

Lab 12

Interfacing ADC with msp430



Spring 2025

Submitted by: Mohsin Sajjad

Registration No: 22pwsce2149

Class Section: A

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

A handwritten signature in black ink that reads "Mohsin Sajjad".

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (01 06, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

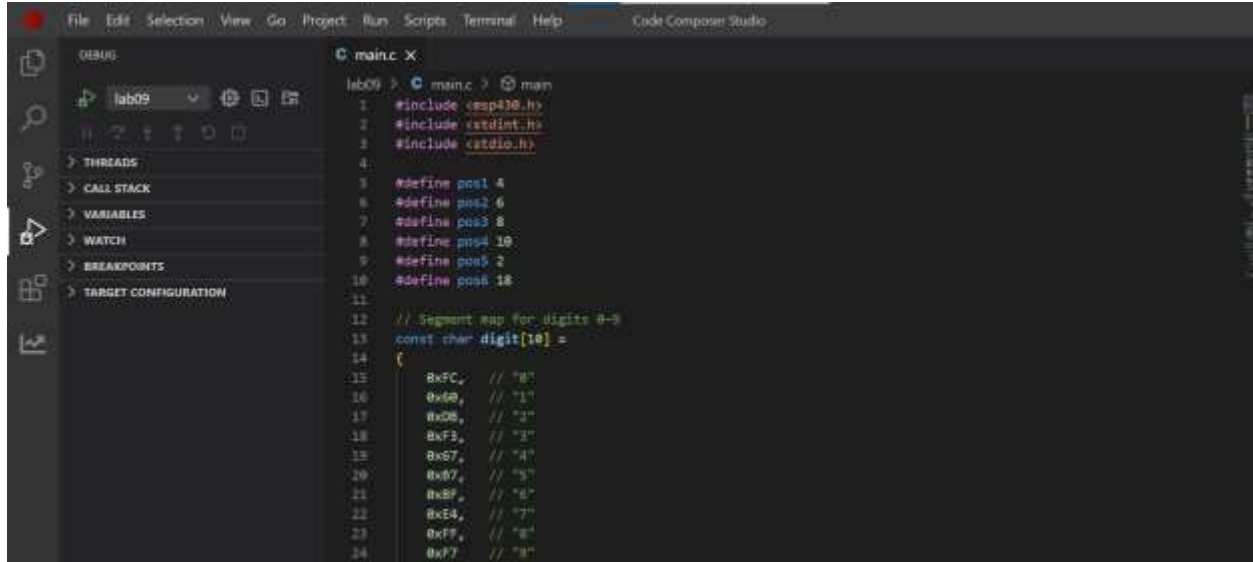
Interfacing ADC with msp430

TASKS:

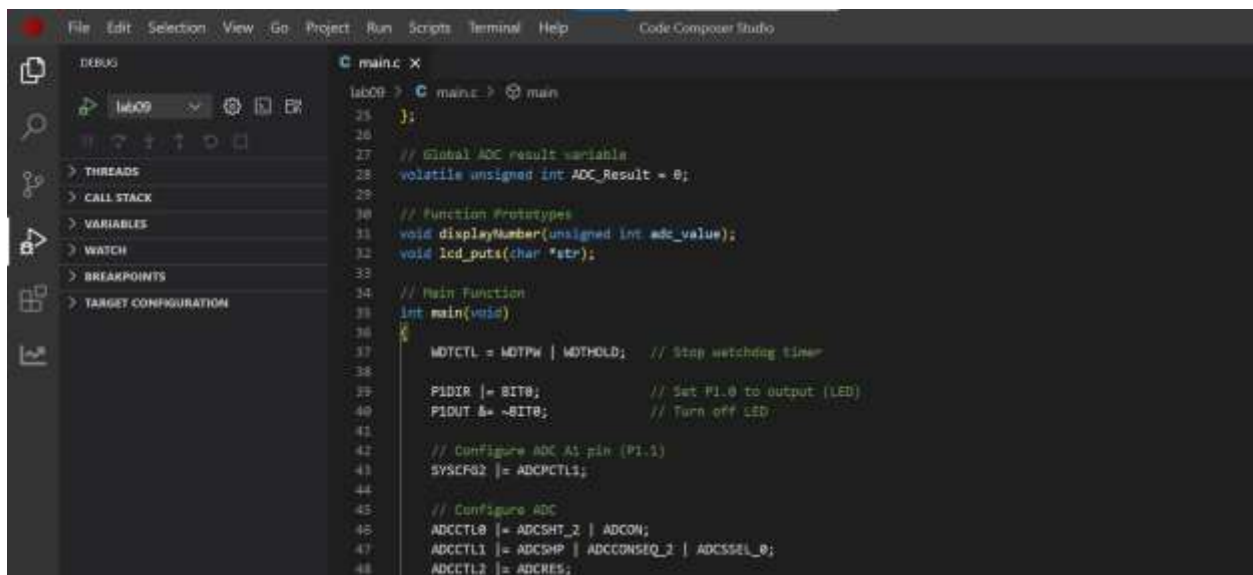
TASK 01:

Run the code given in the lecture

CODD:



```
File Edit Selection View Go Project Run Scripts Terminal Help Code Composer Studio
lab09
> THREADS
> CALL STACK
> VARIABLES
> WATCH
> BREAKPOINTS
> TARGET CONFIGURATION
C main.c X
lab09 > C main.c > main
1 #include <msp430.h>
2 #include <stdint.h>
3 #include <stdio.h>
4
5 #define pos1 4
6 #define pos2 6
7 #define pos3 8
8 #define pos4 10
9 #define pos5 2
10 #define pos6 18
11
12 // Segment map for digits 0-9
13 const char digit[10] =
14 {
15     0xFC, // "0"
16     0x60, // "1"
17     0x0B, // "2"
18     0xF3, // "3"
19     0x67, // "4"
20     0x6F, // "5"
21     0xBF, // "6"
22     0xE4, // "7"
23     0xFF, // "8"
24     0xPF, // "9"
```



```
File Edit Selection View Go Project Run Scripts Terminal Help Code Composer Studio
lab09
> THREADS
> CALL STACK
> VARIABLES
> WATCH
> BREAKPOINTS
> TARGET CONFIGURATION
C main.c X
lab09 > C main.c > main
25 }
26
27 // Global ADC result variable
28 volatile unsigned int ADC_Result = 0;
29
30 // Function Prototypes
31 void displayNumber(unsigned int adc_value);
32 void led_puts(char *str);
33
34 // Main Function
35 int main(void)
36 {
37     MDCTL = MDTPW | MDTHOLD; // Stop watchdog timer
38
39     P1DIR |= BIT0; // Set P1.0 to output (LED)
40     P1OUT &= ~BIT0; // Turn off LED
41
42     // Configure ADC A1 pin (P1.1)
43     SYSCFG2 |= ADCMCTL1;
44
45     // Configure ADC
46     ADCCTL0 |= ADCSHT_2 | ADCON;
47     ADCCTL1 |= ADCSHP | ADCCONSEQ_2 | ADCSSEL_0;
48     ADCCTL2 |= ADCRES;
```

The screenshot shows the Code Composer Studio interface with the 'main.c' file open. The code configures the ADC module, including setting the clock source to HCLK, enabling the ADC, and setting the internal reference voltage to 1.5V. The ADC is configured to use the internal 1.5V reference and the ADC12_1 module.

```

lab09 > C main.c > @ main
90  ADCCTL0 |= ADCINH1 | ADCREF_1;
91  ADCIE |= ADCIE0;
92
93  // Timer A1 for ADC trigger
94  TA1CCR0 = 1024 - 1;
95  TA1CCR1 = 512 - 1;
96  TA1CTL1 = OUTMOD_4;
97  TA1CTL = TASSEL_ACLK | MC_1 | TACLR;
98
99  // Internal reference voltage = 1.5V
100  PMPCCTL0_H = PMPCCTL0_H;
101  PMPCCTL3 |= INTREFEN;
102  __delay_cycles(400);
103
104  ADCCTL0 |= ADCENC;
105
106  __bis_SR_register(LPM0_bits | GIE);
107  __no_operation();
108
109  // ADC ISR
110  #pragma vector=ADC_VECTOR
111  __interrupt void ADC_ISR(void)
112  {
113

```

The screenshot shows the implementation of the ADC interrupt service routine (ADC_ISR) and a helper function, displayNumber. The ISR checks if the ADC result is within a specific range and updates the P1OUT register accordingly. The displayNumber function converts the ADC result to a string and displays it on the LCD.

```

lab09 > C main.c > @ main
113  switch (__even_in_range(ADCIV, ADCIV_ADCIF0))
114  {
115      case ADCIV_ADCIF0:
116          ADC_Result = ADCMEMB;
117          displayNumber(ADC_Result);
118
119          if (ADCMEMB < 8x155)
120              P1OUT &= ~BIT0;
121          else
122              P1OUT |= BIT0;
123          break;
124      default:
125          break;
126  }
127
128  // Function to convert ADC result to string and show on LCD
129  void displayNumber(unsigned int value)
130  {
131      char buffer[5];
132
133      buffer[0] = (value / 1000) % 10 + '0';
134      buffer[1] = (value / 100) % 10 + '0';
135      buffer[2] = (value / 10) % 10 + '0';
136

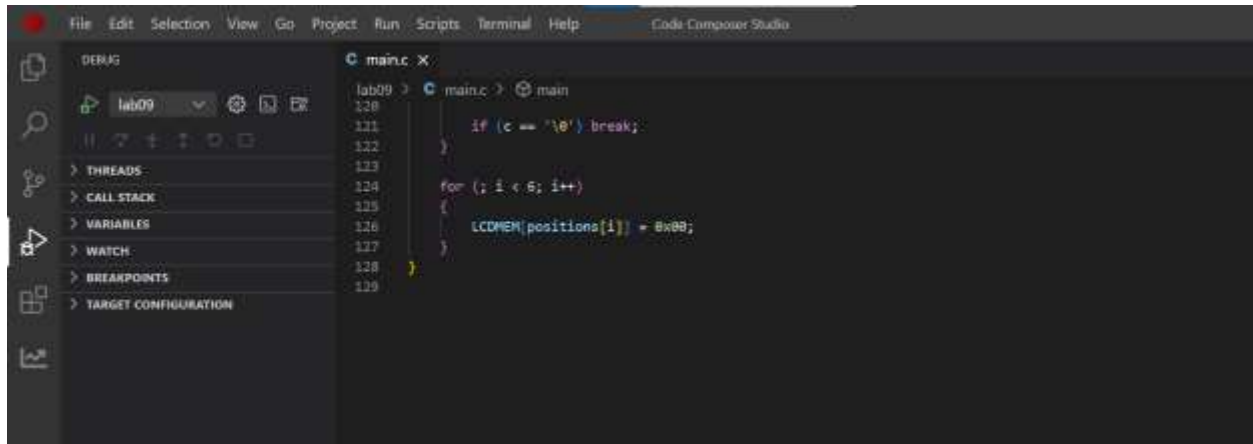
```

The screenshot shows the implementation of the LCD display function, lcd_puts. The function takes a string and displays it on the LCD, with positions for each character defined in a constant array.

```

lab09 > C main.c > @ main
137      buffer[3] = (value % 10) + '0';
138      buffer[4] = '\0';
139
140      lcd_puts(buffer);
141  }
142
143  // Show up to 6 characters on LCD
144  void lcd_puts(char *str)
145  {
146      const uint8_t positions[6] = {pos1, pos2, pos3, pos4, pos5, pos6};
147      int i;
148
149      for (i = 0; i < 6; i++)
150      {
151          char c = str[i];
152          if (c >= '0' && c <= '9')
153          {
154              LCDMEM[positions[i]] = digit[c - '0'];
155          }
156          else
157          {
158              LCDMEM[positions[i]] = 0x00; // Blank
159          }
160      }
161  }

```



TASK02:

From temperature sensor (LM35) read temperature and convert it into Digital value by using ADC0804 and display the value on the LCD. In LCD at first line write your registration Number and on the second line display the value of the temperature sensor attached with ADC0804. (use proteus can also use MSP430FR4133)

CODE:

```

main.c
1 #include <msp430.h>
2 #include <stdio.h>
3
4 #define ADC_DATA_IN P1IN
5 #define ADC_WR_PIN BIT5
6 #define ADC_RD_PIN BIT4
7 #define ADC_INTR_PIN BIT2
8
9 #define LCD_DATA_OUT P2OUT
10 #define LCD_RS_PIN BIT7
11 #define LCD_E_PIN BIT6
12
13 void delay_us(unsigned int us) {
14     while (us--) {
15         __delay_cycles(3);
16     }
17 }
18
19 void delay_ms(unsigned int ms) {
20     while (ms--) {
21         __delay_cycles(3000);
22     }
23 }
24
25 void writcmd(unsigned char cmd) {
26     P3OUT&= ~LCD_RS_PIN;
27     LCD_DATA_OUT= cmd;
28     P3OUT= LCD_E_PIN;
29     delay_us(50);
30     P3OUT&= ~LCD_E_PIN;
31     delay_us(50);
32 }
33
34 void writedata(unsigned char data) {
35     P3OUT= LCD_RS_PIN;
36     LCD_DATA_OUT= data;
37     P3OUT= LCD_E_PIN;

```

```

38     delay_us(50);
39     P3OUT&= ~LCD_E_PIN;
40     delay_us(50);
41 }
42
43 void lcdinit(void) {
44     delay_ms(20);
45     writecmd(0x38);
46     delay_ms(2);
47     writecmd(0x0C);
48     delay_ms(2);
49     writecmd(0x01);
50     delay_ms(2);
51     writecmd(0x06);
52     delay_ms(2);
53 }
54
55 unsigned int adc_read_raw(void) {
56     unsigned char raw;
57     P3OUT&= ~ADC_WR_PIN;
58     delay_us(2);
59     P3OUT|= ADC_WR_PIN;
60     while (P3IN & ADC_INTR_PIN) {
61     }
62     P3OUT&= ~ADC_RD_PIN;
63     delay_us(2);
64     raw = ADC_DATA_IN;
65     P3OUT|= ADC_RD_PIN;
66     return (unsigned int)raw;
67 }
68
69 double adc_to_celsius(unsigned int raw) {
70     return ((double)raw * 1.953125);
71 }
72
73
74 int temp_times10 = (int)(temp * 10 + 0.5);
75 int integer_part = temp_times10 / 10;
76 int decimal_part = temp_times10 % 10;
77 char buff[4];
78 int idx = 0;
79 if (integer_part >= 100) {
80     buff[idx++] = '0' + (integer_part / 100);
81     buff[idx++] = '0' + ((integer_part / 10) % 10);
82     buff[idx++] = '0' + (integer_part % 10);
83 } else if (integer_part >= 10) {
84     buff[idx++] = '0' + (integer_part / 10);
85     buff[idx++] = '0' + (integer_part % 10);
86 } else {
87     buff[idx++] = '0' + integer_part;
88 }
89 buff[idx] = '\0';
90 {
91     int i;
92     for (i = 0; i < idx; i++) {
93         writedata(buff[i]);
94     }
95 }
96 writedata(' ');
97 writedata('0' + decimal_part);
98 writedata(0xDF);
99 writedata(' ');
100 }
101
102 int main(void) {
103     WDTCTL= WDTPW|WDTHOLD;
104     P1DIR= 0x00;
105     P1REN= 0x00;
106     P2DIR= 0xFF;
107     P2OUT= 0x00;
108     P3DIR= (ADC_RD_PIN|ADC_WR_PIN|LCD_E_PIN|LCD_RS_PIN);
109     P3DIR&= ~ADC_INTR_PIN;

```

Output:

