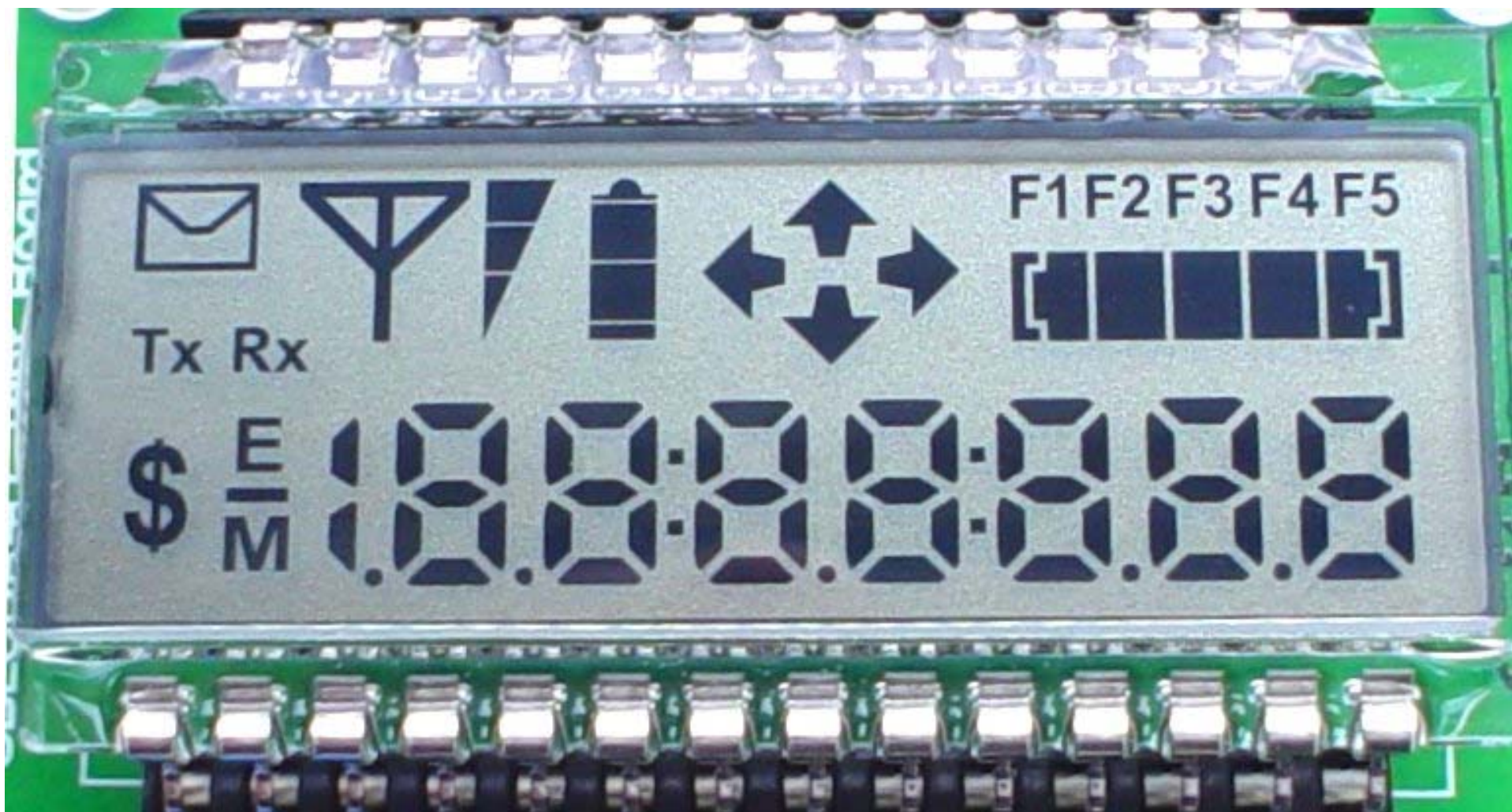




Writing to an LCD

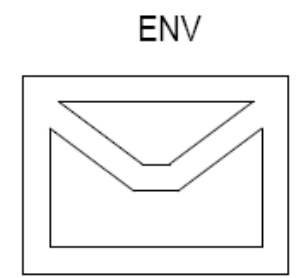
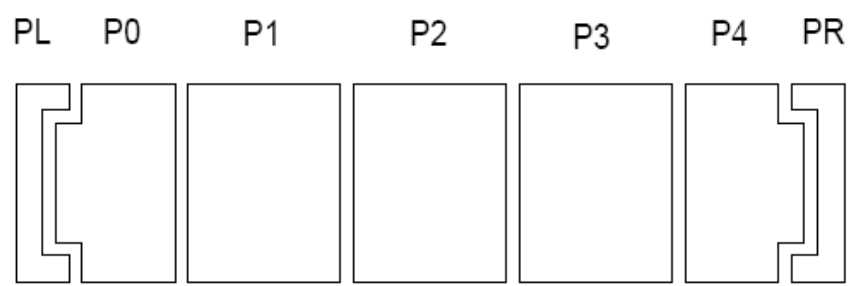
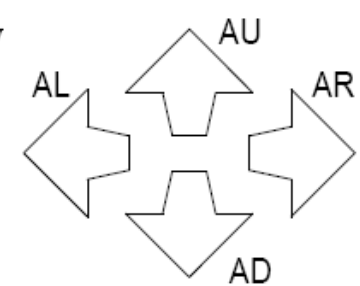
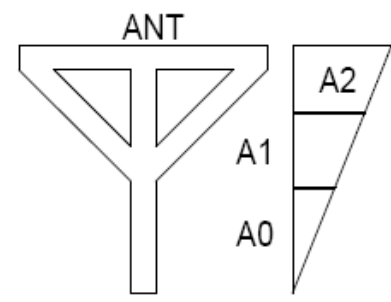
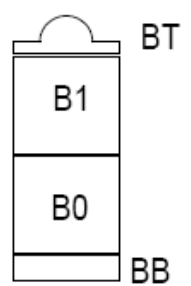
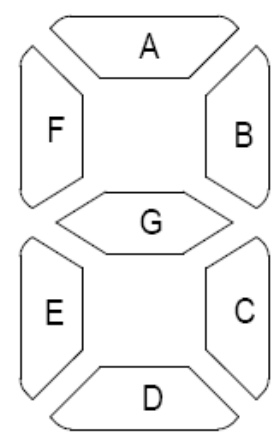
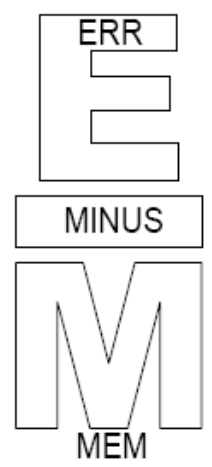
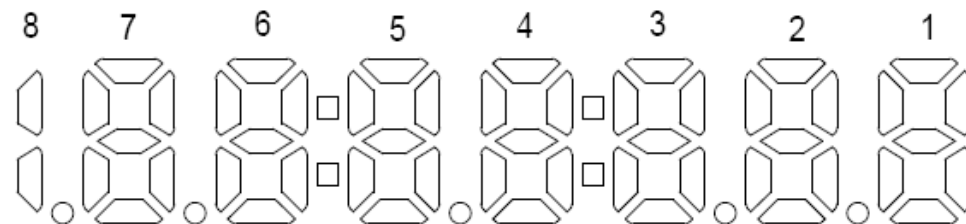
SoftbaughSBLCDA4 LCD
Display and Drivers

SoftBaugh SBLCDA4 LCD display



SoftBaugh SBLCDA4 Features

- 4-MUX operation
- 2.7v to 3.6v operation
- 7.1 seven-segment with minus symbology allows versatile text
- Envelope, Error, and Memory symbols
- Arrows left, up, right, and down
- Battery with two-segment meter
- Antenna with three-segment meter, plus Tx/Rx symbols
- Colons for HH:MM:SS operation, plus five decimal options
- Progress bar for convenient user feedback
- Five function symbols
- 6 oclock viewing angle



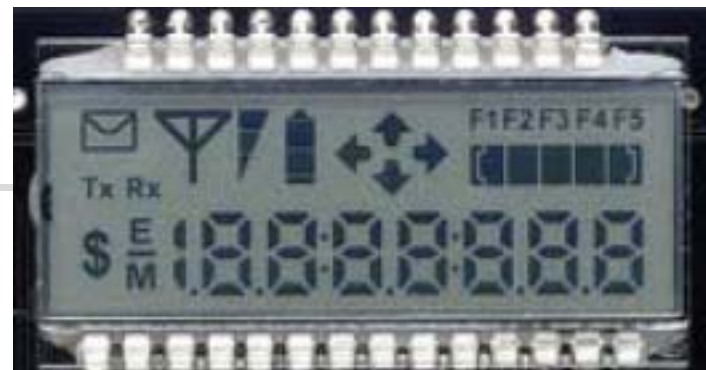
Segments

Segment Mapping



PIN	COM3	COM2	COM1	COM0		COM3	COM2	COM1	COM0	PIN
1	DP7	7E	7G	7F						
2	7D	7C	7B	7A		MEM	MINUS	ERR	DOL	26
3	DP6	6E	6G	6F		8BC	RX	TX	ENV	25
4	6D	6C	6B	6A		A0	A1	A2	ANT	24
5	COL5	5E	5G	5F		BB	B0	B1	BT	23
6	5D	5C	5B	5A		AL	AD	AR	AU	22
7	DP4	4E	4G	4F		P2	P1	P0	PL	21
8	4D	4C	4B	4A		F4	F3	F2	F1	20
9	COL3	3E	3G	3F		P3	P4	PR	F5	19
10	3D	3C	3B	3A					COM0	18
11	DP2	2E	2G	2F				COM1		17
12	2D	2C	2B	2A			COM2			16
13	DP1	1E	1G	1F		COM3				15
14	1D	1C	1B	1A						

LCD



SoftBaugh SBLCDA4

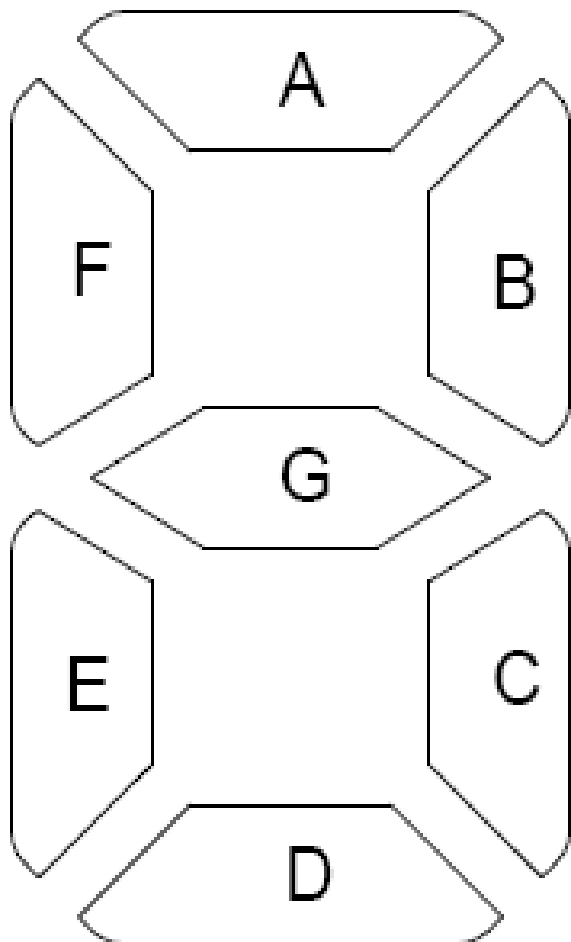
S0	P\$14	1A_1B_1C_1D			
S1	P\$13	1F_1G_1E_DP1	DOL_ERR_MINUS_MEM	P\$26	S21
S2	P\$12	2A_2B_2C_2D	ENV_TX_RX_8BC	P\$25	S20
S3	P\$11	2F_2G_2E_DP2	ANT_A2_A1_A0	P\$24	S19
S4	P\$10	3A_3B_3C_3D	BT_B1_B0_BB	P\$23	S18
S5	P\$9	3F_3G_3E_COL3	AU_AR_AD_AL	P\$22	S17
S6	P\$8	4A_4B_4C_4D	PL_P0_P1_P2	P\$21	S16
S7	P\$7	4F_4G_4E_DP4	F1_F2_F3_F4	P\$20	S15
S8	P\$6	5A_5B_5C_5D	F5_PR_P4_P3	P\$19	S14
S9	P\$5	5F_5G_5E_COL5	COM0	P\$18	COM0
S10	P\$4	6A_6B_6C_6D	COM1	P\$17	COM1
S11	P\$3	6F_6G_6E_DP6	COM2	P\$16	COM2
S12	P\$2	7A_7B_7C_7D	COM3	P\$15	COM3
S13	P\$1	7F_7G_7E_DP7			

LCD Segment Displays

- Lower tech, earlier design, cheap
- Here, each segment is made of 7 bars

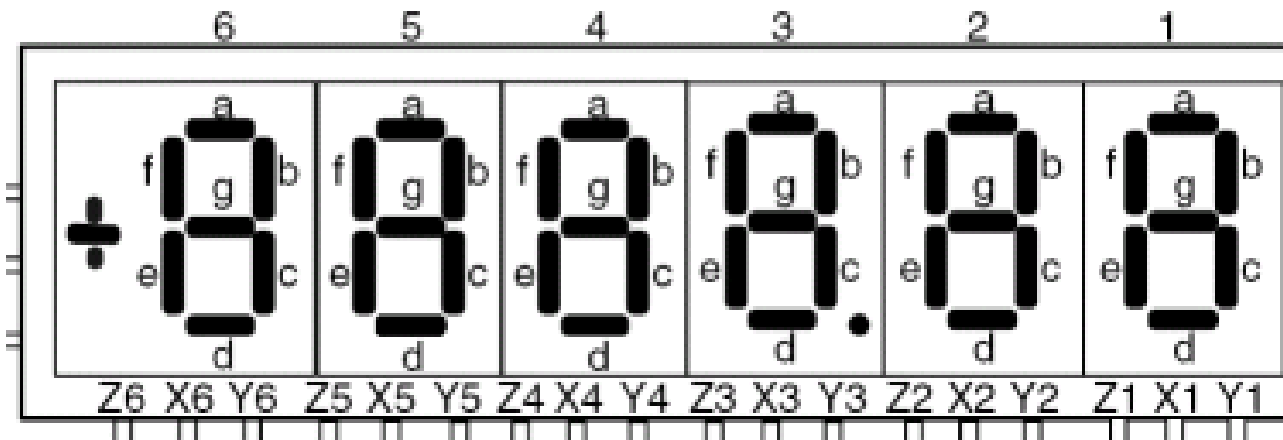


Coding a Segment



- Each bar in a segment is labeled with a letter
- Create map of bars to display each letter and number
- What is a '7'?
- What is a '6'?

Coding Segments



1 : -bc- ----	6 : a-cd efg-
2 : ab-d e-g-	7 : abc- ----
3 : abcd --g-	8 : abcd efg-
4 : -bc- -fg-	9 : abcd -fg-
5 : a-cd -fg-	'A': abc- efg-

```

29 // LCD Segments
30 #define LCD_A BIT0
31 #define LCD_B BIT1
32 #define LCD_C BIT2
33 #define LCD_D BIT3
34 #define LCD_E BIT6
35 #define LCD_F BIT4
36 #define LCD_G BIT5
37 #define LCD_H BIT7

```

from LCD.h

Mapping Segments in C

```

3  const UInt8 LCD_Char_Map[] =
4  {
5      LCD_A+LCD_B+LCD_C+LCD_D+LCD_E+LCD_F,          // '0' or 'O'
6      LCD_B+LCD_C,                                  // '1' or 'I'
7      LCD_A+LCD_B+LCD_D+LCD_E+LCD_G,              // '2' or 'Z'
8      LCD_A+LCD_B+LCD_C+LCD_D+LCD_G,              // '3'
9      LCD_B+LCD_C+LCD_F+LCD_G,                    // '4' or 'y'
10     LCD_A+LCD_C+LCD_D+LCD_F+LCD_G,               // '5' or 'S'
11     LCD_A+LCD_C+LCD_D+LCD_E+LCD_F+LCD_G,         // '6' or 'b'
12     LCD_A+LCD_B+LCD_C,                            // '7'
13     LCD_A+LCD_B+LCD_C+LCD_D+LCD_E+LCD_F+LCD_G,   // '8' or 'B'
14     LCD_A+LCD_B+LCD_C+LCD_F+LCD_G,               // '9' or 'g'
15     LCD_A+LCD_B+LCD_C+LCD_E+LCD_F+LCD_G,         // 'A'
16     LCD_A+LCD_D+LCD_E+LCD_F,                     // 'C'
17     LCD_B+LCD_C+LCD_D+LCD_E+LCD_G,               // 'd'
18     LCD_A+LCD_D+LCD_E+LCD_F+LCD_G,               // 'E'
19     LCD_A+LCD_E+LCD_F+LCD_G,                     // 'F'
20     LCD_B+LCD_C+LCD_E+LCD_F+LCD_G,               // 'H'
21     LCD_B+LCD_C+LCD_D+LCD_E,                     // 'J'
22     LCD_D+LCD_E+LCD_F,                            // 'L'
23     LCD_A+LCD_B+LCD_E+LCD_F+LCD_G,               // 'P'
24     LCD_B+LCD_C+LCD_D+LCD_E+LCD_F                // 'U'
25 };
26

```

from LCD.c

Accessing the LCD

```
1 // clear LCD
2 void clrLCD(void)
3 {
4     int i;
5
6     for(i = LCD_MEM_OFFSET; i < (LCD_MEM_OFFSET+LCD_MEM_LOC); i++)
7     {
8         LCDMEM[i] = 0;
9     }
10 }
11
12 // Display all segments on LCD
13 void dispAllLCDSegs(void)
14 {
15     int i;
16
17     for(i = LCD_MEM_OFFSET; i < (LCD_MEM_OFFSET+LCD_MEM_LOC); i++)
18     {
19         LCDMEM[i] = 0xff;
20     }
21 }
```

- LCDMEM[] – memory locations for LCD
- LCD_MEM_OFFSET 2 Offset from LCDMEM[0]
- LCD_MEM_LOC 11 Num of LCDMEM[] locations used

```

62 void main(void)
63 {
64     WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
65     FLL_CTL0 |= XCAP18PF;              // Set load cap for 32k xtal
66
67     initPortPins();                     // Initialize port pins
68     initBasicTimer();                   // Initialize basic timer
69     initLCD_A();                         // Initialize LCD_A
70
71     testAll();                           /* Excite all segments on the display */
72
73     di spChar(0,0);
74     di spChar(1,1);
75     di spChar(2,2);
76     di spChar(3,13);
77     di spChar(4,15);
78     di spChar(5,1);
79     di spChar(6,19);
80
81
82     for(;;)
83     {
84         _BIS_SR(LPM3_bits + GIE);      // LPM3, enable interrupts
85
86         testChar();
87         testSpecialChar();
88         testSigLvl();
89         testBatt();
90         testPwrLvl();
91         testFunc();
92         testArrow();
93         testSymbol();
94     }
95 }

```

board.c

Writing a Character to the LCD

```
1 // Display character on LCD
2 void dispChar(UInt8 pos, UInt8 index)
3 {
4     LCDMEM[pos + LCD_MEM_OFFSET] &= ~LCD_Char_Map[8];
5
6     if( pos < LCD_NUM_DIGITS )
7     {
8         if( index < LCD_MAX_CHARS )
9         {
10            LCDMEM[pos + LCD_MEM_OFFSET] |= LCD_Char_Map[index];
11        }
12    }
13 }
```