Line by 8-Character Display Example

- Power on
  - Wait for more than 15 ms after \(V_{CC}\) rises to 4.5 V

- RS \(\bar{RW}0\) \(\bar{RW}7\) \(\bar{RS}0\) \(\bar{DSR}\) \(\bar{DB}4\)
  - \(0\) \(0\) \(0\) \(0\) \(1\)
  - Wait for more than 4.1 ms

- RS \(\bar{RW}0\) \(\bar{RW}7\) \(\bar{RS}0\) \(\bar{DSR}\) \(\bar{DB}4\)
  - \(0\) \(0\) \(0\) \(0\) \(1\)
  - Wait for more than 100 \(\mu\)s

- RS \(\bar{RW}0\) \(\bar{RW}7\) \(\bar{RS}0\) \(\bar{DSR}\) \(\bar{DB}4\)
  - \(0\) \(0\) \(0\) \(0\) \(1\)

- RS \(\bar{RW}0\) \(\bar{RW}7\) \(\bar{RS}0\) \(\bar{DSR}\) \(\bar{DB}4\)
  - \(0\) \(0\) \(0\) \(0\) \(1\)
  - \(0\) \(0\) \(0\) \(0\) \(F\) \(\times\)
  - \(0\) \(0\) \(0\) \(0\) \(0\)
  - \(0\) \(0\) \(0\) \(0\) \(0\)
  - \(0\) \(0\) \(0\) \(0\) \(1\)
  - \(0\) \(0\) \(0\) \(1\) \(V_{CC}\)

  Initialization ends

  - BF cannot be checked before this instruction
  - Function set (Interface is 8 bits long)

  BF can be checked after the following instructions: When BF is not checked, the waiting time between instructions is longer than the execution instruction time (See Table 5.3)

  - Function set (Interface is 8 bits long. Specify the number of display lines and character font. The number of display lines and character font cannot be changed after this point. Display off. Display clear. Entry mode set)

### Figure 26 4-Bit Interface
Clear display
; 1) outCsr($01)  causes Clear
; 2) blind cycle 1.64ms wait
; 3) outCsr($02)  sends the Cursor to home
; 4) blind cycle 1.64ms wait

Move cursor to
; 1) outCsr(DDaddr+$80)
    first row (left-most 8) DDaddr is 0 to 7
    second row (right-most 8) DDaddr is $40 to $47

Define up to 8 new fonts
// font is the font number 0 to 7
// n0-n7 is the 5 by 8 pixel map
CGaddr=((font&0x07)<<3)+0x40;
outCsr(CGaddr); // set CG RAM address
LCD_OutChar(n0);
LCD_OutChar(n1);
LCD_OutChar(n2);
LCD_OutChar(n3);
LCD_OutChar(n4);
LCD_OutChar(n5);
LCD_OutChar(n6);
LCD_OutChar(n7);
outCsr(0x80); // revert back to DD RAM
Example
//        n0 = $00
//        n1 = $11
//        n2 = $1F
//        n3 = $15
//        n4 = $15
//        n5 = $15
//        n6 = $0E
//        n7 = $04
// set font=5 to this graphic
// CGaddr=((5&0x07)<<3)+0x40 = $68
outCsr(0x68); // set CG RAM address
LCD_OutChar(0x00);
LCD_OutChar(0x11);
LCD_OutChar(0x1F);
LCD_OutChar(0x15);
LCD_OutChar(0x15);
LCD_OutChar(0x15);
LCD_OutChar(0x0E);
LCD_OutChar(0x04);
outCsr(0x80); // revert back to DD RAM

// To output this graphic
LCD_OutChar(5);

The floating-point format
Bit 31  sign, s=0 for positive, s=1 for negative
Bits 30:23  8-bit biased binary exponent \(0 \leq e \leq 255\)
Bits 22:0  24-bit mantissa, \(m\)
expressed as a binary fraction
a binary 1 as the most significant bit is implied
\[ m = 1.m_1m_2m_3...m_{23} \]

\[
\begin{array}{|c|c|c|c|}
\hline
s & e_7 & e_0 & m_1 & m_{23} \\
\hline
\end{array}
\]

\[ f = (-1)^s \cdot 2^{e-127} \cdot m \]

**Fixed point numbers**

**Why?** (wish to represent non-integer values)
Next lab measures distance from 0 to 3 cm
E.g., 1.234 cm

**When?** (range is known, range is small)
Range is 0 to 3cm
Resolution is 0.003 cm

**How?** (value = Integer*\( \Delta \))
16-bit unsigned integer
\( \Delta = 10^{-3} \) decimal fixed-point
Range becomes 0.000 to 65.535

The design of **LCD_OutFix**

**What does it mean?**
Fixed-point number (integer part) to ASCII
\( \Delta = 10^{-3} \) decimal fixed-point

Work through **LCD_OutFix** with Input = 1234

** LCD_OutFix**
0) save any registers that will be destroyed by pushing on the stack
1) allocate local variables letter and num on the stack
2) initialize num to input parameter, which is the integer part
3) if number is less or equal to 9999, go the step 6
4) output the string "*.*** " calling LCD_OutString
5) go to step 19
6) perform the division num/1000, putting the quotient in letter, and the remainder in num
7) convert the ones digit to ASCII, letter = letter+$30
8) output letter to the LCD by calling LCD_OutChar
9) output "." to the LCD by calling LCD_OutChar
10) perform the division \(\text{num}/100\), putting the quotient in \(\text{letter}\), and the remainder in \(\text{num}\).
11) convert the tenths digit to ASCII, \(\text{letter} = \text{letter} + \$30\).
12) output letter to the LCD by calling LCD_OutChar.
13) perform the division \(\text{num}/10\), putting the quotient in \(\text{letter}\), and the remainder in \(\text{num}\).
14) convert the hundredths digit to ASCII, \(\text{letter} = \text{letter} + \$30\).
15) output letter to the LCD by calling LCD_OutChar.
16) convert the thousandths digit to ASCII, \(\text{letter} = \text{num} + \$30\).
17) output letter to the LCD by calling LCD_OutChar.
18) output ‘ ’ to the LCD by calling LCD_OutChar.
19) deallocate variables.
20) restore the registers by pulling off the stack.

How do we test \(\text{LCD\_OutFix}\)?

Stabilize inputs with a test set of inputs
User types in input, your program displays results.

Recursion, a function that calls itself, see book Program 3.12

// Variable format 1 to 5 digits with no space before or after
// uses recursion to convert a decimal number to an ASCII string
void OutUDec(unsigned short n){
    if(n >= 10){
        OutUDec(n/10);
        n = n%10;
    }
    OutChar(n+'0'); /* n is between 0 and 9 */
}

Program 3.12. Print unsigned 16-bit decimal to an output device.

Put variables on the stack
Save and restore variables as needed

End condition
Simple answer,
Return value

Body
Call itself, calculate result
Return value

Fact(1) = 1
Fact(n) = n*Fact(n-1) if n>1
; recursive implementation
; Input: RegD is n

Jonathan W. Valvano
; Output: RegD is Fact(n)
n    set  2
Fact  pshd
    pshy
    cpd  #1    ; end condition
    bls  done
    subd  #1    ; n-1
    bsr  Fact  ; RegD=Fact(n-1)
    ldy  n,s  ; RegY=n
    emul       ; RegD=n*Fact(n-1)
done  puly
    leas 2,s
    rts

Draw a stack frame

The bottom line
Stack is useful for local variables, parameter passing
Draw stack picture by hand executing assembly code
LCD is an output device for embedded systems
    Text, graphics, touch pads, color
    On board controller handles bit-level functions
Fixed-point allows non-integers without FP hardware