LCD
(Chapter 7)
and the Basic Timer
Basic Timer

• Driven by ACLK.
• BTCNT2 can generate interrupts (as slow as 1 Hz: 32768 / 256 / 128)
• BTCNT1 is used to generate a clock for LCD controller
How LCD works

To align crystal, voltage must be pure AC (average=0) – typically 50-150 Hz.
Reflective LCD

FIGURE 7: REFLECTIVE LCD PATH OF LIGHT

Light Source

Backplane Electrode

Segment Electrode

Front Polarizer (Vertical)

LIQUID CRYSTAL

OFF

Rear Polarizer (Horizontal)

Reflector

Light Source

Backplane Electrode

Segment Electrode

Front Polarizer (Vertical)

LIQUID CRYSTAL

ON

Rear Polarizer (Horizontal)

Reflector
One Common (Backplane) Electrode

8 Segment (SEG0→SEG7) electrodes are required (7 for numeral, one for decimal point).

SEG0 and SEG1 are out of phase (SEG2-7 not shown). (SEG0 also in phase with SEG2, 3, 5 and 7)

COM0-SEG0 is large AC signal, so segment is displayed.

COM0-SEG1 is small (zero) AC signal, so segment is *not* displayed.

8 Segment + 1 COM is 9 total electrodes.
Two Common (Backplane) Electrode

Multiplexing reduces total number of electrodes (4 Seg + 2 COM = 6 electrodes)

COM0 and COM1 are more complex

SEG0 and SEG1 are more complex

COM0-SEG0 is large AC signal, so segment is displayed.

COM0-SEG1 is small (but non-zero) AC signal (below some threshold value), so segment is *not* displayed.

Off segments have small AC voltages,
On segments have large voltages

Total number of segments is:
(# SEG) * (# COM)
Three Common (Backplane) Electrode

And so on...

... but always the total number of segments is: (# SEG) * (# COM)
LCD_A Module

- Directly drives LCD displays by creating the ac segment and common voltage signals automatically.
- Features include:
  - Display memory
  - Automatic signal generation
  - Configurable frame frequency
  - Blinking capability
  - Regulated charge pump
  - Contrast control by software
  - Support for 4 types of LCDs:
    - Static
    - 2-mux, 1/2 bias or 1/3 bias
    - 3-mux, 1/2 bias or 1/3 bias
    - 4-mux, 1/2 bias or 1/3 bias
- Outputs
  - 40 SEG (Selected as LCD pins in groups)
  - 4 COM (COM1,2,3 multiplexed with P5)
  - Up to 160 segments
Voltage generation
(See User’s Manual for details)
Display Controller

- Display memory has 20 bytes = 160 bits.
- One bit for each displayable segment.
- The key to using it is understanding the mapping between bits and segments.
- Controller takes care of all voltage generation...

*Figure 26–1. LCD_A Controller Block Diagram*
Experimenter’s board LCD

- 22 SEG * 4 COM = 88 segments
- Each SEG line goes to 4 segments
- One of the four is chosen (serially) by the COM lines
From a software perspective, we don’t care about SEG’s and COM’s, just the bits in the LCD memory. The LCD controller takes care of details.

You’d think the mappings are as shown... but you’d be wrong!
Correct Mapping

Let’s look at digit 4 (S6,7_{SBLCD4A} = S10,11_{MSP430} = mem 0x96)
// LCD Segments
#define LCD_A 0x01
#define LCD_B 0x02
#define LCD_C 0x04
#define LCD_D 0x08
#define LCD_E 0x40
#define LCD_F 0x10
#define LCD_G 0x20

// LCD Segment Mapping
const UInt8 LCD_Char_Map[] =
{
    LCD_A+LCD_B+LCD_C+LCD_D+LCD_E+LCD_F, // '0'
    LCD_B+LCD_C,                          // '1'
    LCD_A+LCD_B+LCD_D+LCD_E+LCD_G,        // '2'
    LCD_A+LCD_B+LCD_C+LCD_D+LCD_G,        // '3'
    LCD_B+LCD_C+LCD_F+LCD_G,              // '4'
    LCD_A+LCD_C+LCD_D+LCD_F+LCD_G,        // '5'
    LCD_A+LCD_C+LCD_D+LCD_E+LCD_F+LCD_G,  // '6'
    LCD_A+LCD_B+LCD_C,                    // '7'
    LCD_A+LCD_B+LCD_C+LCD_D+LCD_E+LCD_F+LCD_G, // '8'
    LCD_A+LCD_B+LCD_C+LCD_D+LCD_E+LCD_G,  // '9'
    LCD_A+LCD_B+LCD_C+LCD_F+LCD_G,        // '6'
    LCD_A+LCD_C+LCD_D+LCD_E+LCD_F+LCD_G,  // '8'
    LCD_A+LCD_B+LCD_C+LCD_F+LCD_G,        // '9'
};
All digits

```c
// LCD Constants
#define LCD_MEM_OFFSET  2 // Offset from LCDMEM[0]

void dispChar(UInt8 pos, UInt8 index) { // Display character on LCD
    LCDMEM[pos + LCD_MEM_OFFSET] &= 0x80; // Clear segments (but not 8th bit)
    LCDMEM[pos + LCD_MEM_OFFSET] |= LCD_Char_Map[index]; // Set Segments
}
```

Main code setup

```c
void main(void) {
    //*******Code not shown************/

    // Set ticker to 32768/(256*128) = 1 Hz, Enable BT interrupt
    BTCTL = BT_fCLK2_DIV128 | BT_fCLK2_ACLK_DIV256;
    IE2 |= BTIE;

    P2DIR |= BIT2; // Bit for LED

    initLCD_A(); // Initialize LCD

    while(1) {
        //*******Code not shown************/
    }
}
```

// Basic Timer Interrupt Service Routine
#pragma vector=BASICTIMER_VECTOR
__interrupt void basic_timer_ISR(void) {
    _bic_SR_register_on_exit(LPM3_bits);
```
while (1) {
  liveTime = BTCNT2; // Value of BTCNT2 measures time since interrupt.

  _BIS_SR(LPM3_bits + GIE); // LPM3, enable interrupts
  P2OUT = BIT2; // Turn on P2.2 (LED)

  if (++sec1>9) { // Increment seconds and check for rollover
    sec1=0; // into 10's of seconds. If so, set sec1=0 and...
    if (++sec10>5) { // Increment seconds and check for rollover
      sec10=0; // into minutes. If so, set sec10=0 and...
        if (++min>10) min=0; // Increment minutes and check for rollover.
    }
  }
  dispChar (1,sec1); // Display seconds
  dispChar (2,sec10); // Display 10's of seconds
  dispChar (3,min); // Display minutes

  LCDMEM[2 + LCD_MEM_OFFSET] ^= COLON3; // Toggle colon.

  P2OUT &= ~BIT2; // Turn off P2.2 (LED)
}
# LCD Registers

**Table 26-2. LCD Controller Registers**

<table>
<thead>
<tr>
<th>Register</th>
<th>Short Form</th>
<th>Register Type</th>
<th>Address</th>
<th>Initial State</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_A control register</td>
<td>LCDACTL</td>
<td>Read/write</td>
<td>090h</td>
<td>Reset with PUC</td>
</tr>
<tr>
<td>LCD memory 1</td>
<td>LCDM1</td>
<td>Read/write</td>
<td>091h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 2</td>
<td>LCDM2</td>
<td>Read/write</td>
<td>092h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 3</td>
<td>LCDM3</td>
<td>Read/write</td>
<td>093h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 11</td>
<td>LCDM11</td>
<td>Read/write</td>
<td>09Bh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 12</td>
<td>LCDM12</td>
<td>Read/write</td>
<td>09Ch</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 13</td>
<td>LCDM13</td>
<td>Read/write</td>
<td>09Dh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 14</td>
<td>LCDM14</td>
<td>Read/write</td>
<td>09 Eh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 15</td>
<td>LCDM15</td>
<td>Read/write</td>
<td>09Fh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 16</td>
<td>LCDM16</td>
<td>Read/write</td>
<td>0A0h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 17</td>
<td>LCDM17</td>
<td>Read/write</td>
<td>0A1h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 18</td>
<td>LCDM18</td>
<td>Read/write</td>
<td>0A2h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 19</td>
<td>LCDM19</td>
<td>Read/write</td>
<td>0A3h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 20</td>
<td>LCDM20</td>
<td>Read/write</td>
<td>0A4h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD_A port control 0</td>
<td>LCDAPCTL0</td>
<td>Read/write</td>
<td>0ACh</td>
<td>Reset with PUC</td>
</tr>
<tr>
<td>LCD_A port control 1</td>
<td>LCDAPCTL1</td>
<td>Read/write</td>
<td>0ADh</td>
<td>Reset with PUC</td>
</tr>
<tr>
<td>LCD_A voltage control 0</td>
<td>LCDAVCTL0</td>
<td>Read/write</td>
<td>0AEh</td>
<td>Reset with PUC</td>
</tr>
<tr>
<td>LCD_A voltage control 1</td>
<td>LCDAVCTL1</td>
<td>Read/write</td>
<td>0AFh</td>
<td>Reset with PUC</td>
</tr>
</tbody>
</table>
### LCD ACTL, LCD_A Control Register

<table>
<thead>
<tr>
<th></th>
<th>LCDFREQx</th>
<th>LCDMXx</th>
<th>LCDSON</th>
<th>Unused</th>
<th>LCDON</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-5</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
</tr>
</tbody>
</table>

**LCDFREQx** Bits

- LCD frequency select. These bits select the ACLK divider for the LCD frequency.
- 000 Divide by 32
- 001 Divide by 64
- 010 Divide by 96
- 011 Divide by 128
- 100 Divide by 192
- 101 Divide by 256
- 110 Divide by 384
- 111 Divide by 512

**LCDMXx** Bits

- LCD mux rate. These bits select the LCD mode.
- 4-3
  - 00 Static
  - 01 2-mux
  - 10 3-mux
  - 11 4-mux

**LCDSON** Bit 2

- LCD segments on. This bit supports flashing LCD applications by turning off all segment lines, while leaving the LCD timing generator and R33 enabled.
- 0 All LCD segments are off
- 1 All LCD segments are enabled and on or off according to their corresponding memory location.

**Unused** Bit 1

- Unused

**LCDON** Bit 0

- LCD On. This bit turns on the LCD_A module.
- 0 LCD_A module off.
- 1 LCD_A module on.
Enabling Segments

Segments are enabled 4 at a time

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>LCDS28 28 to 31 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>6</td>
<td>LCDS24 24 to 27 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>5</td>
<td>LCDS20 20 to 23 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>4</td>
<td>LCDS16 16 to 19 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>3</td>
<td>LCDS12 12 to 15 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>2</td>
<td>LCDS8 8 to 11 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>1</td>
<td>LCDS4 4 to 7 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
<tr>
<td>0</td>
<td>LCDS0 0 to 3 enable</td>
</tr>
<tr>
<td></td>
<td>This bit only affects pins with multiplexed functions. Dedicated LCD pins are always LCD function.</td>
</tr>
<tr>
<td></td>
<td>0: Multiplexed pins are port functions.</td>
</tr>
<tr>
<td></td>
<td>1: Pins are LCD functions</td>
</tr>
</tbody>
</table>

* Segments S0–S3 on the MSP430FG461x devices are disabled from LCD functionality when charge pump is enabled.
References

• Very complete example code at: http://focus.ti.com/docs/toolsw/folders/print/msp-exp430fg4618.html
  This code uses some C-syntax… that we haven’t covered. Feel free to ask me if it is unclear.

Also