List of Experiments

Hours/Week: 04
CIE Marks: 40
Semester: 4

Exam Hours: 03
Total Hours: 36
SEE Marks: 60

Sl No.	PART A (Conduct the following experiments by writing program using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool)	Page No.
1.	Write a program to multiply two 16 bit binary numbers.	
2.	Write a program to find the sum of first 10 integer numbers.	
3.	Write a program to find factorial of a number.	
4.	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.	
5.	Write a program to find the square of a number (1 to 10) using look-up table.	
6.	Write a program to find the largest/smallest number in an array of 32 numbers	
7.	Write a program to arrange a series of 32 bit numbers in ascending/descending order.	
8.	Write a program to count the number of ones and zeros in two consecutive memory locations.	
	PART B (Conduct the following experiments on an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.)	
9.	Display "Hello World" message using Internal UART.	
10.	Interface and Control a DC Motor.	
11.	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.	
12.	Determine Digital output for a given Analog input using Internal ADC of ARM controller.	
13.	Interface a DAC and generate Triangular and Square waveforms.	
14.	Interface a 4x4 keyboard and display the key code on an LCD.	
15.	Demonstrate the use of an external interrupt to toggle an LED On/Off.	
16.	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between.	

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with

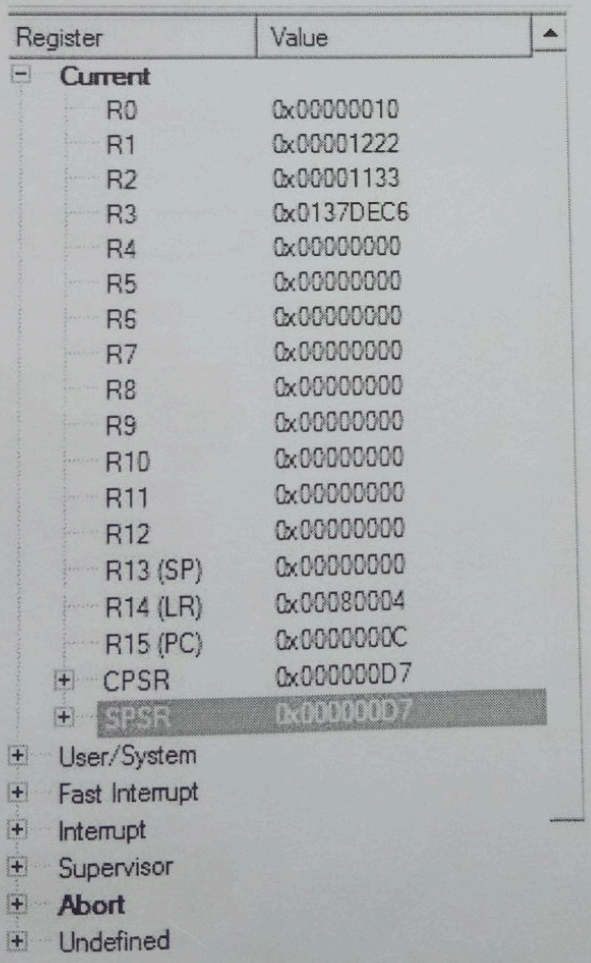
PART A

1. Write a program to multiply two 16 bit binary numbers.

```
        AREA Multiply, CODE, READONLY
        ENTRY
        LDR         R0, =NUM        ; load address of multiplicand
        LDRH        R1, [R0]        ; load First number
        LDRH        R2, [R0,#2]     ; load Second number
        MUL         R3, R1, R2      ; R3 = R1 x R2
STOP B STOP                         ; all done
NUM  DCW  0X1222,0X1133             ; Declaration of no's to be multiply

        END
```

Output:

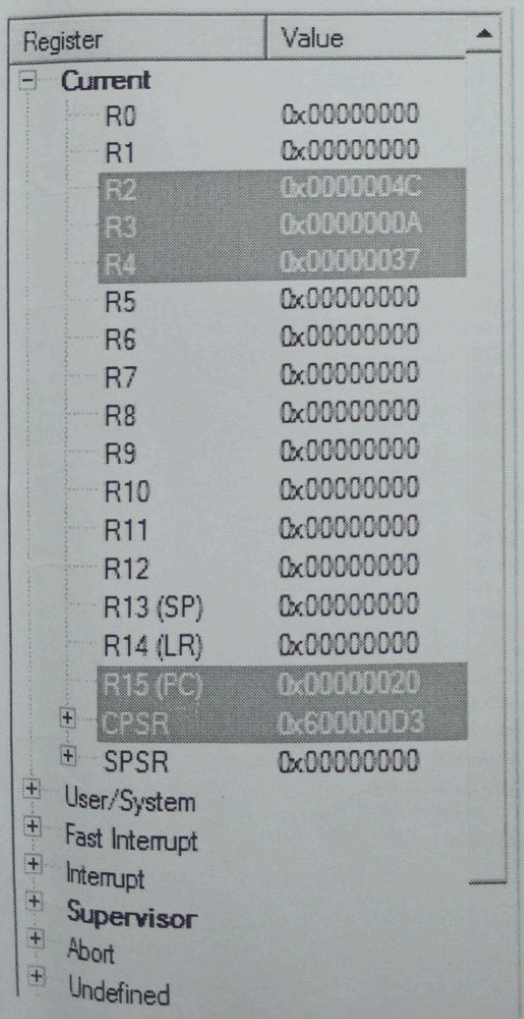


2. Write a program to find the sum of first 10 integer numbers.

```
        AREA ADD1TO10, CODE, READONLY
        ENTRY
        MOV R1,#10                  ;length of array
        LDR R2,=ARRAY               ;Load the starting address of the array
        MOV R4,#0                   ;Initial sum
NEXT    LDR R3,[R2],#4              ;Load first integer of the array in R3
        ADD R4,R4,R3                ;R4=sum of integers
        SUBS R1,R1,#1
        BNE NEXT                    ;repeat until R1=0
STOP    B STOP

ARRAY   DCD 1,2,3,4,5,6,7,8,9,10
        END
```

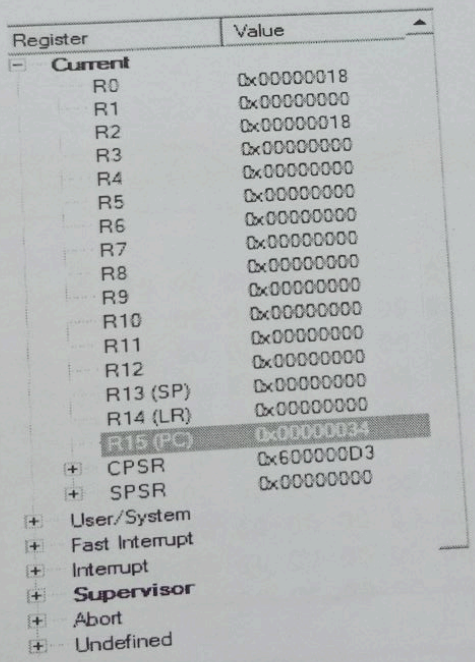
OUTPUT:



3. Write a program to find factorial of a number.

```
        AREA Factorial, CODE, READONLY
        ENTRY
        MOV  R0,#4              ; load the number in R0
        CMP R0,#0               ; check if the number is 0
        BEQ ANS                 ;if number is 0, go to label ANS
        CMP R0,#1               ; check if the number is 1
        BEQ ANS                 ;if number is 1, go to label ANS
        MOV  R1,R0              ; Copy the number in R1
UP      SUBS R1,R1,#1           ; decrement the value in R1 till 0
        BEQ  STOP               ; if yes store factorial value
        MUL  R2,R1,R0           ; if not fact= R0 x R1
        MOV  R0,R2              ; move fact value
        B       UP              ;repeat until R1 is 0
ANS     MOV R0,#1
STOP    B STOP                  ; Stop
```

OUTPUT :

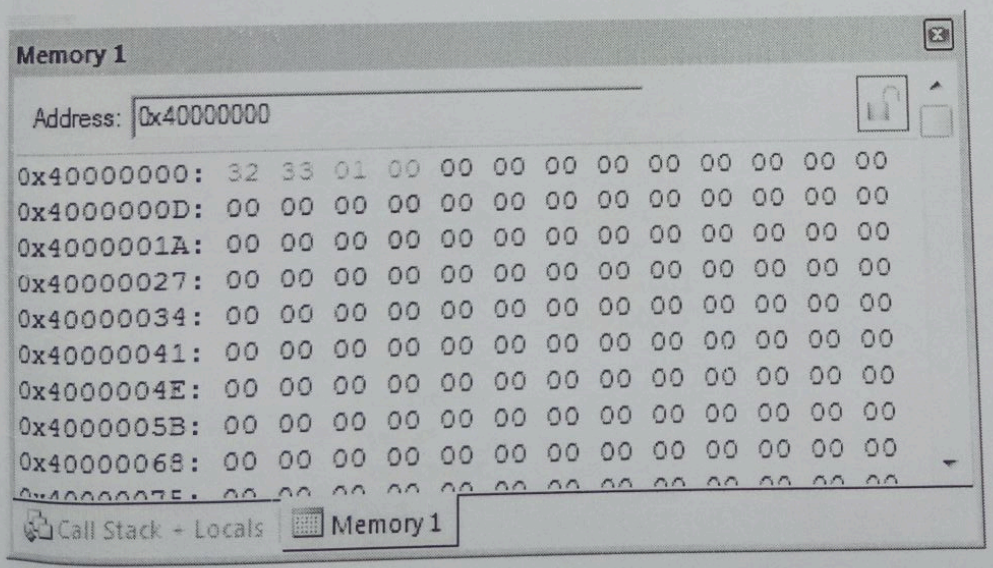


4. Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.

```
        AREA ADDITION, CODE, EADONLY
        ENTRY
        MOV R5,#6                   ;length of array
        MOV R0,#0                   ;initial sum
        LDR R1,=VALUE1              ;starting address of the array
LOOP    LDRH R2,[R1],#2             ;R2=first element of array
        ADD R0,R0,R2                ;add first element with initial sum
        SUBS R5,R5,#1
        BNE LOOP                    ;repeat addition until r5=0
        LDR R4,=RESULT
        STR R0,[R4]                 ;store the result in memory
STOP    B STOP

VALUE1  DCW 0X1111,0X2222,0X3333,0X4444,0X3333,0X5555
        AREA DATA2,DATA,READWRITE
RESULT  DCD 0X0
        END
```

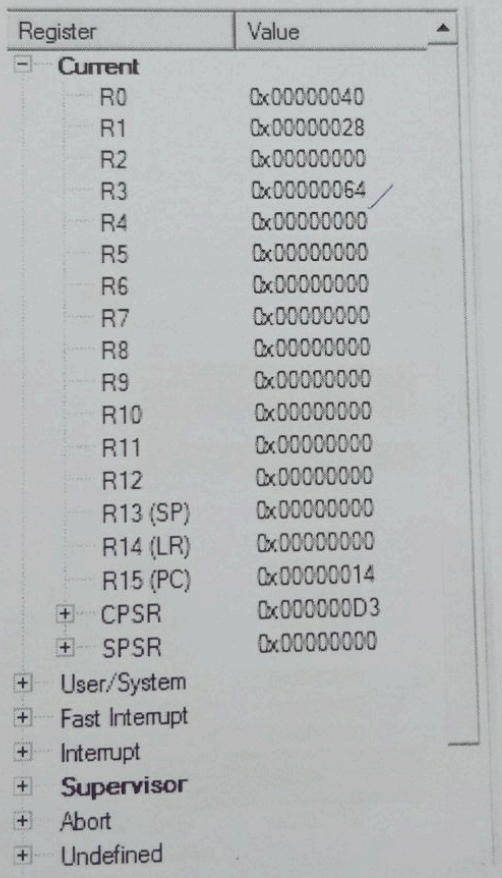
OUTPUT :



5. Write a program to find the square of a number (1 to 10) using look-up table.

```
        AREA square, CODE, READONLY
        ENTRY
        MOV  R1,#0X3            ; load the number to be squared
        LDR  R0,=LOOKUP         ; load the starting address of the lookup table
        MOV  R1,R1,LSL#0X2      ; offset of value to be squared
        ADD  R0,R0,R1           ; points to mem where square of the given no is sorted
        LDR  R3,[R0]            ; load the squared value from look-up table
STOP B       STOP
LOOKUP  DCD  0X0,0X1,0x4,0x9,0x10,0x19,0x24,0x31,0x40,0x51,0x64
                                        ; look-up table
        END
```

OUTPUT :



Register	Value
Current	
R0	0x00000040
R1	0x00000028
R2	0x00000000
R3	0x00000064
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	

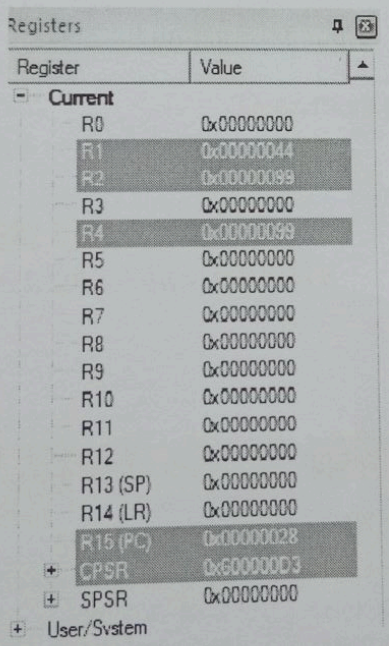
6. Write a program to find the largest/smallest number in an array of 32 numbers.

```
        AREA LARGE,CODE,READONLY
        ENTRY
        MOV R5,#5              ;R5 = length of array - 1
        LDR R1,=ARRAY          ;load starting addressing of array
        LDR R2,[R1],#4         ;load 1st element of array
LOOP    LDR R4,[R1],#4         ;load next element of array
        CMP R2,R4              ;compare 1st and 2nd element
        BHI NEXT
        MOV R2,R4              ;R2=largest value
NEXT    SUBS R5,R5,#1          ;decrement the counter after every comparison
        BNE LOOP               ; repeat until R5=0
STOP    B STOP

ARRAY DCD 0X23,0X45,0X65,0X76,0X12,0X99

        END
```

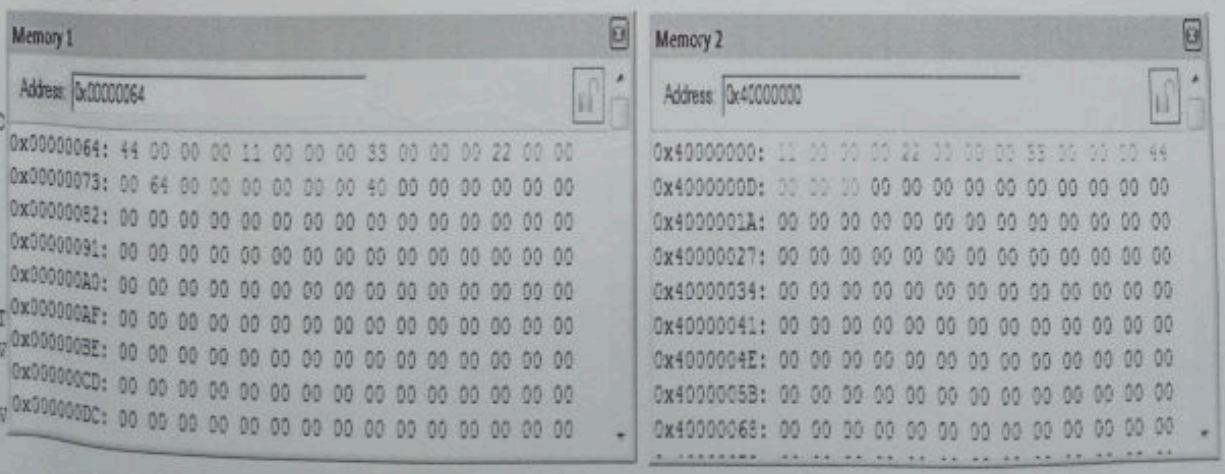
OUTPUT:



7. Write a program to arrange a series of 32 bit numbers in ascending/descending order.

```
                AREA Ascending, CODE, READONLY
                ENTRY
                MOV R8,#4               ;Length of the array
                LDR R2,=SVALUE          ;Starting address of the source array
                LDR R3,=DVALUE          ;Starting address of the destination array
LOOP0           LDR R1,[R2],#4      ;Loop0 copies all the elements of source ary to dest ary
                STR R1,[R3],#4
                SUBS R8,R8,#1
                CMP R8,#0
                BNE LOOP0
                MOV R7,#3               ;R7=Number of pass
NXTPAS          MOV R5,R7               ;R5=Number of comparisons
                LDR R1,=DVALUE          ;Loads the starting address of dest array in R1
NXTCMP          LDR R2,[R1],#4
                LDR R3,[R1]
                CMP R2,R3               ;Compares first and second element of the array
                BLT NOSWP               ;If first element is smaller, no swapping
                STR R2,[R1],#-4         ;Swaps the elements of the array
                STR R3,[R1]
                ADD R1,R1,#4
NOSWP           SUBS R5,R5,#1           ;Decrement comparison counter by 1 till 0
                BNE NXTCMP
                SUBS R7,R7,#1           ;Decrement pass counter by 1 till 0
                BNE NXTPAS
STOP            B STOP
SVALUE          DCD 0X44,0X11,0X33,0X22
                AREA DATA1,DATA,READWRITE
DVALUE          DCD 0X00
                END
```

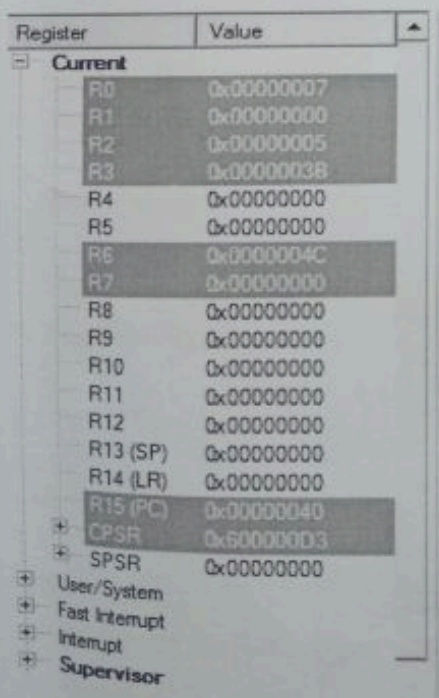
OUTPUT :



8. Write a program to count the number of ones and zeros in two consecutive memory locations.

```
            AREA ONEZERO, CODE, READONLY
            ENTRY
            MOV R2,#0           ;Counter for ones
            MOV R3,#0           ;Counter for zeros
            MOV R7,#2           ;Counter of 2 numbers
            LDR  R6,=LOOKUP     ;Load starting address of numbers
LOOP        MOV R1,#32          ;Number of bits in each number
            LDR R0,[R6]         ;Load 1st number to r0
NEXTBIT     MOVS R0,R0,ROR #1   ;Check the bit is one or zero
            BHI ONES            ; IF CF=1 increment r2 else increment r3
ZEROS       ADD R3,R3,#1        ;R3 stores count of 0s
            B REPEAT
ONES        ADD R2,R2,#1        ;R2 stores count of 1s
REPEAT      SUBS R1,R1,#1       ; Decrement the bit counter by 1 till 0
            BNE NEXTBIT         ; Repeat until r1=0
            ADD R6,R6,#4        ; Load r6=address of next number
            SUBS R7,R7,#1       ; Decrement the number counter by 1 till 0
            BNE LOOP
STOP        B  STOP
            LOOKUP    DCD  0X5,0X7          ; Memory address of lookup table
            END
```

OUTPUT :



PART B

1. Display "Hello World" message using Internal UART.

```
#include <lpc214x.h>
void uart_interrupt(void)_irq ;
unsigned char temp , temp1 = 0x00 ;
unsigned char rx_flag = 0 , tx_flag = 0 ;

int main(void)
{
PINSEL0=0X0000005;                    //select TXD0 and RXD0 lines

U0LCR = 0X00000083;                   //enable baud rate divisor loading and
U0DLM = 0X00;                         //select the data format
U0DLL = 0x13;                         //select baud rate 9600 bps
U0LCR = 0X00000003;
U0IER = 0X03;                        //select Transmit and Recieve interrupt

   VICVectAddr0 = (unsigned long)uart_interrupt;          //UART 0 INTERRUPT
   VICVectCntl0 = 0x20|6;          // Assign the VIC channel uart-0 to interrupt priority 0
VICIntEnable = 0x00000040;         // Enable the uart-0 interrupt

   rx_flag = 0x00;
   tx_flag = 0x00;

   while(1)
   {
   while(rx_flag == 0x00);             //wait for receive flag to set
rx_flag = 0x00;                        //clear the flag

U0THR = temp1 ;

      while(tx_flag == 0x00);         //wait for transmit flag to set
 tx_flag = 0x00;                      //clear the flag
    }
 }

 void uart_interrupt(void)_irq
 {
 temp = U0IIR;
    temp = temp & 0x06;              //check bits, data sending or receiving

     if(temp  == 0x02)               //check data is sending
  {
```

```
  tx_flag = 0xff;                // flag that indicate data is sending via UART0
   VICVectAddr=0;
}

  else if(temp  == 0x04)         // check any data available to receive
  {
// U0THR = U0RBR;
emp1 = U0RBR ;                   // copy data into variable
   rx_flag = 0xff;               // set flag to indicate that data is received
   VICVectAddr=0;
}
}
```

2. Interface and Control a DC Motor.

```
#include<lpc214x.h>
void clock_wise(void);
void anti_clock_wise(void);
unsigned int j=0;

int main()
{
 PINSEL2 = 0XFFFFFFF0;
       //IO1CLR = 0X0000ff00;
       IO1DIR= 0X00030000;        //p1.16 and p1.17 are selected as outputs.
       IO1SET= 0X00010000;        //P1.16 should always high.

       while(1)
       {

       clock_wise();
       for(j=0;j<500000;j++);              //delay

       anti_clock_wise();
       for(j=0;j<500000;j++);              //delay

       }                                   //End of while(1)
 }                                         //End of Main

 void clock_wise(void)
 {
       IO1CLR = 0x00030000;        //stop motor and also turn off relay
       for(j=0;j<500000;j++);      //small delay to allow motor to turn off
       IO1SET = 0X00030000; //Selecting the P1.17 line for clockwise and turn on motor
  }

  void anti_clock_wise(void)
```

```
{
IO1CLR = 0X00030000;                    //stop motor and also turn off relay
 for(j=0;j<1000000;j++);                //small delay to allow motor to turn off
 IO1SET = 0X00010000;                   //not selecting the P1.17 line for Anti clockwise

}
```

3. Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.

```
#include <LPC21xx.h>
void clock_wise(void) ;
void anti_clock_wise(void) ;
unsigned int var1 ;
unsigned long int i = 0 , j = 0 , k = 0 ;

int main(void)
{
        PINSEL2 = 0x00000000;           //P1.20 to P1.23 GPIO
        IO1DIR |= 0x00F00000 ;          //P1.20 to P1.23 made as output

        while(1)
        {

        for( j = 0 ; j < 50 ; j++ )     // 50 times in Clock wise Rotation
        clock_wise() ;                   // rotate one round clockwise
        IO1CLR =0x00F00000 ;            //clearing all 4 bits
        while(1);
        for( k = 0 ; k < 65000 ; k++ ) ;  // Delay to show anti_clock Rotation
        for( j=0 ; j < 50 ; j++ )       // 50 times in Anti Clock wise Rotation
        anti_clock_wise() ;             // rotate one round anticlockwise
        for( k = 0 ; k < 65000 ; k++ ) ;  // Delay to show ANTI_clock Rotation

        }

}                                       // End of main
 void clock_wise(void)
{
        var1 = 0x00080000;              //For Clockwise
    for( i = 0 ; i <= 3 ; i++ )         // for A B C D Stepping
        {
                var1 <<= 1 ;

                IO1CLR =0x00F00000 ;        //clearing all 4 bits
```

```
                IO1SET = var1 ;                 // setting perticular bit

                for( k = 0 ; k < 3000 ; k++ ); //for step speed variation
        }
}

void anti_clock_wise(void)
{
        var1 = 0x01000000 ;                     //For Anticlockwise

   for( i = 0 ; i <=3 ; i++ )                   // for A B C D Stepping
        {
                var1 >>=1;                      //rotating bits
                 IO1CLR =0x00F00000 ;           // clear all bits before setting
                IO1SET = var1 ;                 // setting perticular bit
     for( k = 0 ; k < 3000 ; k++ ) ;            //for step speed variation

        }
}
```

4. Determine Digital output for a given Analog input using Internal ADC of ARM controller.

```
#include <lpc214x.h>
#include <Stdio.h>
#define vol 3.3                 //Reference voltage
#define fullscale 0x3ff         //10 bit adc fullscale
unsigned int data_lcd=0,i=0,n=0;
unsigned int adc_value=0,temp_adc=0,temp1,temp2,adc[8];
float temp,adc1[8];
unsigned char var[15],var1[15],fst_flag=0xff;
unsigned char *ptr,arr[]= "ADC O/P= ";
unsigned char *ptr1,dis[]="A I/P = ";
```

```
void lcd_init(void);
void wr_cn(void);
void clr_disp(void);
void delay(unsigned int);
void lcd_com(void);
void wr_dn(void);
void lcd_data(void);

int main()
{
        PINSEL1 = 0X04000000;   //AD0.2 pin is selected
  IO0DIR = 0x000000FC;    //configure o/p lines for lcd

       delay(3200);
       lcd_init();                              //LCD initialization
 delay(3200);
       clr_disp();                              //clear display
       delay(3200);                       //delay

       ptr = dis;
       temp1 = 0x80;                      //Display starting address of 1st line on LCD
       lcd_com();
       delay(800);

  while(*ptr!='\0')
  {
            temp1 = *ptr;
       lcd_data();
             ptr ++;
       }

       ptr1 = arr;
       temp1 = 0xC0;                            //Display starting address of 2nd line on
LCD
       lcd_com();
       delay(800);

       while(*ptr1!='\0')
   {
       temp1 = *ptr1;
       lcd_data();
             ptr1 ++;
       }
```

```
  while(1)    //infinite loop
  {
   temp = 0;
   adc_value = 0;
                AD0CR = 0x01200004;                         ////CONTROL register for ADC-
AD0.4

                while(((temp_adc = AD0GDR) &0x80000000) == 0x00000000); //to check
the interrupt bit

                adc_value = AD0GDR;                 //reading the ADC value
                adc_value >>=6;
                adc_value &= 0x000003ff;
                temp = ((float)adc_value * (float)vol)/(float)fullscale;

                if(fst_flag)
                {
  fst_flag = 0x00;
                        for(i=0;i<8;i++)
                        {
                                adc[i] = adc_value;
                                adc1[i] = temp;
                        }
                }

                else
                {
                        n=7;
                        for(i=n;i>0;i--)
                        {
                        adc[i] = adc[i-1];
                        adc1[n] = adc1[n-1];
                        n = n-1;
                        }
                 adc[0] = adc_value;
                 adc1[0] = temp;
                }
                temp=0;
                adc_value=0;
                for(i=0;i<8;i++)
                {
                        temp += adc1[i];
                        adc_value += adc[i];
```

```
                }
                 temp = (temp/8);
                 adc_value = (adc_value/8);

                sprintf(var1,"%4.2fV",temp);
                sprintf(var,"%3x",adc_value);

                temp1 = 0x89;
                 lcd_com();
                 delay(1200);
                 ptr1 = var1;

                while(*ptr1!='\0')
     {
                        temp1=*ptr1;
                         lcd_data();
     ptr1++;
                }

     temp1 = 0xc9;
     lcd_com();
     delay(1200);

     ptr1 = var;
     while(*ptr1!='\0')
     {
       temp1=*ptr1;
                        lcd_data();
     ptr1++;
              }
       }        // end of while(1)
} //end of main()

//**** LCD initialization ****//
void lcd_init()
{
        temp2=0x30;
        wr_cn();
        delay(800);

        temp2=0x30;
        wr_cn();
        delay(800);
```

```
        temp2=0x30;
        wr_cn();
        delay(800);

        temp2=0x20;
        wr_cn();
        delay(800);

        temp1 = 0x28;
        lcd_com();
        delay(800);

        temp1 = 0x0c;
        lcd_com();
        delay(800);

        temp1 = 0x06;
        lcd_com();
        delay(800);

        temp1 = 0x80;
        lcd_com();
        delay(800);
}

void lcd_com(void)
{
        temp2= temp1 & 0xf0;
        wr_cn();
        temp2 = temp1 & 0x0f;
        temp2 = temp2 << 4;
        wr_cn();
   delay(500);
}

// command nibble o/p routine
void wr_cn(void)            // write command reg
{
        IO0CLR  = 0x000000FC;              // clear the port lines.
        IO0SET        = temp2;                    // Assign the value to the PORT lines
        IO0CLR  = 0x00000004;              // clear bit RS = 0
        IO0SET        = 0x00000008;        // ENABLE=1
        delay(10);
        IO0CLR = 0x00000008;
```

```
}

// data nibble o/p routine
void wr_dn(void)
{
        IO0CLR  = 0x000000FC;                   // clear the port lines.
        IO0SET = temp2;                                 // Assign the value to the PORT lines
        IO0SET = 0x00000004; // set bit RS = 1
        IO0SET = 0x00000008; // ENABLE=1
        delay(10);
        IO0CLR = 0x00000008;
}

// data o/p routine which also outputs high nibble first
// and lower nibble next
void lcd_data(void)
{
        temp2 = temp1 & 0xf0;
   wr_dn();
   temp2= temp1 & 0x0f;
   temp2= temp2 << 4;
   wr_dn();
   delay(100);
}

void delay(unsigned int r1)
{
        unsigned int r;
        for(r=0;r<r1;r++);
}
void clr_disp(void)
{
        temp1 = 0x01;
        lcd_com();
        delay(500);
}
```

5. Interface a DAC and generate Triangular and Square waveforms.

// program to generate Triangular wave with DAC interface

```
#include <LPC21xx.h>
int main ()
{
        unsigned long int temp=0x00000000;
        unsigned int i=0;
        IO0DIR=0x00FF0000;
     while(1)
   {
        // output 0 to FE
      for(i=0;i!=0xFF;i++)
      {
        temp=i;
        temp = temp << 16;
        IO0PIN=temp;
      }

                // output FF to 1
      for(i=0xFF; i!=0;i--)
      {
        temp=i;
        temp = temp << 16;
        IO0PIN=temp;
      }
        }//End of while(1)
}//End of main()
```

// program to generate square wave with DAC interface

```
#include <lpc21xx.h>
void delay(void);
int main ()
{
        PINSEL0 = 0x00000000 ;                    // Configure P0.0
to P0.15 as GPIO
        PINSEL1 = 0x00000000 ;                    // Configure P0.16
to P0.31 as GPIO
        IO0DIR = 0x00FF0000 ;

        while(1)
   {
         IO0PIN = 0x00000000;
```

```
        delay();
        IO0PIN = 0x00FF0000;
        delay();
    }
}

void delay(void)
{
        unsigned int i=0;
       for(i=0;i<=3000;i++);
}
```

6. Interface a 4x4 keyboard and display the key code on an LCD

```
#include<lpc21xx.h>
#include<stdio.h>

void scan(void);
void get_key(void);
void display(void);
void delay(unsigned int);
void init_port(void);

void lcd_init(void);
void clr_disp(void);
void lcd_com(void);              // LCD routines
void lcd_data(void);
void wr_cn(void);
void wr_dn(void);

unsigned long int scan_code[16]= { 0x0000EE00,0x0000ED00,0x0000EB00,0x0000E700 ,
                                   0x0000DE00,0x0000DD00,0x0000DB00,0x0000D700 ,
                                   0x0000BE00,0x0000BD00,0x0000BB00,0x0000B700 ,
                                   0x00007E00,0x00007D00,0x00007B00,0x00007700 };



unsigned char ASCII_CODE[16]= {'0','4','8','C',
                               '1','5','9','D',
                               '2','6','A','E',
                               '3','7','B','F'};
unsigned char row,col;
```

```
unsigned char temp,flag,i,result,temp1;
unsigned int r,r1;
unsigned long int var,var1,var2,res1,temp2,temp3,temp4;
unsigned char *ptr;
unsigned char disp0[] = "KEYPAD TESTING";
unsigned char disp1[] = "KEY = ";
int main()
{
      PINSEL0 = 0X00000000;                  // configure P0.0 TO P0.15 as GPIO
      init_port();        //port intialisation
      delay(3200);                           //delay
      lcd_init();         //lcd intialisation
      delay(3200);                           //delay
clr_disp();                                  //clear display
      delay(500);          //delay
      clr_disp();

ptr = disp0;
      temp1 = 0x80;                  // Display starting address of 1st line on LCD
      lcd_com();

      while(*ptr!='\0')
 {
       temp1 = *ptr;
      lcd_data();
              ptr ++;
 }

 ptr = disp1;
       temp1 = 0xC0;                         // Display starting address of 2nd line on LCD
       lcd_com();

     while(*ptr!='\0')
{
      temp1 = *ptr;
  lcd_data();
              ptr ++;
}


    while(1)
    {
     get_key();
  display();
```

```
    }
} //end of main()

void get_key(void)              //get the key from the keyboard
{
      unsigned int k;
      flag = 0x00;
IO0PIN=0x0000F000;

    while(1)
    {
    for(row=0X00;row<0X04;row++)    //Writing one for col's
 {
    if( row == 0X00)
    {
          temp3=0x00000E00;
               }
    else if(row == 0X01)
    {
          temp3=0x00000D00;
               }
               else if(row == 0X02)
               {
          temp3=0x00000B00;
               }
   else if(row == 0X03)
               {
          temp3=0x00000700;
               }

                  var1 = temp3;
   IO0PIN = var1;               // each time var1 value is put to port1
   IO0CLR =~var1;               // Once again Confirming (clearing all other bits)

   scan();
   delay(100);                  //delay

   if(flag == 0xff)
         break;

   } // end of for loop

         if(flag == 0xff)
```

```
                    break;
        } // end of while

    for(k=0;k<16;k++)
    {
        if(scan_code[k] == res1)      //equate the scan_code with res1
        {
                    result =  ASCII_CODE[k];   //same position value of ascii code
                    break;                     //is assigned to result
        }
    }
}// end of get_key();

void scan(void)
{
  unsigned long int t;
  temp2 = IO0PIN;                        // status of port1
  temp2 = temp2 & 0x0000F000;              // Verifying column key
  if(temp2 != 0x0000F000)                 // Check for Key Press or Not
  {
      delay(3000);                        //delay(100)//give debounce delay check again
      temp2 = IO0PIN; //IO0
      temp2 = temp2 & 0x0000F000;           //changed condition is same

    if(temp2 != 0x0000F000)               // store the value in res1
    {
      flag = 0xff;
      res1 = temp2;
      t = (temp3 & 0x00000F00);          //Verfying Row Write
      res1 = res1 | t;                  //final scan value is stored in res1
    }
    else
    {
      flag = 0x00;
    }
  }
} // end of scan()

void display(void)
{

  temp1 = 0xC6;                          //display address for key value
        lcd_com();
  temp1 = result;
```

```
  lcd_data();
}
void lcd_init (void)
{
      temp = 0x30;
      wr_cn();
      delay(3200);

      temp = 0x30;
      wr_cn();
      delay(3200);

      temp = 0x30;
      wr_cn();
      delay(3200);

      temp = 0x20;
      wr_cn();
      delay(3200);

      temp = 0x28; // load command for lcd function setting with lcd in 4 bit mode,
      lcd_com();           // 2 line and 5x7 matrix display
      delay(3200);

      temp1 = 0x0C;        // load a command for display on, cursor on and blinking off

      lcd_com();
      delay(800);

      temp1 = 0x06;        // command for cursor increment after data dump
      lcd_com();
      delay(800);

      temp1 = 0x80;
      lcd_com();
      delay(800);
}

void lcd_data(void)
{
  temp = temp1 & 0xf0;
  wr_dn();
  temp= temp1 & 0x0f;
```

```
    temp= temp << 4;
    wr_dn();
    delay(100);
  }

  void wr_dn(void)                          ////write data reg
  {
        IO0CLR  = 0x000000FC;       // clear the port lines.
        IO0SET = temp;                    // Assign the value to the PORT lines
        IO0SET = 0x00000004;        // set bit RS = 1
        IO0SET = 0x00000008;        // Enable=1
        delay(10);
        IO0CLR = 0x00000008;
  }

  void lcd_com(void)
  {
    temp = temp1 & 0xf0;
    wr_cn();
    temp = temp1 & 0x0f;
    temp = temp << 4;
    wr_cn();
    delay(500);
  }

  void wr_cn(void)            //write command reg
  {
        IO0CLR  = 0x000000FC;                 // clear the port lines.
        IO0SET        = temp;                           // Assign the value to the
 PORT lines
        IO0CLR  = 0x00000004;                 // clear bit RS = 0
        IO0SET        = 0x00000008;         // Enable=1
        delay(10);
        IO0CLR = 0x00000008;
  }

  void clr_disp(void)
  {
    temp1 = 0x01;                     // command to clear lcd display
    lcd_com();
    delay(500);
  }

  void delay(unsigned int r1)
```

```
{
        for(r=0;r<r1;r++);
}

void init_port()
{

  IO0DIR = 0x00000FFC;      //Configured LCD Lines and Rows as O/P(P0.8-P0.11) and
Columns as I/P(P0.12-P0.15)

        IO0SET = 0x0000FF00; //Set the Rows high.
}
```

7. Interface a 4x4 keyboard and display the key code on an LCD

```
#include<lpc21xx.h>
#include<stdio.h>

void scan(void);
void get_key(void);
void display(void);
void delay(unsigned int);
void init_port(void);

void lcd_init(void);
void clr_disp(void);
void lcd_com(void);              // LCD routines
void lcd_data(void);
void wr_cn(void);
void wr_dn(void);

unsigned long int scan_code[16]= { 0x0000EE00,0x0000ED00,0x0000EB00,0x0000E700 ,
                                   0x0000DE00,0x0000DD00,0x0000DB00,0x0000D700 ,
                                   0x0000BE00,0x0000BD00,0x0000BB00,0x0000B700 ,
                                   0x00007E00,0x00007D00,0x00007B00,0x00007700 };



unsigned char ASCII_CODE[16]= {'0','4','8','C',
                               '1','5','9','D',
                               '2','6','A','E',
                               '3','7','B','F'};
unsigned char row,col;
unsigned char temp,flag,i,result,temp1;
unsigned int r,r1;
unsigned long int var,var1,var2,res1,temp2,temp3,temp4;
unsigned char *ptr;
unsigned char disp0[] = "KEYPAD TESTING";
unsigned char disp1[] = "KEY = ";
int main()
{
        PINSEL0 = 0X00000000;                   // configure P0.0 TO P0.15 as GPIO
        init_port();          //port intialisation
        delay(3200);                            //delay
        lcd_init();           //lcd intialisation
        delay(3200);                            //delay
                                                //clear display
  clr_disp();
```

```
        delay(500);          //delay
        clr_disp();

ptr = disp0;
        temp1 = 0x80;                    // Display starting address of 1st line on LCD
        lcd_com();

        while(*ptr!='\0')
{
         temp1 = *ptr;
        lcd_data();
                ptr ++;
}

 ptr = disp1;
          temp1 = 0xC0;                  // Display starting address of 2nd line on LCD
          lcd_com();

        while(*ptr!='\0')
{
         temp1 = *ptr;
    lcd_data();
                ptr ++;
}


        while(1)
        {
         get_key();
     display();
        }

} //end of main()

void get_key(void)              //get the key from the keyboard
{
         unsigned int k;
         flag = 0x00;
   IO0PIN=0x0000F000;

        while(1)
        {
        for(row=0X00;row<0X04;row++)    //Writing one for col's
     {
        if( row == 0X00)
```

```
{
        temp3=0x00000E00;
            }
else if(row == 0X01)
{
        temp3=0x00000D00;
            }
            else if(row == 0X02)
            {
        temp3=0x00000B00;
            }
else if(row == 0X03)
            {
        temp3=0x00000700;
            }

                var1 = temp3;
IO0PIN = var1;              // each time var1 value is put to port1
IO0CLR =~var1;              // Once again Confirming (clearing all other bits)

scan();
delay(100);                 //delay

if(flag == 0xff)
        break;

} // end of for loop

        if(flag == 0xff)
            break;
} // end of while

for(k=0;k<16;k++)
{
    if(scan_code[k] == res1)    //equate the scan_code with res1
    {
                result =  ASCII_CODE[k];   //same position value of ascii code
                break;                     //is assigned to result

    }
}
}// end of get_key();

void scan(void)
{
  unsigned long int t;
```

```
    temp2 = IO0PIN;
    temp2 = temp2 & 0x0000F000;          // status of port1
    if(temp2 != 0x0000F000)                    // Verifying column key
    {                                          // Check for Key Press or Not
        delay(3000);
        temp2 = IO0PIN; //IO0                  //delay(100)//give debounce delay check again
        temp2 = temp2 & 0x0000F000;                //changed condition is same

     if(temp2 != 0x0000F000)                   // store the value in res1
     {
       flag = 0xff;
       res1 = temp2;
       t = (temp3 & 0x00000F00);               //Verfying Row Write
       res1 = res1 | t;                  //final scan value is stored in res1
     }
     else
     {
       flag = 0x00;
     }
   }
 } // end of scan()

 void display(void)
 {

   temp1 = 0xC6;                           //display address for key value
         lcd_com();
   temp1 = result;
   lcd_data();
 }

 void lcd_init (void)
 {
        temp = 0x30;
        wr_cn();
        delay(3200);

        temp = 0x30;
        wr_cn();
        delay(3200);

        temp = 0x30;
        wr_cn();
        delay(3200);
```

```
        temp = 0x20;
        wr_cn();
        delay(3200);

        temp = 0x28; // load command for lcd function setting with lcd in 4 bit mode,
        lcd_com();              // 2 line and 5x7 matrix display
        delay(3200);

        temp1 = 0x0C;           // load a command for display on, cursor on and blinking off

        lcd_com();
        delay(800);

        temp1 = 0x06;           // command for cursor increment after data dump
        lcd_com();
        delay(800);

        temp1 = 0x80;
        lcd_com();
        delay(800);
}

void lcd_data(void)
{
  temp = temp1 & 0xf0;
  wr_dn();
  temp= temp1 & 0x0f;
  temp= temp << 4;
  wr_dn();
  delay(100);
}

void wr_dn(void)                    ////write data reg
{
      IO0CLR  = 0x000000FC;         // clear the port lines.
                                    // Assign the value to the PORT lines
      IO0SET = temp;
      IO0SET = 0x00000004;          // set bit RS = 1
      IO0SET = 0x00000008;          // Enable=1
      delay(10);
      IO0CLR = 0x00000008;
}

void lcd_com(void)
{
  temp = temp1 & 0xf0;
```

```
    wr_cn();
    temp = temp1 & 0x0f;
    temp = temp << 4;
    wr_cn();
    delay(500);
  }

  void wr_cn(void)          //write command reg
  {
        IO0CLR  = 0x000000FC;          // clear the port lines.
        IO0SET        = temp;                  // Assign the value to the
  PORT lines
        IO0CLR  = 0x00000004;          // clear bit RS = 0
        IO0SET        = 0x00000008;    // Enable=1
        delay(10);
        IO0CLR = 0x00000008;
  }

  void clr_disp(void)
  {
    temp1 = 0x01;           // command to clear lcd display
    lcd_com();
    delay(500);
  }

  void delay(unsigned int r1)

  {
        for(r=0;r<r1;r++);
  }

  void init_port()
  {

    IO0DIR = 0x00000FFC;      //Configured LCD Lines and Rows as O/P(P0.8-P0.11) and
Columns as I/P(P0.12-P0.15)

        IO0SET = 0x0000FF00; //Set the Rows high.
}
                        flag = 0x00;
                    }
                }
        }
}
```

```
void EINT0_Init(void)
{
        IO1DIR |= 0X02000000;                                   // P1.25 for
LED indication
//      PINSEL1 &= ~0x00000003;
        PINSEL1         = 0X00000001;                           // Setup P0.16 to alternate
function EINT0
        EXTMODE     = 0x01;                                     // edge i.e falling
        / Assign the EINT0 ISR function VICVectCntl0 = 0x20 | 14;          // Assign the
        VIC
channel EINT0 to interrupt priority 0
        VICIntEnable |= 0x00004000;                             // Enable the
EINT0 interrupt
}

void Extint0_Isr(void)__irq                                     // whenever there is a low level on
EINT0
{
        EXTINT |= 0x01;
        int_flg = 0xFF;                                         // Clear interrupt
        VICVectAddr = 0;                                        //
Acknowledge Interrupt
}
```

8. Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between.

```
#include <LPC21XX.h>
unsigned int delay,j;
unsigned int Switchcount=0;
unsigned int Disp[16]={0x003F0000, 0x00060000, 0x005B0000, 0x004F0000,
0x00660000,0x006D0000,
                       0x007D0000, 0x00070000, 0x007F0000,
0x006F0000, 0x00770000,0x007C0000,
                       0x00390000, 0x005E0000, 0x00790000, 0x00710000

};

int main (void)
{
        PINSEL0 = 0x00000000;
        PINSEL1 = 0x00000000;
        IO0DIR = 0x00FF0000;
```

1 | 8 c . h
2

1000 1111
8 F

```
        //IO1DIR = 0x00000000;

        while(1)
        {
                IO0CLR = 0x00FF0000;                    // clear the data lines to 7-
segment displays
                IO0SET = Disp[Switchcount];             // get the 7-segment display value
from the array

                for(j=0;j<10;j++)
                        for(delay=0;delay<30000;delay++);   // 1s delay

                                Switchcount++;
                                if(Switchcount == 0x10)     // 0 to F has been displayed  go
back to 0
                        {
                                Switchcount = 0;
                                IO0CLR = 0x00FF0000;
                        }
        }
}
```