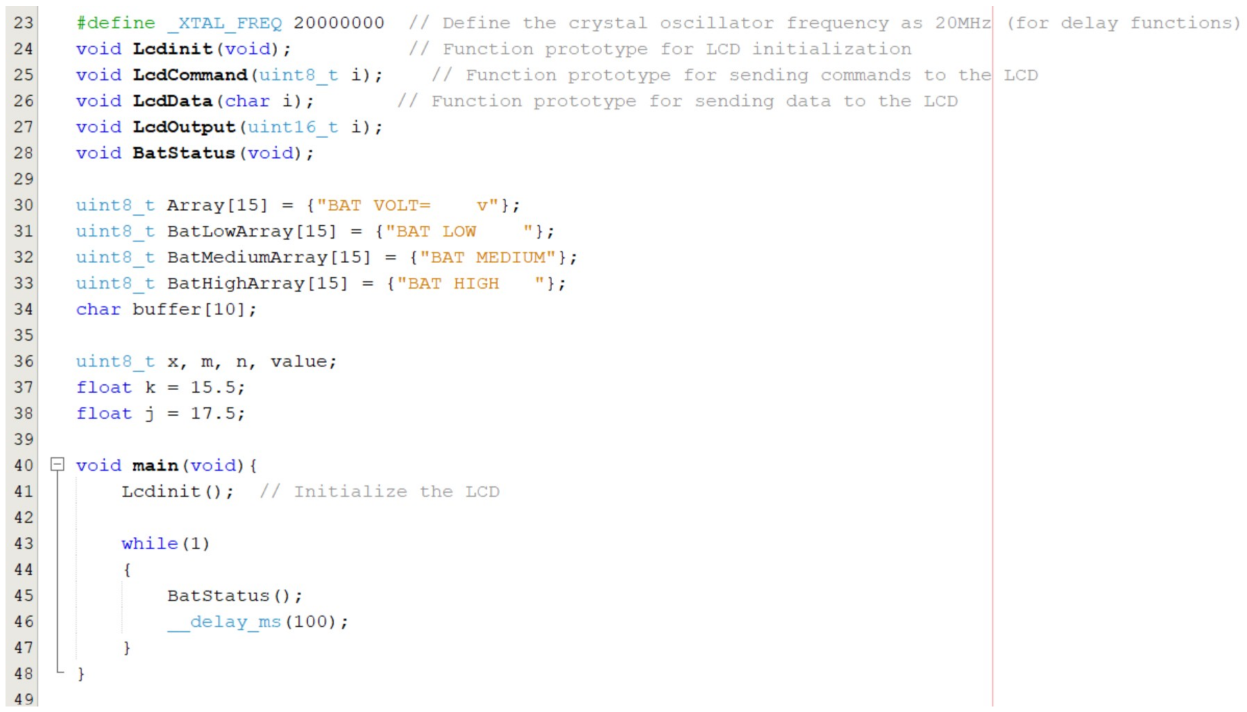
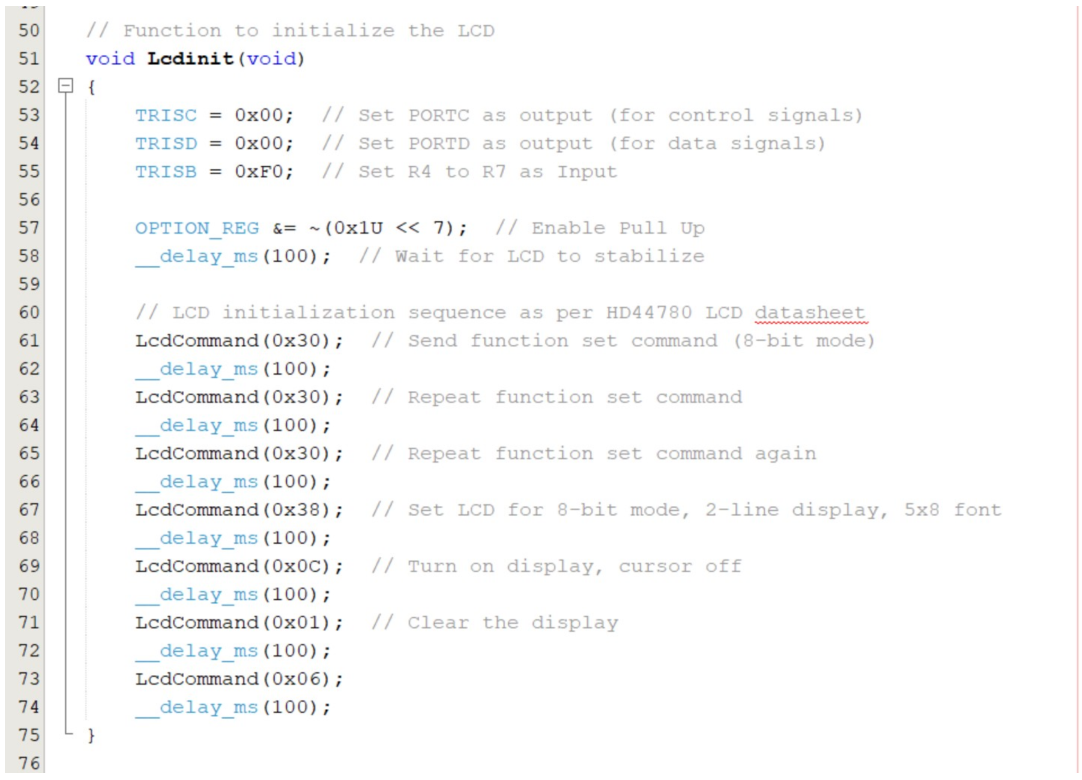
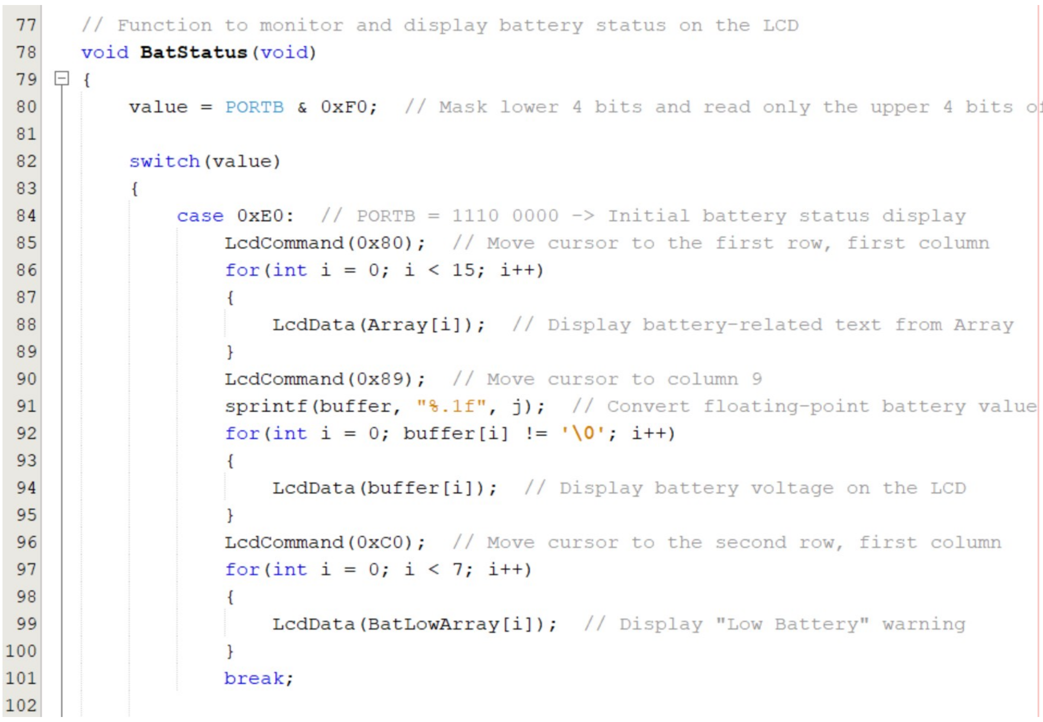
Class 4 Task Program 1:  Print Difference of Two Number on LCD



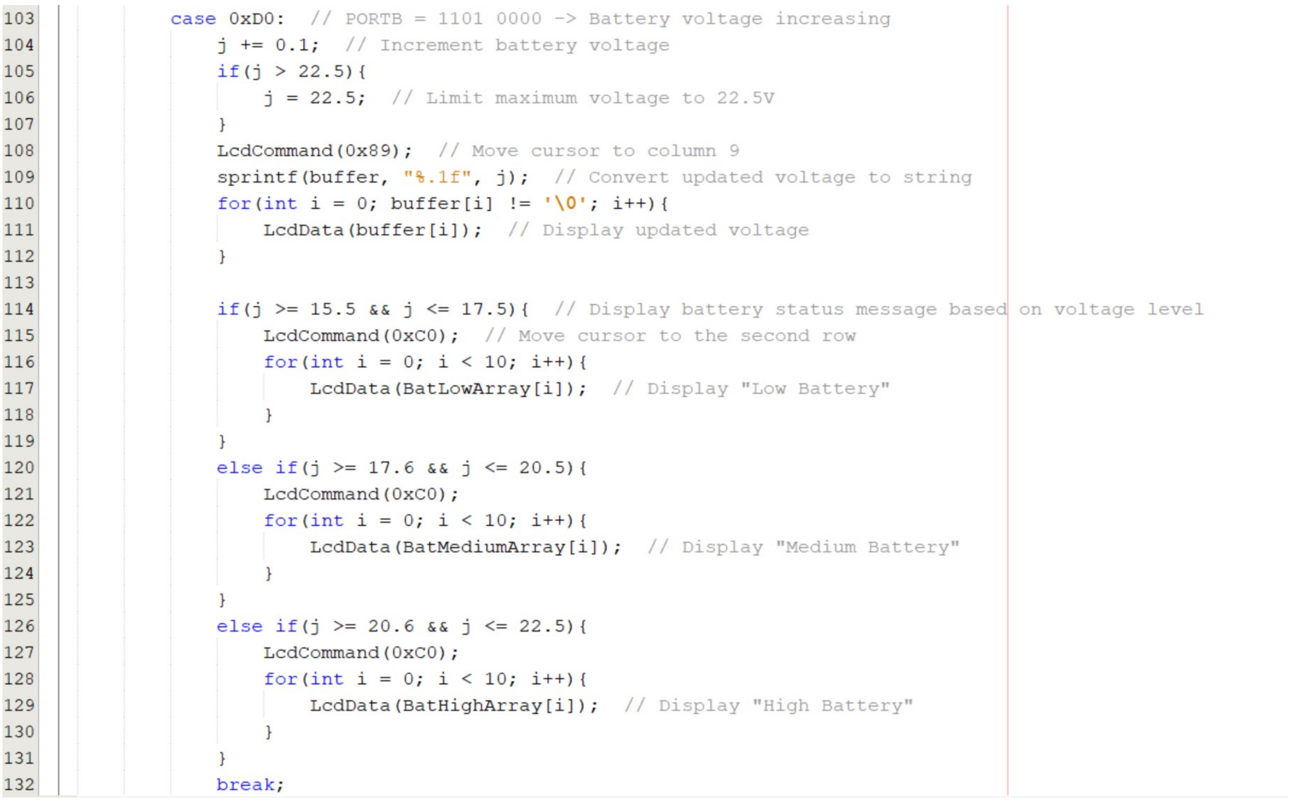
```c
#define _XTAL_FREQ 20000000  // Define the crystal oscillator frequency as 20MHz (for delay functions)
void Lcdinit(void);          // Function prototype for LCD initialization
void LcdCommand(uint8_t i);    // Function prototype for sending commands to the LCD
void LcdData(char i);        // Function prototype for sending data to the LCD
void LcdOutput(uint16_t i);
void BatStatus(void);

uint8_t Array[15] = {"BAT VOLT=    v"};
uint8_t BatLowArray[15] = {"BAT LOW    "};
uint8_t BatMediumArray[15] = {"BAT MEDIUM"};
uint8_t BatHighArray[15] = {"BAT HIGH   "};
char buffer[10];

uint8_t x, m, n, value;
float k = 15.5;
float j = 17.5;

void main(void){
    Lcdinit();  // Initialize the LCD

    while(1)
    {
        BatStatus();
        __delay_ms(100);
    }
}
```



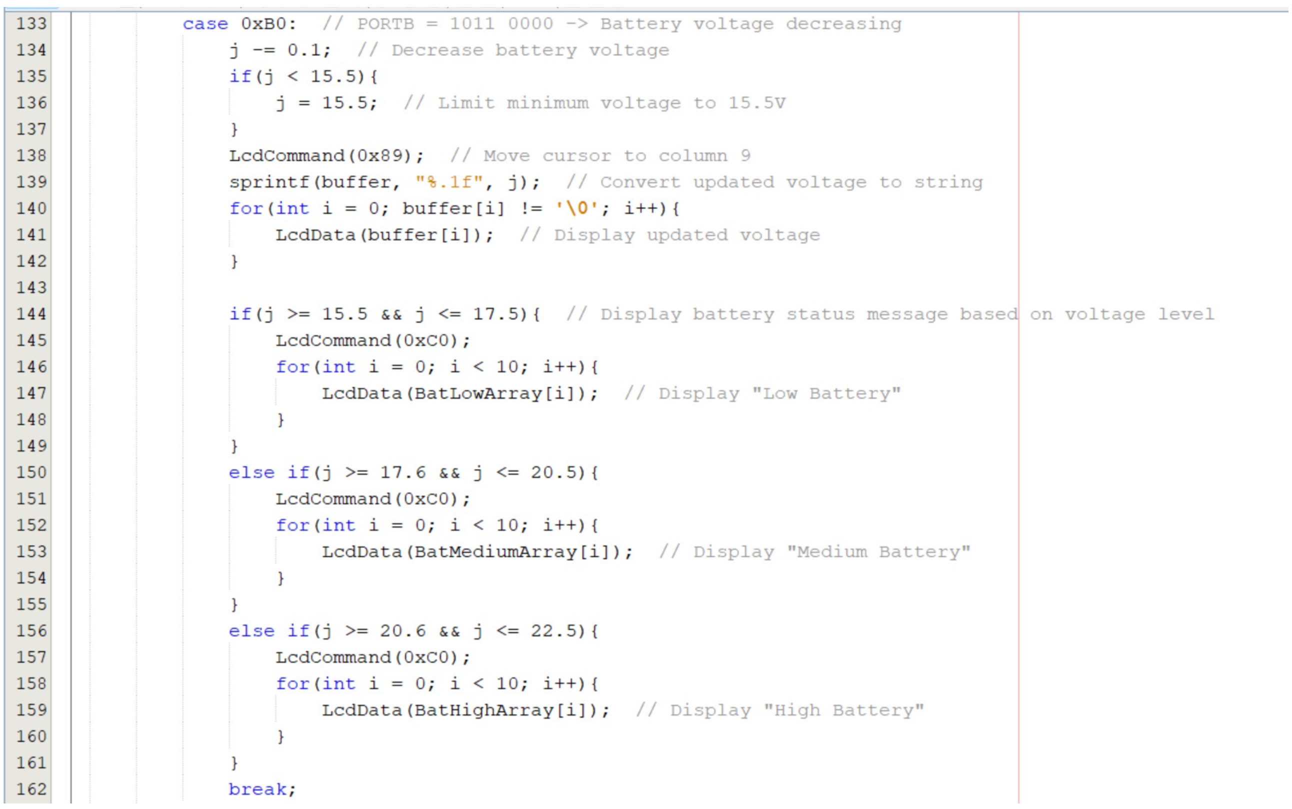
```c
// Function to initialize the LCD
void Lcdinit(void)
{
    TRISC = 0x00;  // Set PORTC as output (for control signals)
    TRISD = 0x00;  // Set PORTD as output (for data signals)
    TRISB = 0xF0;  // Set R4 to R7 as Input

    OPTION_REG &= ~(0x1U << 7);  // Enable Pull Up
    __delay_ms(100);  // Wait for LCD to stabilize

    // LCD initialization sequence as per HD44780 LCD datasheet
    LcdCommand(0x30);  // Send function set command (8-bit mode)
    __delay_ms(100);
    LcdCommand(0x30);  // Repeat function set command
    __delay_ms(100);
    LcdCommand(0x30);  // Repeat function set command again
    __delay_ms(100);
    LcdCommand(0x38);  // Set LCD for 8-bit mode, 2-line display, 5x8 font
    __delay_ms(100);
    LcdCommand(0x0C);  // Turn on display, cursor off
    __delay_ms(100);
    LcdCommand(0x01);  // Clear the display
    __delay_ms(100);
    LcdCommand(0x06);
    __delay_ms(100);
}
```



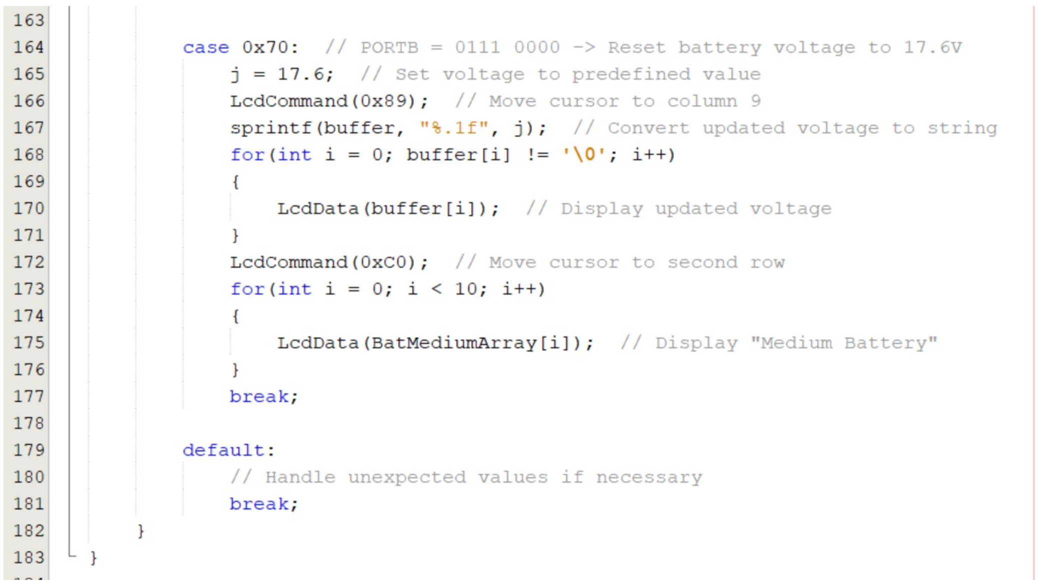
```c
// Function to monitor and display battery status on the LCD
void BatStatus(void)
{
    value = PORTB & 0xF0;  // Mask lower 4 bits and read only the upper 4 bits of PORTB

    switch(value)
    {
        case 0xE0:  // PORTB = 1110 0000 -> Initial battery status display
            LcdCommand(0x80);  // Move cursor to the first row, first column
            for(int i = 0; i < 15; i++)
            {
                LcdData(Array[i]);  // Display battery-related text from Array
            }
            LcdCommand(0x89);  // Move cursor to column 9
            sprintf(buffer, "%.1f", j);  // Convert floating-point battery value to a string
            for(int i = 0; buffer[i] != '\0'; i++)
            {
                LcdData(buffer[i]);  // Display battery voltage on the LCD
            }
            LcdCommand(0xC0);  // Move cursor to the second row, first column
            for(int i = 0; i < 7; i++)
            {
                LcdData(BatLowArray[i]);  // Display "Low Battery" warning
            }
            break;

```



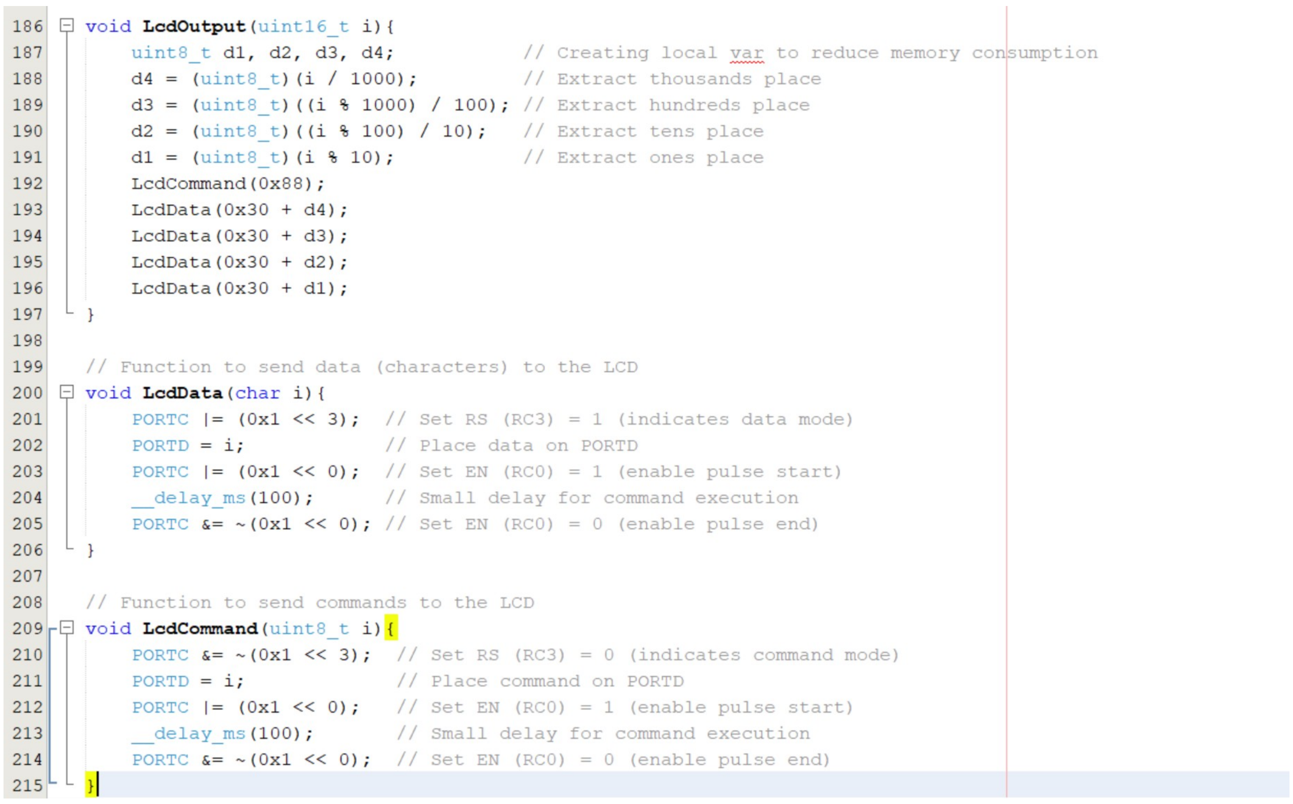
```c
        case 0xD0:  // PORTB = 1101 0000 -> Battery voltage increasing
            j += 0.1;  // Increment battery voltage
            if(j > 22.5){
                j = 22.5;  // Limit maximum voltage to 22.5V
            }
            LcdCommand(0x89);  // Move cursor to column 9
            sprintf(buffer, "%.1f", j);  // Convert updated voltage to string
            for(int i = 0; buffer[i] != '\0'; i++){
                LcdData(buffer[i]);  // Display updated voltage
            }

            if(j >= 15.5 && j <= 17.5){  // Display battery status message based on voltage level
                LcdCommand(0xC0);  // Move cursor to the second row
                for(int i = 0; i < 10; i++){
                    LcdData(BatLowArray[i]);  // Display "Low Battery"
                }
            }
            else if(j >= 17.6 && j <= 20.5){
                LcdCommand(0xC0);
                for(int i = 0; i < 10; i++){
                    LcdData(BatMediumArray[i]);  // Display "Medium Battery"
                }
            }
            else if(j >= 20.6 && j <= 22.5){
                LcdCommand(0xC0);
                for(int i = 0; i < 10; i++){
                    LcdData(BatHighArray[i]);  // Display "High Battery"
                }
            }
            break;
```



```c
        case 0xB0:  // PORTB = 1011 0000 -> Battery voltage decreasing
            j -= 0.1;  // Decrease battery voltage
            if(j < 15.5){
                j = 15.5;  // Limit minimum voltage to 15.5V
            }
            LcdCommand(0x89);  // Move cursor to column 9
            sprintf(buffer, "%.1f", j);  // Convert updated voltage to string
            for(int i = 0; buffer[i] != '\0'; i++){
                LcdData(buffer[i]);  // Display updated voltage
            }

            if(j >= 15.5 && j <= 17.5){  // Display battery status message based on voltage level
                LcdCommand(0xC0);
                for(int i = 0; i < 10; i++){
                    LcdData(BatLowArray[i]);  // Display "Low Battery"
                }
            }
            else if(j >= 17.6 && j <= 20.5){
                LcdCommand(0xC0);
                for(int i = 0; i < 10; i++){
                    LcdData(BatMediumArray[i]);  // Display "Medium Battery"
                }
            }
            else if(j >= 20.6 && j <= 22.5){
                LcdCommand(0xC0);
                for(int i = 0; i < 10; i++){
                    LcdData(BatHighArray[i]);  // Display "High Battery"
                }
            }
            break;
```



```c

        case 0x70:  // PORTB = 0111 0000 -> Reset battery voltage to 17.6V
            j = 17.6;  // Set voltage to predefined value
            LcdCommand(0x89);  // Move cursor to column 9
            sprintf(buffer, "%.1f", j);  // Convert updated voltage to string
            for(int i = 0; buffer[i] != '\0'; i++)
            {
                LcdData(buffer[i]);  // Display updated voltage
            }
            LcdCommand(0xC0);  // Move cursor to second row
            for(int i = 0; i < 10; i++)
            {
                LcdData(BatMediumArray[i]);  // Display "Medium Battery"
            }
            break;

        default:
            // Handle unexpected values if necessary
            break;
    }
}
```



```c
void LcdOutput(uint16_t i){
    uint8_t d1, d2, d3, d4;           // Creating local var to reduce memory consumption
    d4 = (uint8_t)(i / 1000);         // Extract thousands place
    d3 = (uint8_t)((i % 1000) / 100); // Extract hundreds place
    d2 = (uint8_t)((i % 100) / 10);   // Extract tens place
    d1 = (uint8_t)(i % 10);           // Extract ones place
    LcdCommand(0x88);
    LcdData(0x30 + d4);
    LcdData(0x30 + d3);
    LcdData(0x30 + d2);
    LcdData(0x30 + d1);
}

// Function to send data (characters) to the LCD
void LcdData(char i){
    PORTC |= (0x1 << 3);  // Set RS (RC3) = 1 (indicates data mode)
    PORTD = i;            // Place data on PORTD
    PORTC |= (0x1 << 0);  // Set EN (RC0) = 1 (enable pulse start)
    __delay_ms(100);      // Small delay for command execution
    PORTC &= ~(0x1 << 0); // Set EN (RC0) = 0 (enable pulse end)
}

// Function to send commands to the LCD
void LcdCommand(uint8_t i){
    PORTC &= ~(0x1 << 3);  // Set RS (RC3) = 0 (indicates command mode)
    PORTD = i;             // Place command on PORTD
    PORTC |= (0x1 << 0);   // Set EN (RC0) = 1 (enable pulse start)
    __delay_ms(100);       // Small delay for command execution
    PORTC &= ~(0x1 << 0);  // Set EN (RC0) = 0 (enable pulse end)
}
```