EE2016 : Microprocessor Theory and Lab

Lab Experiment # 8

Serial Communication & ADC / DAC Implementation in ViARM 2378 Development Board (through C - Interface)

Batch 35

Sarathchandra Koppera EE21B073

Vikas Gollapalli EE21B051

AIM:

1. To understand C-interfacing (use C-programming) in an ARM platform

2. To study and implement serial communication in ARM platform

3. To study and implement ADC / DAC in ARM platform

Tasks to be done:

1.Serial Communication:

Write a program (in C) to display the ASCII code in LEDs, corresponding to the key pressed in the key board of the PC interfaced to ViARM-2378. Use the RS232 serial cable interfaced to the Vi Microsystem's ViARM 2378 development board

2.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?

3.DAC

Given the ViARM2378 ARM development board, generate

1. Square wave

2. Triangular wave

3. Sine wave (using lookup table)

Task-1

Code:

```
#include "LPC23xx.h"

/***************************
****** Routine to set processor and peripheral clock *******
***************************/
void TargetResetInit(void)
{
    // 72 MHz Frequency
    if((PLLSTAT&0x02000000)>0)
    {
        /* If the PLL is already running */
        PLLCON&=~0x02; /* Disconnect the PLL */
        PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
        PLLFEED=0x55;
    }
    PLLCON&=~0x01; /* Disable the PLL */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    SCS&=~0x10; /* OSCRANGE=0, Main OSC is between 1 and 20 Mhz */
    SCS|=0x20; /* OSCEN=1, Enable the main oscillator */
    while((SCS&0x40)==0);
    CLKSRCSEL=0x01; /* Select main OSC, 12MHz, as the PLL clock source */
    PLLCFG=(24<<0)|(1<<16); /* Configure the PLL multiplier and divider */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    PLLCON|=0x01; /* Enable the PLL */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    CCLKCFG=3; /* Configure the ARM Core Processor clock divider */
    USBCLKCFG=5; /* Configure the USB clock divider */
    while((PLLSTAT&0x04000000)==0);
    PCLKSEL0=0xAAAAAAAA; /* Set peripheral clocks to be half of main clock
*/
    PCLKSEL1=0x22AAA8AA;
    PLLCON|=0x02; /* Connect the PLL. The PLL is now the active clock
source */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    while((PLLSTAT&0x02000000)==0);
    PCLKSEL0=0x55555555;         /* PCLK is the same as CCLK */
    PCLKSEL1=0x55555555;
}
/***************************
******** Serial Reception Routine **********
***************************/
int serial_rx(void)
{
    while(!(U0LSR&0x01));
    return(U0RBR);
}
/***************************
******** Serial Transmission Routine *********
***************************/
void serial_tx(int ch)
{
```

```
    while((U0LSR&0x20)==0x00);
    U0THR=ch;
}
/***************************
**** Serial Transmission Routine for a string of characters ****
***************************/
void string_tx(char* a)
{
    while(*a!='\0')
    {
        while((U0LSR&0X20)!=0X20);
        U0THR=*a;
        a++;
    }
}
/***************************
********* Main Routine ***************
***************************/
int  main()
{

    unsigned int Fdiv;
    char value;
    TargetResetInit();
    FIO3DIR=0xFF; // Setting the LEDs on the board as output
    /******* UART1 Initialization *********/
    PINSEL0=0x00000050;
    U0LCR=0x83; // 8 bits, no Parity, 1 Stop bit
    Fdiv=(72000000/16)/19200 ;  // Baud rate
    U0DLM=Fdiv / 256;
    U0DLL=Fdiv % 256;
    U0LCR=0x03; // DLAB=0
    while(1)
    {
        value=serial_rx(); // Receiving value from the computer, i.e., key
press
        FIO3PIN=value;  // Displaying the value on the LEDs as output
        serial_tx(value);  // Transmitting the value received by the
processor back to computer to verify if correct data was received by it.
    }
    return 0;
}
```

Output:https://drive.google.com/drive/folders/1YnGlFSesC_GBXuRiaA4bwC3iaArDOUWj?usp=share_link

Observation:

Here in serial communication, we are giving an ASCII letter from keyboard. Now after successful completion it will display the ASCII letter that we pressed on the keyboard on the monitor of our PC. Also, LEDs on the LPC2378 will glow according to the binary value of the ASCII letter which we give from keyboard.

Task-2

Code:

```
#include "LPC23xx.h"

/***************************
****** Routine to set processor and peripheral clock ******
***************************/
void TargetResetInit(void)
{
    // 72 Mhz Frequency
    if ((PLLSTAT&0x02000000)>0)
    {
        /* If the PLL is already running */
        PLLCON&=~0x02; /* Disconnect the PLL */
        PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
        PLLFEED=0x55;
    }
    PLLCON&=~0x01; /* Disable the PLL */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    SCS&=~0x10; /* OSCRANGE=0, Main OSC is between 1 and 20 Mhz */
    SCS|=0x20; /* OSCEN=1, Enable the main oscillator */
    while((SCS&0x40)==0);
    CLKSRCSEL=0x01; /* Select main OSC, 12MHz, as the PLL clock source */
    PLLCFG=(24<<0)|(1<<16); /* Configure the PLL multiplier and divider */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    PLLCON|=0x01; /* Enable the PLL */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    CCLKCFG=3; /* Configure the ARM Core Processor clock divider */
    USBCLKCFG=5; /* Configure the USB clock divider */
    while((PLLSTAT&0x04000000)==0);
    PCLKSEL0=0xAAAAAAAA; /* Set peripheral clocks to be half of main clock
*/
    PCLKSEL1=0x22AAA8AA;
    PLLCON|=0x02; /* Connect the PLL. The PLL is now the active clock
source */
    PLLFEED=0xAA; /* PLL register update sequence, 0xAA, 0x55 */
    PLLFEED=0x55;
    while((PLLSTAT&0x02000000)==0);
    PCLKSEL0=0x55555555;         /* PCLK is the same as CCLK */
    PCLKSEL1=0x55555555;
}
/***************************
******** Serial Transmission Routine *********
***************************/
void serial_tx(int ch)
{
    while ((U0LSR&0x20)!=0x20);
    U0THR=ch;
}
/***************************
********* Hex to ASCII Routine ***********
***************************/
int atoh(int ch)
{
```

```
    if(ch<=0x09)
        ch=ch+0x30;
    else
        ch=ch+0x37;
    return(ch);
}
/**************************
********* Main Routine ***************
**************************/
int main()
{
    unsigned int Fdiv,value,i,j;
    TargetResetInit();

    PCONP|=0X00001000; // switch ADC from disable state to enable state
    PINSEL0=0x00000050; // Pinselection for UART TX and RX lines
    PINSEL1=0X01554000; // Pinselection for ADC0.0
    /******* UART Initialization ***********/
    U0LCR=0x83; // 8 bits, no Parity, 1 Stop bit
    Fdiv=(72000000/16)/19200 ; // Baud rate
    U0DLM=Fdiv/256;
    U0DLL=Fdiv%256;
    U0LCR=0x03; // DLAB=0
    AD0CR=0X01210F01; // ADC initialization
    while(1)
    {
        while((AD0DR0&0X80000000)!=0X80000000){}; // Wait here until ADC
make conversion complete
        /** To get converted value and display it on the serial port **/
        value=(AD0DR0>>6)& 0x3ff ;    //ADC value
        serial_tx('\t');
        serial_tx(atoh((value&0x300)>>8));
        serial_tx(atoh((value&0xf0)>>4));
        serial_tx(atoh(value&0x0f));
        serial_tx(0x0d);
        serial_tx(0x0a);
        for(i=0;i<=0xFF;i++)
            for(j=0;j<=0xFF;j++);
    }
    return 0;
}
```

Output:https://drive.google.com/drive/folders/1YnGlFSesC_GBXuRiaA4bwC3iaArDOUWj?usp=share_link

Explanation:

On increasing the step size, we observe that the change in the analog value required to reflect a change in the digital value becomes larger. On decreasing the number of bits to be used in the digital value, the maximum digital value becomes lesser. This is expected as we are reducing the range of the digital values.

Task-3's Codes:

Sine Wave:

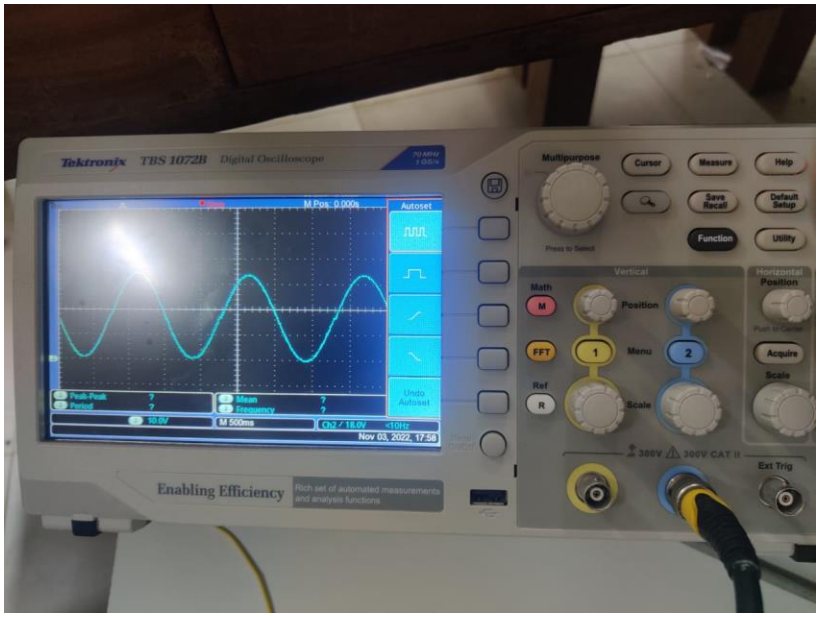
```
#include "LPC23xx.h"
int sin_wave[101]={0x200,0x220,0x240,0x25f,0x27f,
    0x29e,0x2bc,0x2d9,0x2f6,0x312,0x32c,0x346,0x35e,0x374,
    0x38a,0x39d,0x3af,0x3c0,0x3ce,0x3db,0x3e6,0x3ef,0x3f6,
    0x3fb,0x3fe,0x3ff,0x3fe,0x3fb,0x3f6,0x3ef,0x3e6,0x3db,
    0x3ce,0x3c0,0x3af,0x39d,0x38a,0x374,0x35e,0x346,0x32c,
    0x312,0x2f6,0x2d9,0x2bc,0x29e,0x27f,0x25f,0x240,0x220,
    0x200,0x1df,0x1bf,0x1a0,0x180,0x161,0x143,0x126,0x109,
    0xed,0xd3,0xb9,0xa1,0x8b,0x75,0x62,0x50,0x3f,0x31,0x24,
    0x19,0x10,0x9,0x4,0x1,0x0,0x1,0x4,0x9,0x10,0x19,0x24,
    0x31,0x3f,0x50,0x62,0x75,0x8b,0xa1,0xb9,0xd3,0xed,0x109,
    0x126,0x143,0x161,0x180,0x1a0,0x1bf,0x1df,0x200};


void dLAY(int n)
{
    int i,j;
    for(i=0;i<n;i++)
        for(j=0;j<0x0F;j++);
}

int main (void)
{
    PCLKSEL0=0x00C00000;
    PINMODE1=0x00300000;
    PINSEL1=0x00200000;
    int value;
    int i=0;

    //  Lookup Table for sine values

    while(1)
    {
        //Sine Wave
        i=0;
        while(i<101)
        {
            value=sin_wave[i];
            DACR=(value<<6);
            dLAY(100);
            i++;
        }
    }
    return 0;
}
```

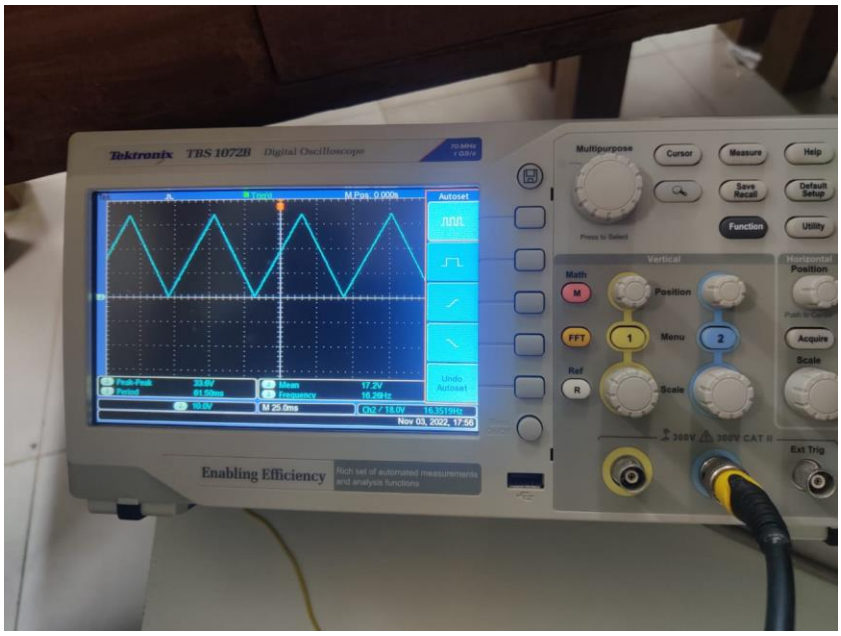


Sine Wave

Triangular Wave:

```
#include "LPC23xx.h"
void dLAY(int n)
{
int i,j;
for(i=0;i<n;i++)
for(j=0;j<0x0F;j++);
}
int main (void)
{
PCLKSEL0=0x00C00000;
PINMODE1=0x00300000;
PINSEL1=0x00200000;
int value;
int i=0;
while(1)
{
//Triangular Wave
value=0;
while(value!=1023)
{
DACR=((1<<16)|(value<<6));
value++;
}
14
while (value!= 0)
{
DACR=((1<<16)|(value<<6));
value--;
}
}
return 0;
```

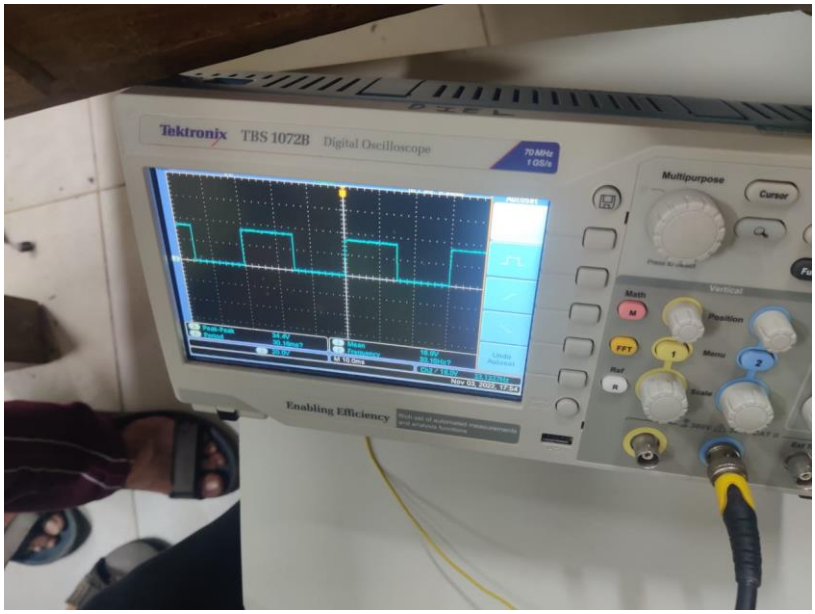
```
}
```



Triangular Wave

Square Wave:

```
#include "LPC23xx.h"
void dLAY(int n)
{
int i,j;
for(i=0;i<n;i++)
for(j=0;j<0x0F;j++);
}
int main (void)
{
PCLKSEL0=0x00C00000;
PINMODE1=0x00300000;
PINSEL1=0x00200000;
int value;
int i=0;
while(1)
{
//Square Wave
value=1023;
DACR=(value<<6);
dLAY(100);
value=0;
DACR=(value<<6);
dLAY(100);
}
return 0;
}
```



Square Wave

Learning Outcomes:

1. By doing this experiment it helped us to get familiar with the ARM assembly language and its commands and how to work with it.
2. Got familiar with using Kiel software
3. It helped in thinking about the problem in different ways so that correct output could be obtained.
4. It helped us to gain experience of flashing the programme on to the ViRAM2378 and then operate it using the teramax application