Potmeter

Aim: To read potentiometer voltage via ADC and display the result in millivolts on a seven-segment display.

Theory:

A potentiometer is a variable resistor used to adjust voltage by changing resistance through a sliding contact (wiper). It acts as a voltage divider, providing a variable output based on wiper position. Commonly used in volume controls or fine-tuning circuits, it limits current to protect components from excess flow.

- **ADC Setup**: The code initializes the ADC to read an analog signal from AIN0, using GPIO pins for alternate functions.
- **Data Processing**: After ADC conversion, the result is split into individual digits (num1, num2, num3, num4) to be displayed on the SSD.
- **SSD Display**: The function shift_out1() manages shifting data bits to the SSD to show the calculated millivolt value.
- **Timing**: A delay function provides a 1-second pause to allow clear visualization of the displayed result.

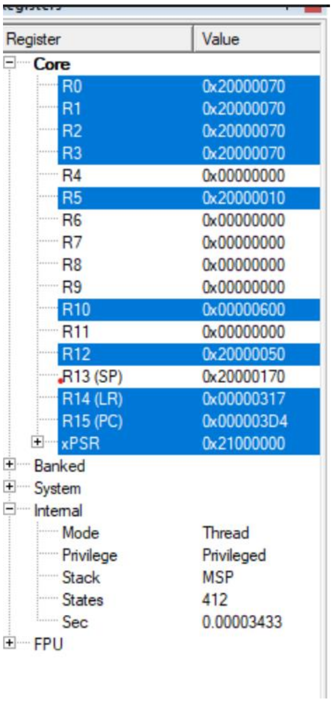
Code:

```
/* potmeter  10k resistor at max , current flowing through resistor is 0.4095ma current flows. value
displayed on ssd id in mv */


#include "inc\tm4c123gh6pm.h"

#include <stdbool.h>

#include <stdint.h>

void delayMs(int n)

        {

        int i,j;

        for(i=0;i<n;i++)

        for(j=0;j<4180;j++)

        {}
```

```
}
void shift_out1(unsigned char str );
volatile int num1,num2,num3,num4;

int main(void)
{
   volatile  int result;
              unsigned char a[16] =
{0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6,0xEE,0x3E,0x9C,0x7A,0x9E,0x8E};

   /* enable clocks */
   SYSCTL_RCGCGPIO_R |= 0x08;   /* enable clock to GPIOE (AIN0 is on PE3) */
   SYSCTL_RCGCADC_R |= 1;       /* enable clock to ADC0 */
   SYSCTL_RCGCGPIO_R |= 0x10;
   GPIO_PORTE_DIR_R |= 0x1F;
   GPIO_PORTE_DEN_R |= 0x1F;
   /* initialize PE3 for AIN0 input  */
   GPIO_PORTD_AFSEL_R |= 8;       /* enable alternate function */
   GPIO_PORTD_DEN_R &= ~8;        /* disable digital function */
   GPIO_PORTD_AMSEL_R |= 8;       /* enable analog function */

  //unsigned int i;
   /* initialize ADC0 */
  ADC0_ACTSS_R &= ~1;        /* disable SS3 during configuration */
  ADC0_EMUX_R &= ~0x000F;    /* software trigger conversion */
  ADC0_SSMUX0_R &= ~0xFFFFFFFF;
            ADC0_SSMUX0_R |= 0x04;         /* get input from channel 0 */
   ADC0_SSCTL0_R |= 0x06;       /* take one sample at a time, set flag at 1st sample */
  ADC0_ACTSS_R |= 0x01;          /* enable ADC0 sequencer 3 */

   while(1)
   {
       ADC0_PSSI_R |= 1;        /* start a conversion sequence 3 */
       while((ADC0_RIS_R & 1) == 0);   /* wait for conversion complete */
       result = ADC0_SSFIFO0_R; /* read conversion result */
       ADC0_ISC_R = 1;          /* clear completion flag */


    num1 = result%10;//copies data from the specified location
                         result = result/10;
              num2 = result%10;
      result = result/10;
              num3 = result%10;
                         result = result/10;
     num4 = result%10;
                                  shift_out1(a[num1]);
                                  shift_out1(a[num2]);
                                  shift_out1(a[num3]);
                                  shift_out1(a[num4]);
                                  shift_out1(0x00);
              delayMs(1000);
                         }
              }



void shift_out1(unsigned char str)
{
unsigned char j=0,check;

              for(j=0;j<=7;j++)
              {
                         GPIO_PORTE_DATA_R = 0x00;       //PE3 pin(sclk) is low (0000 0000)
                         check = (str &(1<<j));
```

```
                    if(check)

                              GPIO_PORTE_DATA_R =  0x04;      //PE2 pin(sdat) is high (0000 0100)


else

                                              GPIO_PORTE_DATA_R |=  0x00;

           GPIO_PORTE_DATA_R |=  0x08;  //PE3 pin(sclk) is high (0000 1000)

                             GPIO_PORTE_DATA_R |=  0x10;

}

}
```

Result:



Register	Value
Core	
R0	0x20000070
R1	0x20000070
R2	0x20000070
R3	0x20000070
R4	0x00000000
R5	0x20000010
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000600
R11	0x00000000
R12	0x20000050
R13 (SP)	0x20000170
R14 (LR)	0x00000317
R15 (PC)	0x000003D4
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	412
Sec	0.00003433
FPU