EE2016_MUP_LAB_EXPERIMENT_08_ADC / DAC Implementation

MIRUDHULA J | EE23B046

OBJECTIVES:

- To understand C-interfacing (use C-programming) in an ARM platform
- To study and implement ADC / DAC in ARM platform

Tasks:

4.1 ADC Given a real-time (analog) signal from a sensor, convert it into a digital signal (Implement an ADC). Decrease the step size? Do you see any change in the bits used to represent the whole range? What is the quantization error?
4.2 DAC Given the LPC2148 ARM development board and given program for square wave, modify it to generate

1. RAMP (saw tooth) wave
2. Triangular wave
3. Sine wave (using lookup table or using formulae)

CODE USED:

ADC:

```
#include <LPC214x.H>                        /* LPC214x
definitions */

#include "delay.h"

unsigned long Read_ADC0(unsigned char);
void Init_ADC0(unsigned char);

#ifndef __ADC_H
#define __ADC_H


#define CHANNEL_0   0
#define CHANNEL_1   1
#define CHANNEL_2   2
#define CHANNEL_3   3
#define CHANNEL_4   4
#define CHANNEL_5   5
#define CHANNEL_6   6
#define CHANNEL_7   7
```

```
/* Crystal frequency,10MHz~25MHz should be the same as actual status.
*/
#define Fosc          12000000   /* 12 MHz is the operational
frequency of o/p dgtlClk */
#define ADC_CLK       1000000         /* set to 1Mhz */

/* A/D Converter 0 (AD0) */
#define AD0_BASE_ADDR      0xE0034000
#define ADC_INDEX               4

#define ADC_DONE                0x80000000
#define ADC_OVERRUN             0x40000000


#define ADC_FullScale_Volt 3.3  // 3.3V - ADC Referance Voltage
#define ADC_FullScale_Count     1024 // 2^10 - 10 bit ADC
#define LED_IOPIN          IO0PIN
#define BIT(x)  (1 << x)

#define LED_D0  (1 << 10)       // P0.10 mapping same as in Exp7
switch LED
#define LED_D1  (1 << 11)       // P0.11
#define LED_D2  (1 << 12)       // P0.12
#define LED_D3  (1 << 13)       // P0.13

#define LED_D4  (1 << 15)       // P0.15
#define LED_D5  (1 << 16)       // P0.16
#define LED_D6  (1 << 17)       // P0.17
#define LED_D7  (1 << 18)       // P0.18
#define LED_DATA_MASK           ((unsigned long)((LED_D7 | LED_D6 |
LED_D5 | LED_D4 | LED_D3 | LED_D2 | LED_D1 | LED_D0)))

#define LED1_ON      LED_IOPIN |= (unsigned long)(LED_D0);      //
LED1 ON
#define LED2_ON      LED_IOPIN |= (unsigned long)(LED_D1);      //
LED2 ON
#define LED3_ON      LED_IOPIN |= (unsigned long)(LED_D2);      //
LED3 ON
#define LED4_ON      LED_IOPIN |= (unsigned long)(LED_D3);      //
LED4 ON
#define LED5_ON      LED_IOPIN |= (unsigned long)(LED_D4);      //
LED5 ON
#define LED6_ON      LED_IOPIN |= (unsigned long)(LED_D5);      //
LED6 ON
#define LED7_ON      LED_IOPIN |= (unsigned long)(LED_D6);      //
LED7 ON
```

```
#define LED8_ON      LED_IOPIN |= (unsigned long)(LED_D7);      //
LED8 ON

#endif
     #ifndef LED_DRIVER_OUTPUT_EN
#define LED_DRIVER_OUTPUT_EN (1 << 5)// P0.5
#endif
//LED definitions




int main (void)
{

     unsigned long ADC_val;

     Init_ADC0(CHANNEL_1);
     Init_ADC0(CHANNEL_2);

     delay_mSec(100);

     IO0DIR |= LED_DATA_MASK;              // GPIO Direction control
-> pin is output
     IO0DIR |= LED_DRIVER_OUTPUT_EN;       // GPIO Direction control
-> pin is output
     IO0CLR |= LED_DRIVER_OUTPUT_EN;


     while(1)
     {
          //ADC_val = Read_ADC0(CHANNEL_1);
          ADC_val = Read_ADC0(CHANNEL_2);
          ADC_val=(ADC_val>>6);
          delay_mSec(5);



     if(ADC_val & BIT(0)) LED8_ON;
     if(ADC_val & BIT(1)) LED7_ON;
     if(ADC_val & BIT(2)) LED6_ON;
     if(ADC_val & BIT(3)) LED5_ON;

//   if(ADC_val & BIT(4)) LED4_ON;
//   if(ADC_val & BIT(5)) LED3_ON;
//   if(ADC_val & BIT(6)) LED2_ON;
```

```
//    if(ADC_val & BIT(7)) LED1_ON;
      }



//     return 0;
}

void Init_ADC0(unsigned char channelNum)
{
      if(channelNum == CHANNEL_1)
            PINSEL1 = (PINSEL1 & ~(3 << 24)) | (1 << 24);         //
P0.28 -> AD0.1

      if(channelNum == CHANNEL_2)
            PINSEL1 = (PINSEL1 & ~(3 << 26)) | (1 << 26);         //
P0.29 -> AD0.2

      if(channelNum == CHANNEL_3)
            PINSEL1 = (PINSEL1 & ~(3 << 28)) | (1 << 28);         //
P0.30 -> AD0.3


    AD0CR = ( 0x01 << 1 ) |                              // SEL=1,
select channel 0, 1 to 4 on ADC0
                (( Fosc / ADC_CLK - 1 ) << 8 ) |      // CLKDIV =
Fpclk / 1000000 - 1
                ( 0 << 16 ) |                                 // BURST =
0, no BURST, software controlled
                ( 0 << 17 ) |                                 // CLKS =
0, 11 clocks/10 bits
                ( 1 << 21 ) |                                 // PDN =
1, normal operation
                ( 0 << 22 ) |                                 // TEST1:0
= 00
                ( 0 << 24 ) |                                 // START =
0 A/D conversion stops
                ( 0 << 27 );                                  /* EDGE =
0 (CAP/MAT singal falling,trigger A/D conversion) */
}
unsigned long Read_ADC0( unsigned char channelNum )
{
    unsigned long regVal, ADC_Data;

     /* Clear all SEL bits */
    AD0CR &= 0xFFFFFF00;
     /* switch channel, start A/D convert */
```

```
    AD0CR |= (1 << 24) | (1 << channelNum);


    /* wait until end of A/D convert */
     while ( 1 )     {

//         regVal = *(volatile unsigned long *)(AD0_BASE_ADDR +
ADC_INDEX);
           regVal = AD0GDR;

           if ( regVal & ADC_DONE ){
           break;
           }
    }

     /* stop ADC now */
    AD0CR &= 0xF8FFFFFF;
     /* save data when it's not overru otherwise, return zero */
    if ( regVal & ADC_OVERRUN ) {
          return ( 0 );
    }
    ADC_Data = ( regVal >> 6 ) & 0x3FF;
    /* return A/D conversion value */
     return ( ADC_Data );

}

void delay_mSec(int dCnt)       // pr_note:~dCnt mSec
{
  int j=0,i=0;

  while(dCnt--)
  {
       for(j=0;j<1000;j++)
       {
         /* At 60Mhz, the below loop introduces
         delay of 10 us */
         for(i=0;i<10;i++);
       }
  }
}
```

OUTPUT:

https://drive.google.com/drive/folders/1zx-CUgiSATQDpOpEM-yAXk2aOkaWWoEG?usp=sharing

10 bit adc: step-size = V/2^10 = V/1024
4 bit adc: step-size = V/2^4 = V/16

Change in Bits to Represent the Range:

- **With a 10-bit ADC, you have 1024 levels (from 0 to 1023).**
- **With a 4-bit ADC, you have only 16 levels (from 0 to 15)**

Quantization Error: step-size/2
On reducing the number of bits to represent a signal the step-size increases, therefore the quantization error also increases.

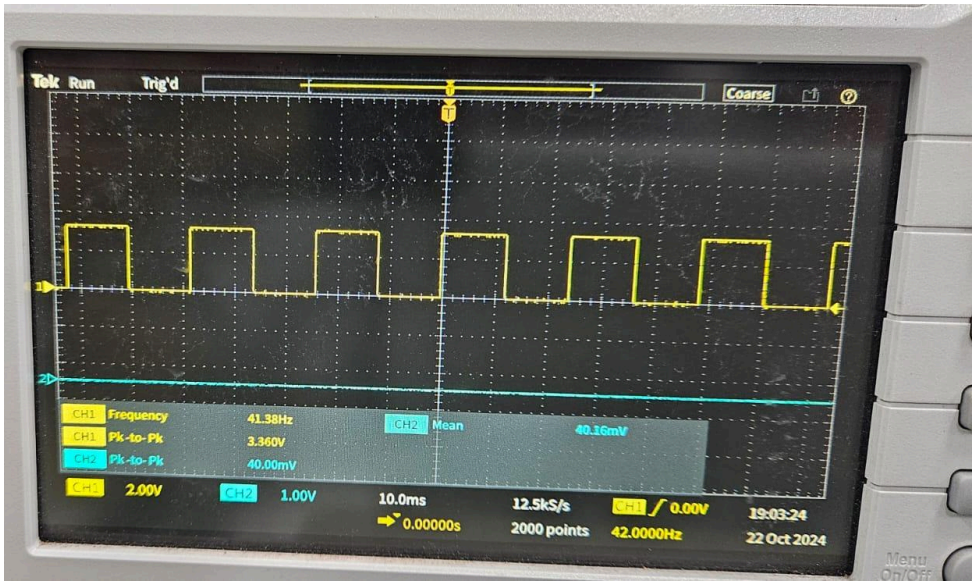
DAC:

1)**Square wave**

```
 //DAC program for LPC2148 SUNTECH KIT


#include "LPC214x.H"                        /* LPC214x definitions */

#define DAC_BIAS                0x00010000
void mydelay(int);

void DACInit( void )
{
    /* setup the related pin to DAC output */
     PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000; /* set p0.25 to DAC output */
    return;
}

int main (void)
{
     DACInit();

//SQUARE WAVE
while(1)
{
     DACR=0|DAC_BIAS;
       mydelay(0x0F);
       DACR=(0x3FF<<6)|DAC_BIAS;
       mydelay(0x0f);
}
     return 0;
```

```
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}
```

OUTPUT:

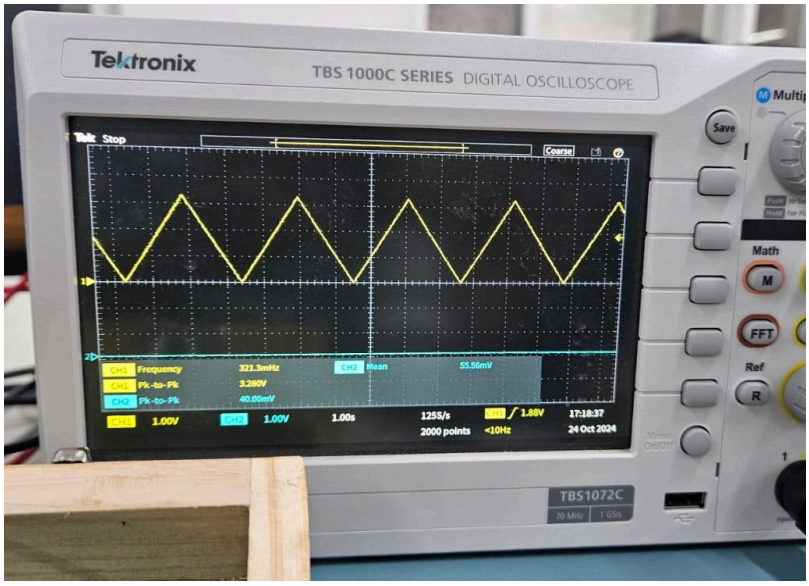


2)Triangular wave:

```
//DAC program for LPC2148 SUNTECH KIT


#include "LPC214x.H"                        /* LPC214x definitions */

#define DAC_BIAS                0x00010000
void mydelay(int);

void DACInit( void )
{
    /* setup the related pin to DAC output */
     PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000; /* set p0.25 to DAC output */
    return;
```

```
}

int main(void)
{
    DACInit();
    unsigned int i;

    while (1)
    {
        // Ramp up
        for (i = 0; i <= 0x3FF; i++)
        {
            DACR = (i << 6) | DAC_BIAS;
            mydelay(0x01);
        }

        // Ramp down
        for (i = 0x3FF; i > 0; i--)
        {
            DACR = (i << 6) | DAC_BIAS;
            mydelay(0x01);
        }
    }
}
```

OUTPUT:

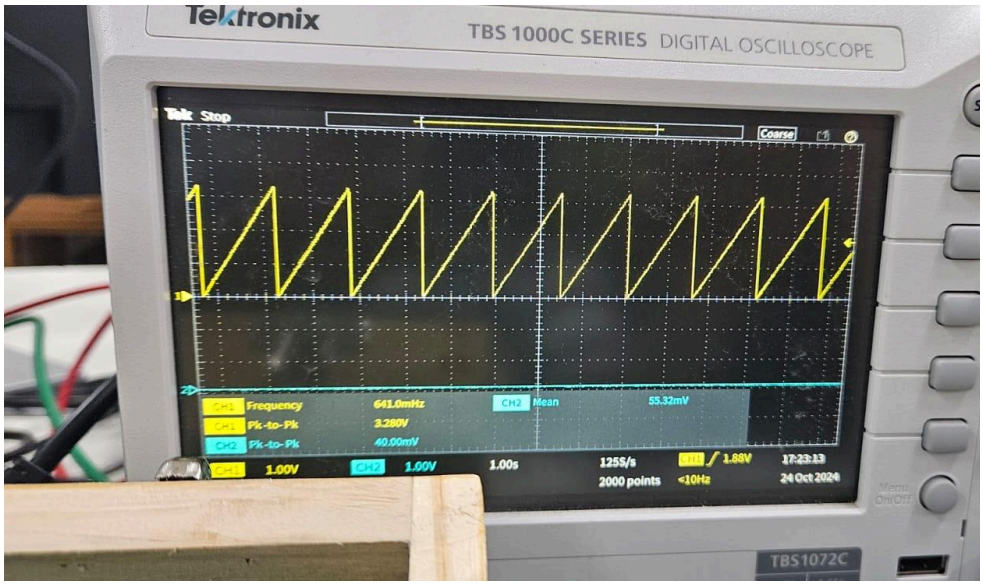


3)Ramp wave:

```
//DAC program for LPC2148 SUNTECH KIT
```

```
#include "LPC214x.H"                        /* LPC214x definitions */

#define DAC_BIAS                0x00010000
void mydelay(int);

void DACInit( void )
{
    /* setup the related pin to DAC output */
     PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000; /* set p0.25 to DAC output */
    return;
}

int main(void)
{
    DACInit();
    unsigned int i;

    while (1)
    {
        // Ramp up
        for (i = 0; i <= 0x3FF; i++)
        {
            DACR = (i << 6) | DAC_BIAS;
            mydelay(0x01);
        }

        // Ramp down
        for (i = 0x3FF; i > 0; i--)
        {
            DACR = (i << 6) | DAC_BIAS;
            mydelay(0x01);
        }
    }
}
```

OUTPUT:

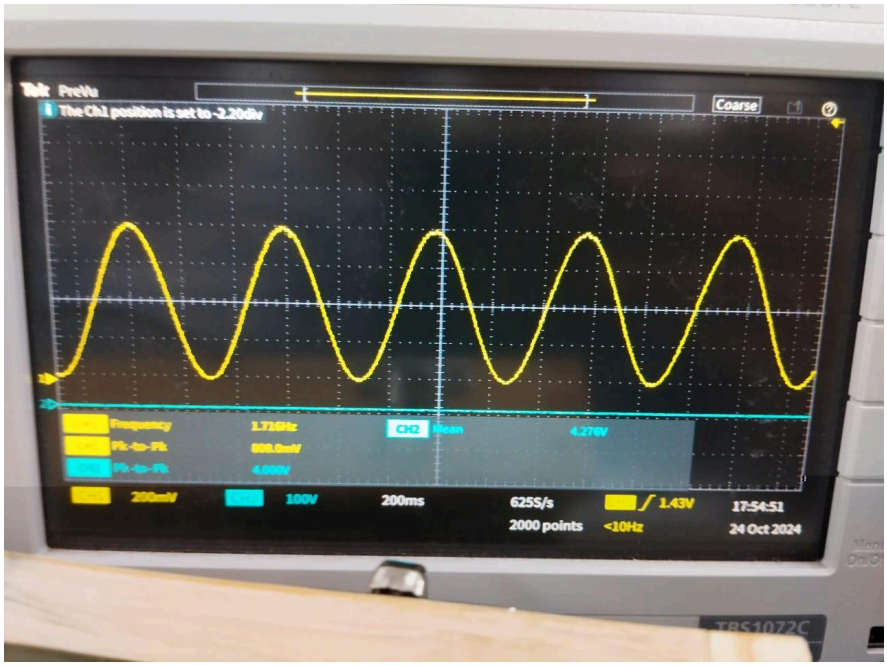


4)Sine wave:

```
//DAC program for LPC2148 SUNTECH KIT


#include "LPC214x.H"                        /* LPC214x definitions */

#define DAC_BIAS                0x00010000
void mydelay(int);

void DACInit( void )
{
    /* setup the related pin to DAC output */
     PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000; /* set p0.25 to DAC output */
    return;
}

#define SINE_TABLE_SIZE 256

unsigned int sine_table[SINE_TABLE_SIZE] = {

127,130,133,136,139,143,146,149,152,155,158,161,164,167,170,173,176,1
78,181,184,187,190,192,195,198,200,203,205,208,210,212,215,217,219,22
1,223,225,227,229,231,233,234,236,238,239,240,

242,243,244,245,247,248,249,249,250,251,252,252,253,253,253,254,254,2
54,254,254,254,254,253,253,253,252,252,251,250,249,249,248,247,245,24
4,243,242,240,239,238,236,234,233,231,229,227,225,223,

221,219,217,215,212,210,208,205,203,200,198,195,192,190,187,184,181,1
```

```
78,176,173,170,167,164,161,158,155,152,149,146,143,139,136,133,130,12
7,124,121,118,115,111,108,105,102,99,96,93,90,87,84,81,78,

76,73,70,67,64,62,59,56,54,51,49,46,44,42,39,37,35,33,31,29,27,25,23,
21,20,18,16,15,14,12,11,10,9,7,6,5,5,4,3,2,2,1,1,1,0,0,0,0,0,0,0,1,1,
1,2,2,3,4,5,5,6,7,9,10,11,12,14,15,16,18,20,21,23,25,27,29,31,

33,35,37,39,42,44,46,49,51,54,56,59,62,64,67,70,73,76,78,81,84,87,90,
93,96,99,102,105,108,111,115,118,121,124
};

int main(void)
{
    DACInit();
    unsigned int i = 0;

    while (1)
    {
        DACR = (sine_table[i] << 6) | DAC_BIAS;  // Output sine wave
value to DAC
        mydelay(0x02);  // Adjust delay to control the frequency

        i++;
        if (i >= SINE_TABLE_SIZE)
            i = 0;  // Wrap around the table
    }
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}
```

OUTPUT:



PROCEDURE FOLLOWED:

- Wrote the C programs for the above problems (one at a time).
- Entered the above program in KEIL software, edit and compile / assemble.
- Then the generated hex file is uploaded into the Flash Magic.
- Then it is programmed into the LPC2148 board.
- The probes are connected to the oscillator and to the LPC2148 output.
- The wave form can be seen in the oscilloscope.
- Finally, demonstrated its working, to TA

MY CONTRIBUTION:

- I was responsible for dac ramp, square and triangular wave generation question coding and implementation.