

## Class 4 Task Program 1: Print Difference of Two Number on LCD

```
23 #define _XTAL_FREQ 2000000 // Define the crystal oscillator frequency as 20MHz (for delay functions)
24 void Lcdinit(void); // Function prototype for LCD initialization
25 void LcdCommand(uint8_t i); // Function prototype for sending commands to the LCD
26 void LcdData(char i); // Function prototype for sending data to the LCD
27 void LcdOutput(uint16_t i);
28 void BatStatus(void);
29
30 uint8_t Array[15] = {"BAT VOLT=   v"};
31 uint8_t BatLowArray[15] = {"BAT LOW   "};
32 uint8_t BatMediumArray[15] = {"BAT MEDIUM"};
33 uint8_t BatHighArray[15] = {"BAT HIGH   "};
34 char buffer[10];
35
36 uint8_t x, m, n, value;
37 float k = 15.5;
38 float j = 17.5;
39
40 void main(void){
41     Lcdinit(); // Initialize the LCD
42
43     while(1)
44     {
45         BatStatus();
46         __delay_ms(100);
47     }
48 }
49
50 // Function to initialize the LCD
51 void Lcdinit(void)
52 {
53     TRISC = 0x00; // Set PORTC as output (for control signals)
54     TRISD = 0x00; // Set PORTD as output (for data signals)
55     TRISB = 0xF0; // Set R4 to R7 as Input
56
57     OPTION_REG &= ~(0x1U << 7); // Enable Pull Up
58     __delay_ms(100); // Wait for LCD to stabilize
59
60     // LCD initialization sequence as per HD44780 LCD datasheet
61     LcdCommand(0x30); // Send function set command (8-bit mode)
62     __delay_ms(100);
63     LcdCommand(0x30); // Repeat function set command
64     __delay_ms(100);
65     LcdCommand(0x30); // Repeat function set command again
66     __delay_ms(100);
67     LcdCommand(0x38); // Set LCD for 8-bit mode, 2-line display, 5x8 font
68     __delay_ms(100);
69     LcdCommand(0x0C); // Turn on display, cursor off
70     __delay_ms(100);
71     LcdCommand(0x01); // Clear the display
72     __delay_ms(100);
73     LcdCommand(0x06);
74     __delay_ms(100);
75 }
76
```

```

77 // Function to monitor and display battery status on the LCD
78 void BatStatus(void)
79 {
80     value = PORTB & 0xF0; // Mask lower 4 bits and read only the upper 4 bits of PORTB
81
82     switch(value)
83     {
84         case 0xE0: // PORTB = 1110 0000 -> Initial battery status display
85             LcdCommand(0x80); // Move cursor to the first row, first column
86             for(int i = 0; i < 15; i++)
87             {
88                 LcdData(Array[i]); // Display battery-related text from Array
89             }
90             LcdCommand(0x89); // Move cursor to column 9
91             sprintf(buffer, "%.1f", j); // Convert floating-point battery value to a string
92             for(int i = 0; buffer[i] != '\0'; i++)
93             {
94                 LcdData(buffer[i]); // Display battery voltage on the LCD
95             }
96             LcdCommand(0xC0); // Move cursor to the second row, first column
97             for(int i = 0; i < 7; i++)
98             {
99                 LcdData(BatLowArray[i]); // Display "Low Battery" warning
100             }
101             break;
102
103         case 0xD0: // PORTB = 1101 0000 -> Battery voltage increasing
104             j += 0.1; // Increment battery voltage
105             if(j > 22.5){
106                 j = 22.5; // Limit maximum voltage to 22.5V
107             }
108             LcdCommand(0x89); // Move cursor to column 9
109             sprintf(buffer, "%.1f", j); // Convert updated voltage to string
110             for(int i = 0; buffer[i] != '\0'; i++){
111                 LcdData(buffer[i]); // Display updated voltage
112             }
113
114             if(j >= 15.5 && j <= 17.5){ // Display battery status message based on voltage level
115                 LcdCommand(0xC0); // Move cursor to the second row
116                 for(int i = 0; i < 10; i++){
117                     LcdData(BatLowArray[i]); // Display "Low Battery"
118                 }
119             }
120             else if(j >= 17.6 && j <= 20.5){
121                 LcdCommand(0xC0);
122                 for(int i = 0; i < 10; i++){
123                     LcdData(BatMediumArray[i]); // Display "Medium Battery"
124                 }
125             }
126             else if(j >= 20.6 && j <= 22.5){
127                 LcdCommand(0xC0);
128                 for(int i = 0; i < 10; i++){
129                     LcdData(BatHighArray[i]); // Display "High Battery"
130                 }
131             }
132             break;

```

```

133 case 0xB0: // PORTB = 1011 0000 -> Battery voltage decreasing
134     j -= 0.1; // Decrease battery voltage
135     if(j < 15.5){
136         j = 15.5; // Limit minimum voltage to 15.5V
137     }
138     LcdCommand(0x89); // Move cursor to column 9
139     sprintf(buffer, "%.1f", j); // Convert updated voltage to string
140     for(int i = 0; buffer[i] != '\0'; i++){
141         LcdData(buffer[i]); // Display updated voltage
142     }
143
144     if(j >= 15.5 && j <= 17.5){ // Display battery status message based on voltage level
145         LcdCommand(0xC0);
146         for(int i = 0; i < 10; i++){
147             LcdData(BatLowArray[i]); // Display "Low Battery"
148         }
149     }
150     else if(j >= 17.6 && j <= 20.5){
151         LcdCommand(0xC0);
152         for(int i = 0; i < 10; i++){
153             LcdData(BatMediumArray[i]); // Display "Medium Battery"
154         }
155     }
156     else if(j >= 20.6 && j <= 22.5){
157         LcdCommand(0xC0);
158         for(int i = 0; i < 10; i++){
159             LcdData(BatHighArray[i]); // Display "High Battery"
160         }
161     }
162     break;

```

```

163
164 case 0x70: // PORTB = 0111 0000 -> Reset battery voltage to 17.6V
165     j = 17.6; // Set voltage to predefined value
166     LcdCommand(0x89); // Move cursor to column 9
167     sprintf(buffer, "%.1f", j); // Convert updated voltage to string
168     for(int i = 0; buffer[i] != '\0'; i++){
169         {
170             LcdData(buffer[i]); // Display updated voltage
171         }
172         LcdCommand(0xC0); // Move cursor to second row
173         for(int i = 0; i < 10; i++){
174             {
175                 LcdData(BatMediumArray[i]); // Display "Medium Battery"
176             }
177         }
178         break;
179
180 default:
181     // Handle unexpected values if necessary
182     break;
183 }

```

```

186 void LcdOutput(uint16_t i){
187     uint8_t d1, d2, d3, d4;          // Creating local var to reduce memory consumption
188     d4 = (uint8_t)(i / 1000);         // Extract thousands place
189     d3 = (uint8_t)((i % 1000) / 100); // Extract hundreds place
190     d2 = (uint8_t)((i % 100) / 10);   // Extract tens place
191     d1 = (uint8_t)(i % 10);           // Extract ones place
192     LcdCommand(0x88);
193     LcdData(0x30 + d4);
194     LcdData(0x30 + d3);
195     LcdData(0x30 + d2);
196     LcdData(0x30 + d1);
197 }
198
199 // Function to send data (characters) to the LCD
200 void LcdData(char i){
201     PORTC |= (0x1 << 3); // Set RS (RC3) = 1 (indicates data mode)
202     PORTD = i;           // Place data on PORTD
203     PORTC |= (0x1 << 0); // Set EN (RC0) = 1 (enable pulse start)
204     __delay_ms(100);      // Small delay for command execution
205     PORTC &= ~(0x1 << 0); // Set EN (RC0) = 0 (enable pulse end)
206 }
207
208 // Function to send commands to the LCD
209 void LcdCommand(uint8_t i){
210     PORTC &= ~(0x1 << 3); // Set RS (RC3) = 0 (indicates command mode)
211     PORTD = i;           // Place command on PORTD
212     PORTC |= (0x1 << 0); // Set EN (RC0) = 1 (enable pulse start)
213     __delay_ms(100);      // Small delay for command execution
214     PORTC &= ~(0x1 << 0); // Set EN (RC0) = 0 (enable pulse end)
215 }

```