EE2016: MICROPROCESSOR THEORY AND LAB

EXPERIMENT #8 REPORT

EE21B016 ANANYA DAS
EE21B098 SAILENDRA
B44

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:

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.

```
#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)!=0x20);
  while ((U0LSR & 0x20)==0);
  U0THR = ch;
}
// serial transmission routine for 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();

/***************************** 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();
serial_tx(value+2);
}
       return 0;
}
```

Video link :

https://drive.google.com/file/d/1iwSokxMf-ZZit8p39MKCEJcC5S2ZE-Uo/view?usp=share_link

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 ?.

```
#include "LPC23xx.h"

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;
}
/***************************** Routine for converting hex value to ascii value
*****************/
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;
// char value;
TargetResetInit();
// init_timer( ((72000000/100) - 1) );

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
//Fdiv = ( 72000000 / 16 ) / 2400 ;  //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(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;
}
```

Video link :

https://drive.google.com/file/d/1JiIoBIIErBtxApMs7TuNwYGJ7S8wUksH/view?usp=share_link

https://drive.google.com/file/d/1JiIoBIIErBtxApMs7TuNwYGJ7S8wUksH/view?usp=share_link

3) DAC

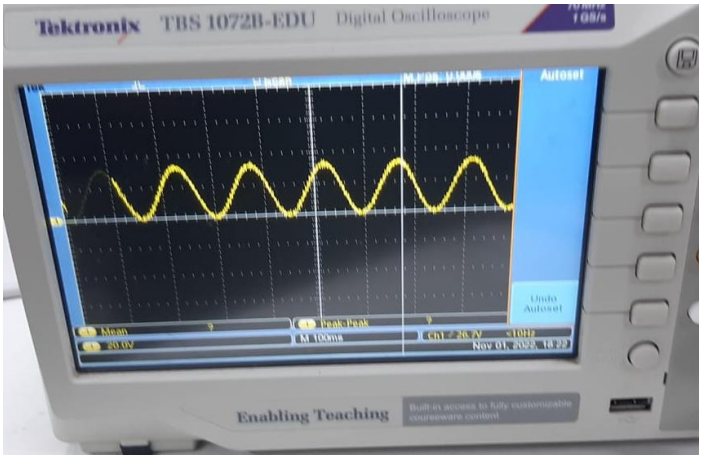
Given the ViARM2378 ARM development board, generate

1. Square wave
2. Triangular wave
3. Sine wave

Sine wave

```
#include "LPC23xx.h"
unsigned int sinevalue[]={512,611,707,796,873,937,984,1013,
    1023,1013,984,937,873,796,707,611,
    512,412,316,227,150,86,39,10,
    0,10,39,86,150,227,316,412}; //DECLARE THE
//ARRAY OUTSIDE MAIN FUNCTION


void delay(unsigned int k)
{
unsigned int i,j;
for(i=0;i<=k;i++)
for(j=0;j<=0xFF;j++);
}

int  main ()
```

```
{
  int i;
PINSEL1 = 0x00200000;
    PCLKSEL0 = 0x00C00000;
    PINMODE1=0x00300000;
while(1)
 {



for(i=0;i<=31;i++)
{
    DACR = (sinevalue[i]<<6 ) ; // ENCLOSE IN BRACKET
  delay(0x01);  //GIVE SOME DELAY FOR THE DAC TO SETTLE
}


}
return 0;
}
```



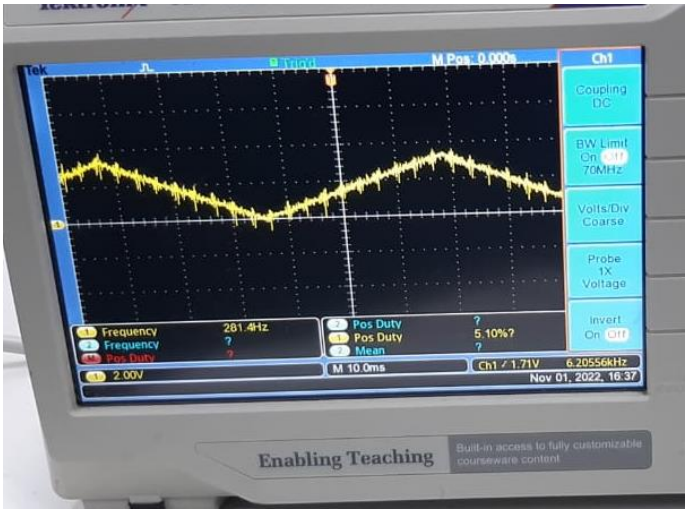
Video link:

https://drive.google.com/file/d/1g8kybjN9uzPg-6JQeHnjFvdaaRFB-EXt/view?usp=share_link

Triangular wave

```
#include "LPC23xx.h"
void delay(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)
   {
      value=0;
      while(value!=1023)
      {
         DACR=((1<<16)|(value<<6));
         value++;
      }
      while (value!= 0)
      {
         DACR=((1<<16)|(value<<6));
         value--;
      }
```

```
    }
    return 0;
}
```



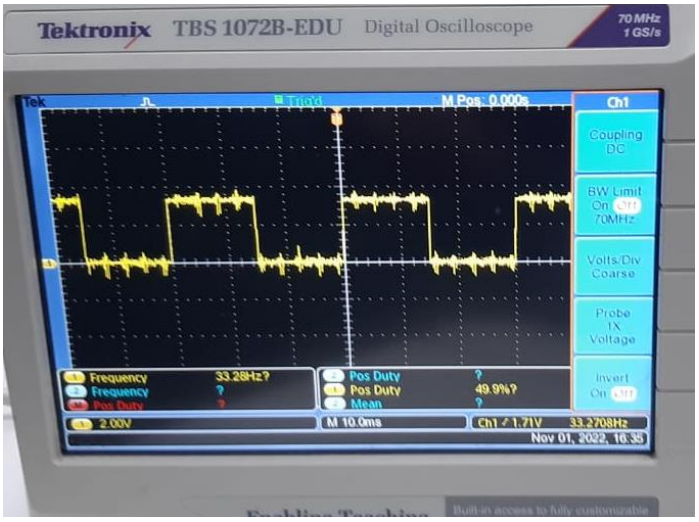
Video link:

https://drive.google.com/file/d/1-KwzlWWS6lYEBEuZ_D4jmYgL32RrkX6Z/view?usp=share_link

Square wave

```
#include "LPC23xx.h"
void delay(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)
    {
        value=1023;
        DACR=(value<<6);
        delay(100);
        value=0;
        DACR=(value<<6);
        delay(100);
    }
    return 0;
}
```



Video link:

https://drive.google.com/file/d/19BxWlLX_dWFkecnR67M570Ir1fvbuq2Y/view?usp=share_link