EXPERIMENT 7 - LCD INTERFACING USING MSP432

LAB CODE AND TITLE : 19CCE283 EMBEDDED COMPUTING LAB

EXPERIMENT NUMBER : 7

DATE - 31/05/2022, TUESDAY

* **AIM:**

Develop a program to interface LCD module using an MSP432 microcontroller. Print characters on the LCD. Also, by supplying varying voltage to an analog pin, read the voltage value and display whether it is low, medium or high.

a) **Hello world display using LCD:**

1. Interface the LCD in 8-bit mode.
2. Make the required function declarations for LCD module and delay
3. In the main program, call function to initialize LCD.
4. In an infinite loop (while), perform the following operations-
   (i) Clear display along with DDRAM contents.
   (ii) Set DDRAM address or cursor position on display.
   (iii) Print "HELLO" on the LCD module.
   (iv) Delay.

I. **Function to configure LCD:-**

1. Set port 4 for LCD data lines (D0 to D7) and P3.5, 6, 7 as output for RS, RW and EN respectively.
2. Function Set: 8-bit, 1 Line, 5x7 Dots
   Function Set: 8-bit, 2 Lines, 5x7 Dots
3. Set the Entry Mode - Cursor Increment (Right Side), Display Shift OFF
4. Clear display (also clear DDRAM contents)
5. Initialize cursor to home position
6. Display ON, Cursor OFF, Blink OFF

PTO

II. Display character using Data Write Function:-

① Set RS = 1, R/W = 0 and move data to P4.

② Give LCD Enable Signal-
(i) Pulse E High
(ii) Clear E
(iii) Wait for controller to do the display

III. Display character using command Write Function :-

① Set RS = 0 and R/W = 0

② Move command to P4 - Put command on the data bus.

③ Give LCD Enable Signal-
(i) Pulse E High
(ii) Clear E
(iii) Wait for controller to do the display

→ C PROGRAM CODE :-

```
#include "msp.h"

// LCD Interfacing - 8-bit mode:-
#define RS 0x20 // Mask P3.5
#define RW 0x40 // Mask P3.6
#define EN 0x80 // Mask P3.7

void LCD_init(void); // Function to configure LCD
void LCD_data(unsigned char data); // Display character using
Data Write Function
void LCD_command(unsigned char command); // Display character
using command write Function
void delayMs(int n); // Delay milliseconds when system clock is at
3MHz for Rev C MCU
```

P10 →

```
// Main Function :-
int main (void)
{
  LCD_init ();  // Call function to initialize LCD

  // Infinite Loop (An embedded program does not stop) :-
  while (1)
  {
    LCD_command (1) ; // Clear display (also clear DDRAM content)
    delayMs (500);
    LCD_command (0x80); // Set DDRAM address or cursor position on display

    // Print "HELLO" on the LCD module :-
    LCD_data ('H'); LCD_data ('E'); LCD_data ('L');
    LCD_data ('L'), LCD_data ('O');
    delayMs (500);
  }
}

// Function to configure LCD :-
void LCD_init (void)
{
    // Set P4 LCD Data (D0 to D7) and P3.5, 6, 7 as output for
RS, RW, EN respectively :-
    P3 -> DIR |= RS|RW|EN;
    P4 -> DIR = 0xFF;
    delayMs (30);

    LCD_command (0x30); // Function Set : 8-bit, 1 Line, 5x7 Dots
    delayMs (10);
    LCD_command (0x30); // Function Set : 8-bit, 1 Line, 5x7 Dots
    delayMs (1);
    LCD_command (0x30); // Function set : 8-bit, 1 Line, 5x7 Dots
```

```
    LCD_command (0x38); // Function set: 8-bit, 2 Lines, 5x7 Dots
    LCD_command (0x01); // Set the entry Mode - Cursor Increment
(Right side), Display shift OFF
    LCD_command (0x01); // clear display (also clear DDRAM contents)
    LCD_command (0x02); // Initialize Cursor to Home Position
    LCD_command (0x0C); // Display ON, Cursor OFF, Blink OFF
}

// Display Character Using Data Write function :-
void LCD_data (unsigned char data)
{
    P3→OUT |= RS; // RS = 1
    P3→OUT &= ~RW; // R/W = 0
    P4→OUT |= data; // Move data to P4;

    // Give LCD Enable Signal :-
    P3→OUT |= EN; // Pulse E HIGH
    delayMs(5);
    P3→OUT &= ~EN; // Clear E
    delayMs(4); // Wait for microcontroller to do the display
}

// Display character using command write function :-
void LCD_command (unsigned char command)
{
  P3→OUT |= ~(RS|RW); // RS = 0, R/W = 0
  P4→OUT |= command; // Move command to P4 - Put command on
the data bus

    // Give LCD Enable Signal :-
    P3→OUT |= EN // Pulse E HIGH
    delayMs(5);
```

```
    P3 -> OUT &= ~EN; // Clear E
    delayMs(4); // Wait for microcontroller to do the display
}

// Delay milliseconds when the system clock is at 3 MHz for
Rev C MCU:-
void delayMs(int n)
{
    int i, j;
    for (j=0; j<n; j++)
      for (i=750; i>0; i--); // Delay of 1ms
}
```

b) **Voltmeter implementation using ADC and LCD :**

① Interface the LCD in 8-bit mode.
② Make the required function declarations for LCD module and delay.
③ In the main program, declare the variables required to print the voltage value on the LCD. Call LCD initialization function.
④ Power ON should be disabled during configuration. sample-and-hold pulse-mode select, sysclk, 32 sample clocks and software trigger.
⑤ Set resolution as 12-bit (14 clock cycle conversion time). A6 input, single-ended and $V_{ref} = AV_{CC}$.
⑥ Configure functionality of P5.1 for ADC pin A6 and its direction as input. To convert for memory register 5, repeat single channel (sequence sample is active) and enable ADC after configuration.
⑦ In an infinite while-loop, perform the following operations -
(i) Start the conversion. Wait for ADC conversion to complete.
(ii) Read ADC data register and calculate the voltage value from the ADC value.
(iii) clear display (also clear DDRAM content) and set DDRAM address or cursor position on display. Delay.
(iv) Based on the result value, print low, high or medium. Delay.

I. Function to configure LCD :-

① Set port 4 for LCD data lines (D0 to D7) and P3.5, 6, 7 as output for RS, RW and EN respectively.

② Function Set : 8-bit, 1 Line, 5X7 Dots
Function Set : 8-bit, 2 Lines, 5X7 Dots

③ Set the entry Mode - Cursor Increment (Right Side), Display shift OFF

④ Clear display (also clear DDRAM contents)

⑤ Initialize cursor to home position.

⑥ Display ON, Cursor OFF, Blink OFF

II. Display character using Data Write Function :-

① Set RS=1, R/W=0 and move data to P4.

② Give LCD Enable Signal-
(i) Pulse E High
(ii) clear E
(iii) wait for controller to do the display.

III. Display character using Command Write Function :-

① Set RS=0 and R/W=0

② Move command to P4 - Put command on the data bus.

③ Give LCD Enable Signal-
(i) Pula E High
(ii) Clear E
(iii) Wait for controller to do the display.

→ C PROGRAM CODE :

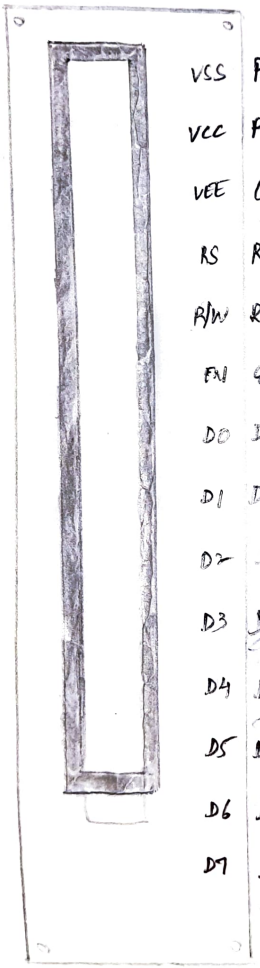
```
#include "mcp.h"
#include <stdio.h>

// LCD Interfacing - 8 bit Mode:-
#define RS 0x20 // Mask P3.5
#define RW 0x40 // mask P3.6
#define EN 0x80 // Mask P3.7
```

```
void LCD_init(void); // Function to configure LCD
void LCD_data(unsigned char data); // Display character using Data Write Function
void LCD_command(unsigned char command); // Display character using Command Write function
void LCD_string(char word[]); // Display string using Data Write Function
void delayMs(int n); // Delay milliseconds when system clock is at 3 MHz for Rev C MCU

// Main Function:
int main(void)
{
    int result; float new_result; char output[4];
    LCD_init(); // call function to initialize LCD

    ADC14->CTL0 = 0x00000010; // Power ON should be disabled during configuration
    ADC14->CTL0 = 0x04080300; // sample-and-hold pulse-mode select, sysclk, 32 sample clocks, software trigger
    ADC14->CTL1 = 0x00000020; // Set resolution as 8-bit (14 clock cycle conversion time)
    ADC14->MCTL[5] = 6; // A6 input, single-ended, Vref = AVcc

    // configure functionality of 5.1 for ADC pin A6 and direction as input :-
    P5->SEL1 |= 0x40;
    P5->SEL0 |= 0x40;

    // Convert for memory register 5:-
    ADC14->CTL1 |= 0x00050000; // Repeat single channel, sequence sample is active
    ADC14->CTL0 |= 2; // Enable ADC after configuration
```

```
// Infinite loop (An embedded program does not stop):
while (1)
{
    ADC14 -> CTL0 |= 1;  // start the conversion now
    while (!ADC14 -> IFGR0);  // Wait for ADC conversion to complete
    result = ADC14 -> MEM[5];  // Read ADC data register.

    new_result = (result/2^12) * 5;  // Calculate the voltage value from the ADC value
    /*
      sprintf stands for "string print".
      Instead of printing on console, it stores output on char buffer which are specified in sprintf.
      That is, returns a formatted string.
    */
    sprintf (output, "%f", new_result);

    LCD_command (1);  // Clear display (also clear DDRAM content)
    delayMs (500);
    LCD_command (0x80);  // Set DDRAM address or cursor position on display (First Line)
    delayMs (500);
    LCD_string (output);
    delayMs (500);
    LCD_command (0xC0);  // Set DDRAM address or cursor position on display (second line)
    delayMs (500);

    if (result > 0 && result <= ((2^12) * (1/3)))
    {
      LCD_data ('L'); LCD_data ('O'); LCD_data ('W');
      delay Ms (500);
    }
```

```
    else if (result > ((2^12) * (1/3)) && result <= ((2^12) * (2/3)))
    {
      LCD_data('M'); LCD_data('E'); LCD_data('D');
      LCD_data('I'); LCD_data('U'); LCD_data('M');
      delayMs(500);
    }

    else
    {
      LCD_data('H'); LCD_data('I'); LCD_data('G'); LCD_data('H');
      delayMs(500);
    }
  }
}

// Function to configure LCD:-
void LCD_init(void)
{
    // set P4 LCD Data (D0 to D7) and P3.5, 6, 7 as output for
RS, RW, EN respectively :-
    P3->DIR |= RS|RW|EN;
    P4->DIR = 0xFF;
    delayMs(30);

    LCD_command(0x30); // Function set: 8-bit, 1 line, 5x7 Dots
    delayMs(10);
    LCD_command(0x30); // Function set: 8-bit, 1 line, 5x7 Dots
    delayMs(1);
    LCD_command(0x30); // Function set: 8-bit, 1 line, 5x7 Dots
    LCD_command(0x38); // Function set: 8-bit, 2 lines, 5x7 Dots
    LCD_command(0x06); // set the Entry Mode - Cursor Increment
(right side), Display shift OFF
```

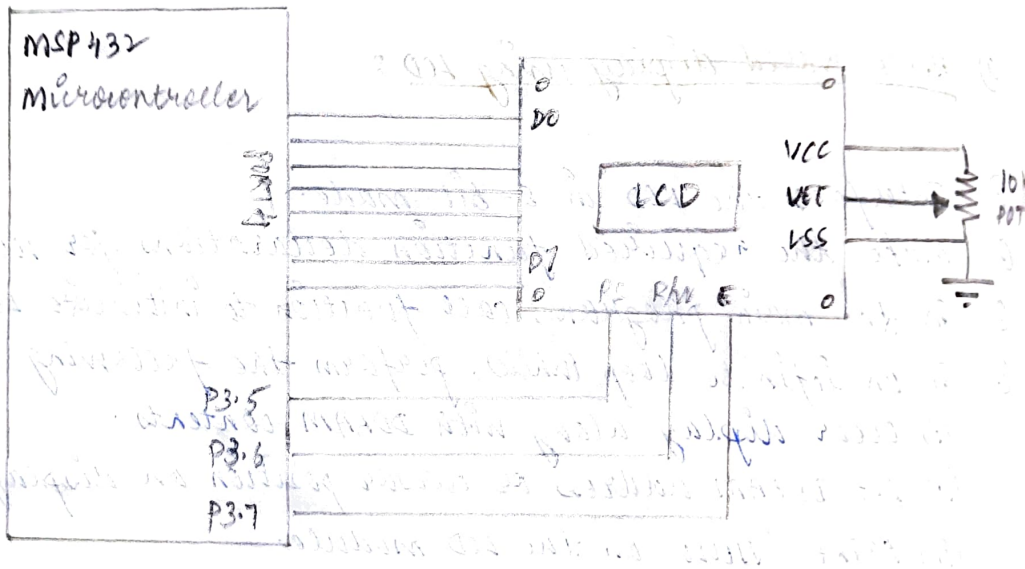
```
    LCD_command(0x01); // Clear display (also clear DDRAM contents)
    LCD_command(0x02); // Initialize Cursor to Home Position
    LCD_command(0x0C); // Display ON, Cursor OFF, Blink OFF
}

// Display character using Data Write Function :-
void LCD_data(unsigned char data)
{
    P3->OUT |= RS;  // RS = 1
    P3->OUT &= ~RW; // R/W = 0
    P4->OUT |= data; // Move data to P4;

    // Give LCD Enable Signal :-
    P3->OUT |= EN; // Pulse E High
    delayMs(5);
    P3->OUT &= ~EN; // Clear E
    delayMs(5); // wait for microcontroller to do the display
}

// Display character using command Write Function :-
void LCD_command(unsigned char command)
{
    P3->OUT |= ~(RS|RW); // RS = 0, R/W = 0
    P4->OUT |= command; // Move command to P4 - Put command on the data bus

    // Give LCD Enable signal :-
    P3->OUT |= EN; // Pulse E High
    delayMs(5);
    P3->OUT &= ~EN; // Clear E
    delayMs(4); // wait for microcontroller to do the display
}
```



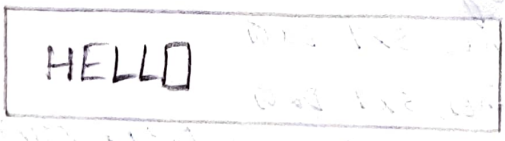
VSS	Power supply (GND)
VCC	Power supply (+5V)
VEE	Contrast adjust control
RS	Register select { 0 = Instruction input, 1 = Data input
R/W	Read/Write { 0 = Write to LCD module, 1 = Read from LCD module
EN	Enable signal
D0	Data bus line 0 (LSB)
D1	Data bus line 1
D2	Data bus line 2
D3	Data bus line 3
D4	Data bus line 4
D5	Data bus line 5
D6	Data bus line 6
D7	Data bus line 7 (MSB)

* Components Required -

① MSP432 Microcontroller (TI)
② LCD module (16 x 2)
③ Bread Board
④ 10K Potentiometer
⑤ Power Supply / Battery
⑥ Connecting Wires, as required
⑦ Resistor



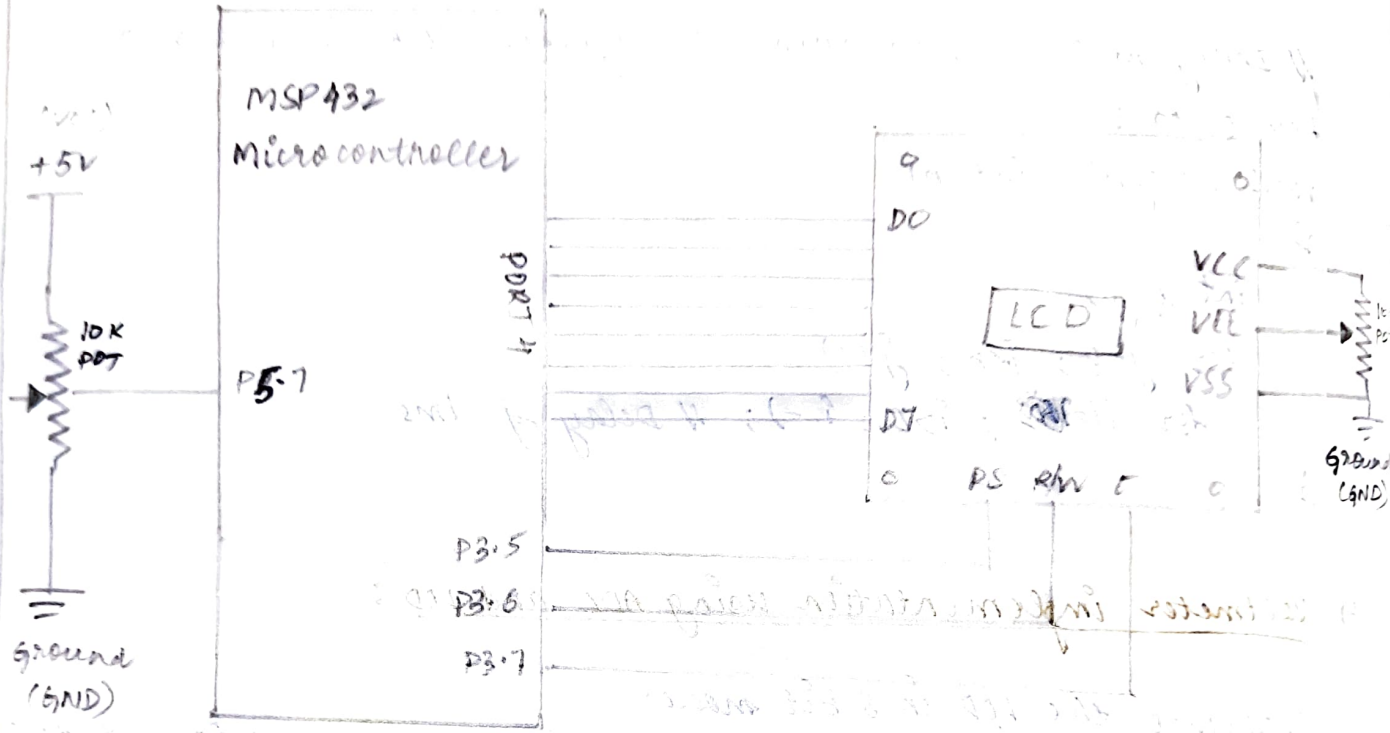
(a) Hello World display using LCD



LCD module display

* Components required -

1. MSP432 microcontroller
2. LED Module (16×2)
3. Bread Board
4. Two 10 K Potentiometer
5. Power Supply / Battery
6. Connecting wires, as required.
7. Resistor



(b) Voltmeter implementation using ADC and LCD

LCD module display - All three cases.

```
// Display string using Data Write function:-
void LCD_string (char word[])
{
  int i;
  for (i=0; word[i] != '\0'; i++)
  {
    LCD_data (word[i]);
  }
}

// Delay milliseconds when system clock is at 3 MHZ for Rev C MCU:
void delayMs (int n)
{
  int i, j;
  for (j=0; j< n; j++)
    for (i= 750; i>0; i--); // Delay of 1 ms
}
```

+ **RESULT:**

Thus, developed a program to interface LCD module using an MSP432 microcontroller. All simulation results were verified successfully.