EXPERIMENT 4 : ANALOG TO DIGITAL CONVERSION

LAB TITLE AND CODE : EMBEDDED COMPUTING LAB (19CCE283)
EXPERIMENT NUMBER : 4
DATE : 10/04/2022 , TUESDAY

★ AIM :
Perform analog to digital conversion using MSP43n by configuring its on-chip peripherals.

★ ALGORITHM : (ADC PERIPHERAL PROGRAMMING AND DIGITAL OUTPUT DISPLAY ON LED)

① Configure functionality of P2.0, P2.1 and P2.2 as simple GPIO pins.
② Configure direction of P2.0, P2.1 and P2.2 as output for tri-color LEDs.
③ Power ON should be disabled during configuration.
④ sample-and-hold pulse-mode select, sysclk, 32 sample clocks and software trigger.
⑤ Set resolution as 12-bit (14 clock cycle conversion time)
⑥ A6 input, single-ended and $V_{ref} = AV_{CC}$.
⑦ Configure functionality of P4.7 for ADC pin A6 and its direction as input.
⑧ To convert for memory register 5, repeat single channel (sequence sample as active) and enable ADC after configuration.
⑨ In an infinite while-loop, perform the following operations -
(i) start the conversion.
(ii) Wait for ADC conversion to complete
(iii) Read ADC data register.
(iv) Display bits 10, 9, 8 on tri-color LEDs.

★ CODE : (ADC PERIPHERAL PROGRAMMING AND DIGITAL OUTPUT DISPLAY ON LED)

```
#include "msp.h"

//main Function:
int main(void)
{
```

```
int result;

P2 -> SEL0 &= ~7; // Configure functionality of P2.0, P2.1 and P2.2 as simple GPIO pins
P2 -> SEL1 &= ~7;
P2 -> DIR |= 7; // Configure direction of P2.0, P2.1 and P2.2 as output for tri-color LEDs

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 12-bit (14 clock cycle conversion time)
ADC14 -> MCTL[5] = 6; // A6 input, single-ended, vref = AVcc

// Configure functionality of P4.7 for ADC pin A6 and direction as input :-
P4 -> SEL1 |= 0x80;
P4 -> SEL0 |= 0x80;

// 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
  P2 -> OUT = result >> 8; // Display bits 10:8 on tri-color LEDs
}
```

CIRCUIT DIAGRAM :

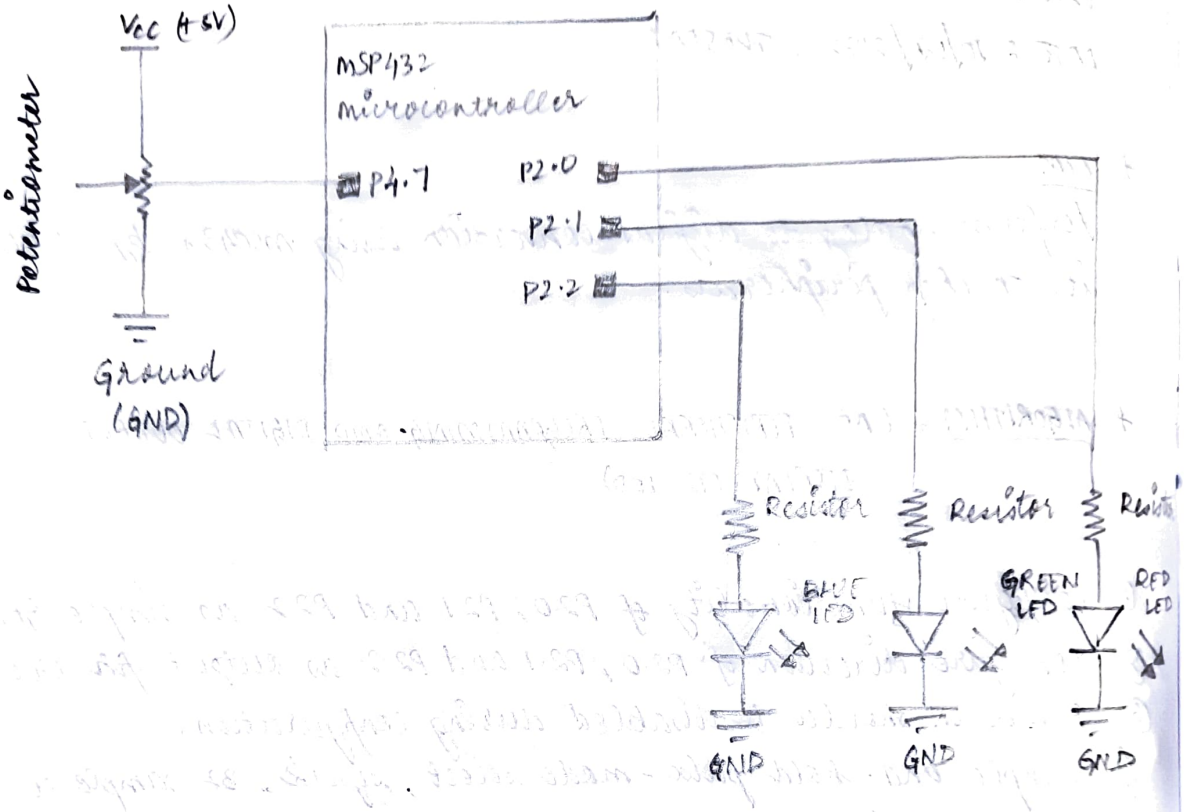

Figure - ADC Peripheral programming and Digital Output Display on LED

* ALGORITHM : (TEMPERATURE CONTROLLER IMPLEMENTATION USING ADC)

① Configure functionality of P2.1 as simple GPIO pins.
② Configure direction of P2.1 as output for GREEN-LED
③ Power ON should be disabled during configuration.
④ sample-and-hold pulse-mode select, sysclk, 32 sample clocks and software trigger.
⑤ Set resolution as 8-bit (9 clock cycle conversion time)
⑥ Set A6 input, single-ended, $V_{REF} = AV_{CC}$
⑦ Configure functionality of P4.1 as ADC pin and it's direction as input.
⑧ To convert for memory register 5, repeat single channel (sequence sample is active) and enable ADC after conversion.
⑨ In an infinite while-loop, perform the following operations -
(i) start the conversion now
(ii) Wait for ADC conversion to complete
(iii) Read ADC data register
(iv) Display bit for GREEN-LED
(v) If temperature of the room goes above 28°, turn ON the GREEN-LED; else OFF.

* CODE : (TEMPERATURE CONTROLLER IMPLEMENTATION USING ADC)

```
#include "msp.h"

// main Function :
int main (void)
{
  int result;

  P2->SEL0 &= ~2; // Configure functionality of P2.1 as simple GPIO pins
  P2->SEL1 &= ~2;
  P2->DIR |= 2; // Configure direction of P2.1 as output for GREEN-LED
```

```
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 = 0x00000000; // set resolution as 8-bit (9 clock cycle conversion time)
ADC14 -> MCTL[5] = 6; // A6 input, single-ended, Vref = AVcc

// Configure functionality of P4.7 as ADC pin and it's direction as input :-
P4 -> SEL1 |= 0x80;
P4 -> SEL0 |= 0x80;

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

// 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
  P2 -> OUT = result >> 8; // display bit for GREEN-LED

  /*
  Temperature Range - 0 degrees (0x00) to 60 degrees (0x11)
  If temperature goes above 28 degrees, turn ON the LED; else OFF.
  60 degrees = 256 bits (2^8 bit-resolution)
  Therefore, 28 degrees = ??? = (256 * 28) / 60 = 119.4667 ~ 120 bits
  In hexadecimal representation, (120)10 = (78)16
  */
```

* CIRCUIT DIAGRAM :

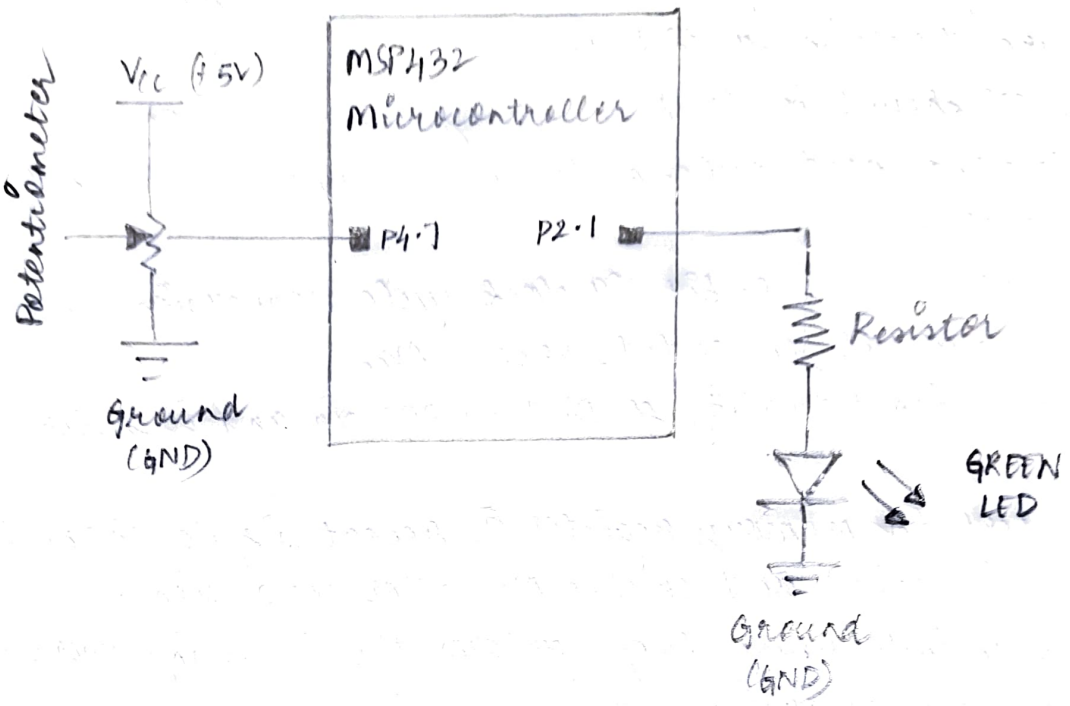

Figure. Temperature controller Implementation using ADC

```
        if (result >> 0x18)
        {
          P2->OUT |= 2; // TURN ON P2.1 GREEN-LED
        }
        else
        {
          P2->OUT &= ~2; // TURN OFF P2.1 GREEN-LED
        }
    }
}
```

* **RESULT:**

Implemented analog to digital conversion using MSP432 by configuring its on-chip peripherals and all simulation results were verified successfully