VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590 010

ARM PROCESSOR

(17EIL77)

LABORATORY MANUAL

VII Semester - B.E.

Prepared By
Manjunath K.G.

Department of Electronics and Instrumentation Engineering

Bapuji Institute of Engineering and Technology,
Davangere - 577 004, Karnataka.

PART-A:

Conduct the following experiments by writing Assembly Language Program (ALP) using ARM Cortex M3 Registers using an evaluation board/simulator and the required software tool.

1. Write an ALP to multiply two 16 bit binary numbers.

2. Write an ALP to find the sum of first 10 integer numbers.

3. Write an ALP to find factorial of a number.

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

5. Write an ALP to add two 64 bit numbers.

6. Write an ALP to find the square of a number(1 to 10) using look-up table.

7. Write an ALP to find the largest/smallest number in an array of 32 numbers .

8. Write an ALP to arrange a series of 32 bit numbers in ascending/descending order.

9. Write an ALP to count the number of ones and zeros in two consecutive memory locations.
10. Write an ALP to Scan a series of 32 bit numbers to find how many are negative.

PART-B:

Conduct the following experiments on an ARM CORTEX M3 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

1. Display “Hello World” message using Internal UART.

2. Interface and Control a DC Motor.

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

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

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

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

7. Using the Internal PWM module of ARM controller generate PWM and vary its duty cycle.

8. Demonstrate the use of an external interrupt to toggle an LED On/Off.

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

10. Interface a simple Switch and display its status through Relay, Buzzer and LED.

1. Write an ALP to multiply two 16 bit binary numbers.

```
;/*     VALUE1:     1900H (6400)         (IN R1)        */
;/*     VALUE2:     0C80H (3200)         (IN R2)        */
;/*     RESULT:     1388000H(20480000)(IN R3)           */

AREA multiply, CODE, READONLY

ENTRY                               ;Mark first instruction to execute

START

        MOV r1,#6400                ; STORE FIRST NUMBER IN R0
        MOV r2,#3200                ; STORE SECOND NUMBER IN R1
        MUL r3,r1,r2                ; MULTIPLICATION

        NOP
        NOP

        END                         ;Mark end of file
```

2. Write an ALP to find factorial of a number.

```
AREA  FACTORIAL , CODE, READONLY

ENTRY                                       ;Mark first instruction to execute

START

                MOV r0, #7                  ; STORE FACTORIAL NUMBER IN R0
                MOV r1,r0                   ; MOVE THE SAME NUMBER IN R1

FACT    SUBS r1, r1, #1                     ; SUBTRACTION
                CMP r1, #1                  ; COMPARISON
                BEQ STOP
                MUL r3,r0,r1;                ; MULTIPLICATION
                MOV   r0,r3                  ; Result
                BNE FACT                    ; BRANCH TO THE LOOP IF NOT EQUAL
STOP

                NOP
                NOP
                NOP

        END                                 ;Mark end of file
```

3. Write an ALP to add an array of 16 bit numbers and store the 32 bit result in internal RAM

```
;/*ARRAY OF 6 NUMBERS 0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC*/
;/* THE SUM IS 29997H THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 &
ALSO IN R0   */

        AREA ADDITION, CODE, READONLY

ENTRY                                        ;Mark first instruction to execute

START
        MOV R5,#6                           ; INTIALISE COUNTER TO 6(i.e. N=6)
        MOV R0,#0                           ; INTIALISE SUM TO ZERO
        LDR R1,=VALUE1                      ; LOADS THE ADDRESS OF FIRST VALUE
LOOP
        LDR R2,[R1],#2                       ; WORD ALIGN T0 ARRAY ELEMENT
        LDR R3,MASK                         ; MASK TO GET 16 BIT
        AND R2,R2,R3                        ; MASK MSB
        ADD R0,R0,R2                         ; ADD THE ELEMENTS
        SUBS R5,R5,#1                       ; DECREMENT COUNTER
        CMP R5,#0
        BNE LOOP                            ;  LOOK BACK TILL ARRAY ENDS
        LDR R4,=RESULT                      ; LOADS THE ADDRESS OF RESULT
        STR R0,[R4]                         ; STORES THE RESULT IN R1
        NOP
        NOP
        NOP
here B here

MASK DCD 0X0000FFFF                                 ;  MASK MSB

VALUE1 DCW      0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC   ; ARRAY OF
16 BIT NUMBERS(N=6)

        AREA DATA2,DATA,READWRITE ; TO STORE RESULT IN GIVEN ADDRESS

RESULT DCD 0X0

        END                                   ; Mark end of file
```

4. Write an ALP to add two 64 bit numbers.

```
;/*     VALUE1      0X1234E640 0X43210010 (R0,R1)*/
;/*      VALUE2     0X12348900 0X43212102 (R2,R3)*/
;/*     RESULT                0X24696F40 0X86422112 (R5,R4)*/

        AREA  ADDITION , CODE, READONLY

ENTRY                                       ;Mark first instruction to execute

START

        LDR R0,=0X1234E640          ;LOAD THE FIRST VALUE IN R0,R1
        LDR R1,=0X43210010
        LDR R2,=0X12348900          ;LOAD THE SECOND VALUE IN R2,R3
        LDR R3,=0X43212102
        ADDS R4,R1,R3               ;RESULT IS STORED IN R4,R5
        ADC R5,R0,R2

        NOP
        NOP
        NOP

        END                                 ;Mark end of file
```

5. Write an ALP to find the largest/smallest number in an array of 32 numbers.

```
;/*ARRAY OF 7 NUMBERS 0X44444444
,0X22222222,0X11111111,0X33333333,0XAAAAAAAA*/
;/*0X88888888 ,0X99999999                                          */
;/* RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN R2
*/

        AREA  LARGEST , CODE, READONLY

ENTRY                               ;Mark first instruction to execute

START
        MOV R5,#6                    ; INTIALISE COUNTER TO 6(i.e. N=7)
        LDR R1,=VALUE1               ; LOADS THE ADDRESS OF FIRST VALUE
        LDR R2,[R1],#4               ; WORD ALIGN T0 ARRAY ELEMENT
LOOP
        LDR R4,[R1],#4              ; WORD ALIGN T0 ARRAY ELEMENT
        CMP R2,R4                   ; COMPARE NUMBERS
        BHI LOOP1                   ; IF THE FIRST NUMBER IS > THEN GOTO LOOP1

        MOV R2,R4   ; IF THE FIRST NUMBER IS < THEN MOV CONTENT R4 TO R2
LOOP1
        SUBS R5,R5,#1               ; DECREMENT COUNTER
        CMP R5,#0                   ; COMPARE COUNTER TO 0
        BNE LOOP                    ; LOOP BACK TILL ARRAY ENDS

        LDR R4,=RESULT              ; LOADS THE ADDRESS OF RESULT
        STR R2,[R4]                 ; STORES THE RESULT IN R1

        NOP
        NOP
        NOP

 ARRAY OF 32 BIT NUMBERS(N=7)

VALUE1
                DCD 0X44444444              ;
                DCD   0X22222222            ;
                DCD   0X11111111            ;
                DCD   0X33333333            ;
                DCD   0XAAAAAAAA            ;
                DCD   0X88888888            ;
                DCD   0X99999999            ;



        AREA DATA2,DATA,READWRITE               ; TO STORE RESULT IN GIVEN
ADDRESS
RESULT DCD 0X0

        END                         ; Mark end of file
```

```
;/* PROGRAM TO FIND SMALLEST NUMBER IN AN ARRAY & STORE IN INTERNAL
RAM              */
;/*ARRAY OF 7 NUMBERS 0X44444444
,0X22222222,0X11111111,0X22222222,0XAAAAAAAA  */
;/*0X88888888 ,0X99999999                                                    */
;/* RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN R2     */

        AREA  SMALLEST , CODE, READONLY

ENTRY                                   ;Mark first instruction to execute

START
        MOV R5,#6                       ; INTIALISE COUNTER TO 6(i.e. N=7)
        LDR R1,=VALUE1                  ; LOADS THE ADDRESS OF FIRST VALUE
        LDR R2,[R1],#4                  ; WORD ALIGN T0 ARRAY ELEMENT
LOOP
        LDR R4,[R1],#4                  ; WORD ALIGN T0 ARRAY ELEMENT
        CMP R2,R4                       ; COMPARE NUMBERS
        BLS LOOP1                       ; IF THE FIRST NUMBER IS < THEN GOTO
LOOP1
        MOV R2,R4   ; IF THE FIRST NUMBER IS > THEN MOV CONTENT R4 TO R2
LOOP1
        SUBS R5,R5,#1                   ; DECREMENT COUNTER
        CMP R5,#0                       ; COMPARE COUNTER TO 0
        BNE LOOP                        ; LOOP BACK TILL ARRAY ENDS
        LDR R4,=RESULT                  ; LOADS THE ADDRESS OF RESULT
        STR R2,[R4]                     ; STORES THE RESULT IN R1
        NOP
        NOP

; ARRAY OF 32 BIT NUMBERS(N=7)
VALUE1
                DCD 0X44444444          ;
                DCD   0X22222222        ;
                DCD   0X11111111        ;
                DCD   0X22222222        ;
                DCD   0XAAAAAAAA        ;
                DCD   0X88888888        ;
                DCD   0X99999999        ;


        AREA DATA2,DATA,READWRITE       ; TO STORE RESULT IN GIVEN
ADDRESS
RESULT DCD 0X0

        END                             ; Mark end of file
```

6. Write an ALP to arrange a series of 32 bit numbers in ascending/descending order.

```
;/*ARRAY OF 4 NUMBERS 0X44444444 ,0X11111111,0X33333333,0X22222222 */
;/* SET A BREAKPOINT AT START1 LABLE & RUN THE PROGRAM               */
;/*CHECK THE UNSORTED NUMBERS AT LOCATION 0X40000000 NEXT      */
;/* SET A BREAKPOINT AT NOP INSTRUCTION,RUN THE PROGRAM & CHECK THE
RESULT                                                                */
;/* RESULT CAN BE VIEWED AT LOCATION 0X40000000                       */

          AREA  ASCENDING , CODE, READONLY

ENTRY                         ;Mark first instruction to execute

START

                    MOV R8,#4               ; INTIALISE COUNTER TO 4(i.e. N=4)
                    LDR R2,=CVALUE          ; ADDRESS OF CODE REGION
                    LDR R3,=DVALUE          ; ADDRESS OF DATA REGION

LOOP0
                    LDR R1,[R2],#4          ;  LOADING VALUES FROM CODE
REGION
                    STR R1,[R3],#4          ;  STORING VALUES TO DATA REGION

                    SUBS R8,R8,#1           ; DECREMENT COUNTER
                    CMP R8,#0               ; COMPARE COUNTER TO 0
                    BNE LOOP0               ; LOOP BACK TILL ARRAY ENDS

START1              MOV R5,#3               ; INTIALISE COUNTER TO 3(i.e. N=4)
                    MOV R7,#0    ; FLAG TO DENOTE EXCHANGE HAS OCCURED
                    LDR R1,=DVALUE   ; LOADS THE ADDRESS OF FIRST VALUE

LOOP          LDR R2,[R1],#4                ; WORD ALIGN T0 ARRAY ELEMENT
                    LDR R3,[R1]             ; LOAD SECOND NUMBER
                    CMP R2,R3      ; COMPARE NUMBERS
                    BLT LOOP2    ; IF THE FIRST NUMBER IS < THEN GOTO LOOP2
                    STR R2,[R1],#-4       ; INTERCHANGE NUMBER R2 & R3
                    STR R3,[R1]           ; INTERCHANGE NUMBER R2 & R3
                    MOV R7,#1    ; FLAG DENOTING EXCHANGE HAS TAKEN PLACE
                    ADD R1,#4                       ; RESTORE THE PTR

LOOP2

                    SUBS R5,R5,#1           ; DECREMENT COUNTER
                    CMP R5,#0               ; COMPARE COUNTER TO 0
```

```
                BNE LOOP                ; LOOP BACK TILL ARRAY ENDS
                CMP R7,#0               ; COMPARING FLAG
                BNE  START1 ; IF FLAG IS NOT ZERO THEN GO TO START1 LOOP

        NOP
        NOP
        NOP



; ARRAY OF 32 BIT NUMBERS(N=4) IN CODE REGION

CVALUE
            DCD 0X44444444              ;
            DCD   0X11111111            ;
            DCD   0X33333333            ;
            DCD   0X22222222            ;


        AREA DATA1,DATA,READWRITE       ;
; ARRAY OF 32 BIT NUMBERS IN DATA REGION
DVALUE
            DCD 0X00000000              ;

        END                     ; Mark end of file




;/* PROGRAM TO sort in Descending order */

;/*ARRAY OF 4 NUMBERS 0X44444444 ,0X11111111,0X33333333,0X22222222*/
;/* SET A BREAKPOINT AT START1 LABLE & RUN THE PROGRAM          */
;/*CHECK THE UNSORTED NUMBERS AT LOCATION 0X40000000 NEXT */
;/* SET A BREAKPOINT AT NOP INSTRUCTION,RUN THE PROGRAM & CHECK THE
RESULT              */
;/* RESULT CAN BE VIEWED AT LOCATION 0X40000000                 */


        AREA  DESCENDING , CODE, READONLY

ENTRY                           ;Mark first instruction to execute

START

                MOV R8,#4               ; INTIALISE COUNTER TO 4(i.e. N=4)
```

```
                        LDR R2,=CVALUE                  ; ADDRESS OF CODE REGION
                        LDR R3,=DVALUE                  ; ADDRESS OF DATA REGION

LOOP0
                        LDR R1,[R2],#4          ;  LOADING VALUES FROM CODE REGION
                        STR R1,[R3],#4          ;  STORING VALUES TO DATA REGION

                        SUBS R8,R8,#1                   ; DECREMENT COUNTER
                        CMP R8,#0                       ; COMPARE COUNTER TO 0
                        BNE LOOP0                       ; LOOP BACK TILL ARRAY ENDS

START1                  MOV R5,#3               ; INTIALISE COUNTER TO 3(i.e. N=4)
                        MOV R7,#0       ; FLAG TO DENOTE EXCHANGE HAS OCCURED
                        LDR R1,=DVALUE   ; LOADS THE ADDRESS OF FIRST VALUE

LOOP            LDR R2,[R1],#4                          ; WORD ALIGN T0 ARRAY ELEMENT
                        LDR R3,[R1]                     ; LOAD SECOND NUMBER
                        CMP R2,R3                       ; COMPARE NUMBERS
                        BGT LOOP2           ; IF THE FIRST NUMBER IS > THEN GOTO LOOP2
                        STR R2,[R1],#-4   ; INTERCHANGE NUMBER R2 & R3
                        STR R3,[R1]         ; INTERCHANGE NUMBER R2 & R3
                        MOV R7,#1       ; FLAG DENOTING EXCHANGE HAS TAKEN PLACE
                        ADD R1,#4               ; RESTORE THE PTR

LOOP2
                        SUBS R5,R5,#1                           ; DECREMENT COUNTER
                        CMP R5,#0                               ; COMPARE COUNTER TO 0
                        BNE LOOP                                ; LOOP BACK TILL ARRAY
ENDS
                        CMP R7,#0               ; COMPARING FLAG
                        BNE  START1 ; IF FLAG IS NOT ZERO THEN GO TO START1 LOOP
        NOP
        NOP
; ARRAY OF 32 BIT NUMBERS(N=4) IN CODE REGION

CVALUE
                DCD 0X44444444                          ;
                DCD   0X11111111                        ;
                DCD   0X33333333                        ;
                DCD   0X22222222                        ;

        AREA DATA1,DATA,READWRITE                       ;
; ARRAY OF 32 BIT NUMBERS IN DATA REGION
DVALUE
                DCD 0X00000000                          ;
        END                             ; Mark end of file
```

7. Write an ALP to count the number of ones and zeros in two consecutive memory locations.

```
;/*WE TOOK TWO NUMBERS i.e. 0X11111111,0XAA55AA55  (R0)    */
;/*CHECK THE RESULT IN R2 FOR ONES & R3 FOR ZEROS           */

AREA  ONEZERO , CODE, READONLY

ENTRY                                         ;Mark first instruction to execute

START

            MOV R2,#0                         ; COUNTER FOR ONES
            MOV R3,#0                         ; COUNTER FOR ZEROS
            MOV R7,#2                         ; COUNTER TO GET TWO WORDS
            LDR R6,=VALUE                     ; LOADS THE ADDRESS OF VALUE

LOOP        MOV R1,#32                        ; 32 BITS COUNTER
        LDR R0,[R6],#4                  ; GET THE 32 BIT VALUE

LOOP0       MOVS R0,R0,ROR #1           ; RIGHT SHIFT TO CHECK CARRY BIT (1's/0's)
            BHI ONES; IF CARRY BIT IS 1 GOTO ONES BRANCH OTHERWISE NEXT

ZEROS ADD R3,R3,#1; IF CARRY BIT IS 0 THEN INCREMENT THE COUNTER BY 1(R3)
            B LOOP1                     ; BRANCH TO LOOP1

ONES  ADD R2,R2,#1; IF CARRY BIT IS 1 THEN INCREMENT THE COUNTER BY 1(R2)

LOOP1       SUBS R1,R1,#1         ; COUNTER VALUE DECREMENTED BY 1
            BNE LOOP0             ; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT

            SUBS R7,R7,#1                     ; COUNTER VALUE DECREMENTED
BY 1
            CMP R7,#0                         ; COMPARE COUNTER R7 TO 0
            BNE LOOP                          ; IF NOT EQUAL GOTO TO LOOP

      NOP
      NOP
      NOP


VALUE DCD 0X11111111,0XAA55AA55;   TWO VALUES IN AN ARRAY

      END                   ; Mark end of file
```

8. Write an ALP to Scan a series of 32 bit numbers to find how many are negative.

```
;/*ARRAY OF 7 NUMBERS
0X12345678,0X8D489867,0X11111111,0X33333333,0XAAAAAAAA    */
;/*0XE605546C ,0X99999999                                 */
;/* RESULT CAN BE VIEWED  IN R2                           */

          AREA  NEGATIVE , CODE, READONLY

ENTRY                              ;Mark first instruction to execute

START
                MOV R5,#7                           ; INTIALISE COUNTER TO 7(i.e. N=7)
                MOV R2,#0                           ; COUNTER
                LDR R4,=VALUE               ; LOADS THE ADDRESS OF FIRST VALUE
LOOP
               LDR R1,[R4],#4           ; WORD ALIGN T0 ARRAY ELEMENT
               ANDS R1,R1,#1<<31            ; TO CHECK NEGATIVE NUMBER
               BHI FOUND  ; IF THE GIVEN NUMBER IS NEGATIVE GOTO FOUND
               B LOOP1; IF THE GIVEN NUMBER IS  NOT NEGATIVE GOTO LOOP1
FOUND
               ADD R2,R2,#1                         ; INCREMENT THE COUNTER
(NEGATIVE NUMBER)
               B LOOP1                                      ; GOTO LOOP1

LOOP1
               SUBS R5,R5,#1                        ; DECREMENT COUNTER
               CMP R5,#0                            ; COMPARE COUNTER TO 0
               BNE LOOP                             ; LOOP BACK TILL ARRAY ENDS

          NOP
          NOP

;ARRAY OF 32 BIT NUMBERS(N=7)

VALUE
               DCD   0X12345678   ;
               DCD   0X8D489867   ;
               DCD   0X11111111   ;
               DCD   0X33333333   ;
               DCD   0XE605546C  ;
               DCD0XAAAAAAAA;
               DCD   0X99999999   ;

          END                      ; Mark end of file
```

PART-B:

Conduct the following experiments on an ARM CORTEX M3 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

1. Display “Hello World” message using Internal UART.

```
#include<LPC17xx.h>
void delay(unsigned int r1);
void UART0_Init(void);
void UART0_IRQHandler(void);
unsigned long int r=0, i = 0;
unsigned char tx0_flag=0;
unsigned char *ptr, arr[] = "Hello world\r";
int main(void)
{
        SystemInit();
        SystemCoreClockUpdate();
        UART0_Init();
        while(1)
        {
                ptr = arr;
                while ( *ptr != '\0'){
                        LPC_UART0->THR = *ptr++;
                        while(tx0_flag == 0x00);
                        tx0_flag = 0x00;
                        for (i=0; i<200; i++);
                }
```

```
                    for (i=0; i<500; i++)

                    delay(625);                         //delay

          }

}

void UART0_Init(void)

{

          LPC_SC->PCONP |= 0x00000008;                      //UART0 peripheral enable

          LPC_PINCON->PINSEL0 = 0x00000050;

          LPC_UART0->LCR = 0x00000083; //enable divisor latch, parity disable, 1 stop bit, 8bit

          LPC_UART0->DLM = 0X00;

          LPC_UART0->DLL = 0x13;                            //select baud rate 9600 bps

          LPC_UART0->LCR = 0X00000003;

          LPC_UART0->FCR = 0x07;

          LPC_UART0->IER = 0X03;                  //select Transmit and receive interrupt


          NVIC_EnableIRQ(UART0_IRQn);                                 //Assigning channel

}
```

2. Interface and Control a DC Motor.

```
#include <LPC17xx.H>

void Clock_Wise(void);

void AClock_Wise(void);

unsigned long i;

int main(void)

{

        LPC_PINCON->PINSEL1 = 0x00000000;   //P0.26 GPIO, P0.26 controls dir

        LPC_PINCON->PINSEL3 = 0x00000000;   //P1.24 GPIO

        LPC_GPIO0->FIODIR |= 0x04000000;    //P0.26 output

        LPC_GPIO1->FIODIR |= 0x01000000;    //P1.24 output

        while(1)

        {

                Clock_Wise();

                for(i=0;i<300000;i++);

                AClock_Wise();

                for(i=0;i<300000;i++);

        }               //end while(1)

}                       //end main


void Clock_Wise(void)

{

        LPC_GPIO1->FIOCLR = 0x01000000;             //P1.24 Kept low to off DCM

        for(i=0;i<10000;i++);                                       //delay to componsate
inertia
```

```
        LPC_GPIO0->FIOSET = 0x04000000;             //coil is on

        LPC_GPIO1->FIOSET = 0x01000000;             //motor in on

}                                                                   //end void
Clock_Wise(void)


void AClock_Wise(void)

{

        LPC_GPIO1->FIOCLR = 0x01000000;             //P1.24 Kept low to off DCM

        for(i=0;i<10000;i++);                   //delay to componsate inertia

        LPC_GPIO0->FIOCLR = 0x04000000;             //coil is off

        LPC_GPIO1->FIOSET = 0x01000000;             //Motor is on

}                                                                   //end void
AClock_Wise(void)
```

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

```
#include <LPC17xx.H>

void clock_wise(void);

void anti_clock_wise(void);

unsigned long int var1,var2;

unsigned int i=0,j=0,k=0;

int main(void)

{

        LPC_PINCON->PINSEL4 = 0x00000000;           //P2.0 to P2.3 GPIO

        LPC_GPIO2->FIODIR = 0x0000000F;                     //P2.0 to P2.3 output

        while(1)

        {
                for(j=0;j<50;j++)                          //50 times in Clock wise Rotation

                        clock_wise();

                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();

                for(k=0;k<65000;k++);                      //Delay to show clock Rotation

        }
                                        //End of while(1)

}
                                        //End of main

void clock_wise(void)

{

                var1 = 0x00000001;                                 //For Clockwise

   for(i=0;i<=3;i++)                                 //for A B C D Stepping

        {

          LPC_GPIO2->FIOCLR =  0X0000000F;

          LPC_GPIO2->FIOSET =  var1;

                 var1 = var1<<1;                                   //For Clockwise

    for(k=0;k<15000;k++);                                          //for step speed variation

  }

}

void anti_clock_wise(void)

{

                var1 = 0x0000008;                                                  //For
Anticlockwise

   for(i=0;i<=3;i++)                                       //for A B C D Stepping
   {
            LPC_GPIO2->FIOCLR =  0X0000000F;
                         LPC_GPIO2->FIOSET =  var1;
                         var1 = var1>>1;                                          //For
Anticlockwise
     for(k=0;k<15000;k++);                                    //for step speed variation
   }
}
```

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

```
#include<LPC17xx.h>
#include "lcd.h"
#include<stdio.h>
#defineRef_Vtg                3.300
#defineFull_Scale     0xFFF                         //12 bit ADC
int main(void)
{
        unsigned int adc_temp;
        unsigned int i;
        float in_vtg;
        unsigned char vtg[7],dval[7], blank[]="   ";
        unsigned char Msg3[11] = {"ANALOG IP:"};
        unsigned char Msg4[12] = {"ADC OUTPUT:"};
```

```
lcd_init();

LPC_PINCON->PINSEL3 |= 0xC0000000;
                                                        //P1.31 as AD0.5

     LPC_SC->PCONP |= (1<<12);

     //enable the peripheral ADC

     temp1 = 0x80;

     lcd_com();

     delay_lcd(800);

     lcd_puts(&Msg3[0]);

     temp1 = 0xC0;

     lcd_com();

     delay_lcd(800);

     lcd_puts(&Msg4[0]);

     while(1)

     {

            LPC_ADC->ADCR = (1<<5)|(1<<21)|(1<<24);

     //0x01200001;//ADC0.5, start conversion and operational

            for(i=0;i<2000;i++);

            //delay for conversion

            while((adc_temp = LPC_ADC->ADGDR) == 0x80000000);
                   //wait till 'done' bit is 1, indicates conversion complete

            adc_temp = LPC_ADC->ADGDR;

            adc_temp >>= 4;

            adc_temp &= 0x00000FFF;

            //12 bit ADC
```

```
        in_vtg = (((float)adc_temp * (float)Ref_Vtg))/((float)Full_Scale);   //calculating input analog voltage

        sprintf(vtg,"%3.2fV",in_vtg);

    //convert the readings into string to display on LCD

        sprintf(dval,"%x",adc_temp);

        for(i=0;i<2000;i++);

        temp1 = 0x8A;

        lcd_com();

        delay_lcd(800);

        lcd_puts(&vtg[0]);

        temp1 = 0xCB;

        lcd_com();

        lcd_puts(&blank[0]);

        temp1 = 0xCB;

        lcd_com();

        delay_lcd(800);

        lcd_puts(&dval[0]);

        for(i=0;i<200000;i++);

        for(i=0;i<7;i++)

        vtg[i] = dval[i] = 0x00;

        adc_temp = 0;

        in_vtg = 0;

    }

}
```

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

Square Wave:

```
#include <LPC17xx.H>

void delay(void);

int main ()

{

            LPC_PINCON->PINSEL0 &= 0xFF0000FF ;
      // Configure P0.4 to P0.11 as GPIO

  LPC_GPIO0->FIODIR  |= 0x00000FF0 ;

            LPC_GPIO0->FIOMASK = 0XFFFFF00F;

      while(1)

  {

      LPC_GPIO0->FIOPIN  = 0x00000FF0 ;

    delay();

    LPC_GPIO0->FIOCLR  = 0x00000FF0 ;

    delay();

  }

}


void delay(void)

{

      unsigned int i=0;

      for(i=0;i<=9500;i++);

}
```

Triangular Wave:

```
#include <LPC17xx.H>
int main ()
{
        unsigned long int temp=0x00000000;
        unsigned int i=0;
  LPC_PINCON->PINSEL0 &= 0xFF0000FF ;        Configure P0.4 to P0.11 as GPIO
        LPC_GPIO0->FIODIR  |= 0x00000FF0 ;
        LPC_GPIO0->FIOMASK = 0XFFFFF00F;
    while(1)
    {
      for(i=0;i!=0xFF;i++)
      {
        temp=i;
        temp = temp << 4;
        LPC_GPIO0->FIOPIN = temp;
      }
      for(i=0xFF; i!=0;i--)
      {
        temp=i;
        temp = temp << 4;
        LPC_GPIO0->FIOPIN = temp;
      }
        }//End of while(1)
}//End of main()
```

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

```
#include <LPC17xx.h>

#include "lcd.h"

void scan(void);

unsigned char Msg1[14] = "ALS BENGALURU";

unsigned char Msg2[13] = "KEY PRESSED=";

unsigned char col,row,var,flag,key,*ptr;

unsigned long int i,var1,temp,temp3;

unsigned char SCAN_CODE[16] = {0x1E,0x1D,0x1B,0x17,

                                                                0x2E,0x2D,0x2B,0x27,

                                                                0x4E,0x4D,0x4B,0x47,

                                                                0x8E,0x8D,0x8B,0x87};

unsigned char ASCII_CODE[16] = {'0','1','2','3',

                                                                 '4','5','6','7',

                                                                 '8','9','A','B',

                                                                 'C','D','E','F'};

int main(void)

{

        LPC_PINCON->PINSEL3 = 0x00000000;                  //P1.20 to P1.23 MADE
GPIO

        LPC_PINCON->PINSEL0 = 0x00000000;                  //P0.15 as GPIO

        LPC_PINCON->PINSEL1 = 0x00000000;                  //P0.16 t0 P0.18 made GPIO

        LPC_GPIO0->FIODIR &= ~0x00078000;                  //made INput P0.15 to P0.18
(cols)

        LPC_GPIO1->FIODIR |= 0x00F00000;                           //made output P1.20
to P1.23 (rows)
```

```
LPC_GPIO1->FIOSET = 0x00F00000;

lcd_init();

temp1 = 0x80;
                    //point to first line of LCD

lcd_com();

delay_lcd(800);

lcd_puts(&Msg1[0]);
//display the messsage

temp1 = 0xC0;
                              //point to first line of LCD

lcd_com();

delay_lcd(800);

lcd_puts(&Msg2[0]);
//display the messsage

while(1)

{

          while(1)

          {

                    for(row=1;row<5;row++)

                    {

                              if(row == 1)

                              var1 = 0x00100000;

                              else if(row == 2)

                              var1 = 0x00200000;

                              else if(row == 3)

                              var1 = 0x00400000;

                              else if(row == 4)
```

```
        var1 = 0x00800000;

        temp = var1;

        LPC_GPIO1->FIOSET = 0x00F00000;

        LPC_GPIO1->FIOCLR = var1;

        flag = 0;

        scan();

        if(flag == 1)

        break;

    }
                    //end for(row=1;row<5;row++)

    if(flag == 1)

    break;

}
                        //2nd while(1)

for(i=0;i<16;i++)

{

    if(key == SCAN_CODE[i])

    {

        key = ASCII_CODE[i];

        break;

    }
                //end if(key == SCAN_CODE[i])

}
                    //end for(i=0;i<16;i++)

temp1 = 0xCC;

lcd_com();

delay_lcd(800);
```

```
        lcd_puts(&key);

    }
                    //end while 1

}
                        //end main

void scan(void)

{

    unsigned long temp3;

    temp3 = LPC_GPIO0->FIOPIN;

    temp3 &= 0x00078000;

    if(temp3 != 0x00078000)

    {   for(i=0;i<500;i++);

        temp3 = LPC_GPIO0->FIOPIN;

        temp3 &= 0x00078000;

        if(temp3 != 0x00078000)

        {

            flag = 1;

            temp3 >>= 15;
    //Shifted to come at LN of byte

            temp >>= 16;
    //shifted to come at HN of byte

            key = temp3|temp;

        }
                    //2nd if(temp3 != 0x00000000)

    }
                    //1st if(temp3 != 0x00000000)

}
                    //end scan
```

7. Using the Internal PWM module of ARM controller generate PWM and vary its duty cycle.

```
#include <LPC17xx.H>

void pwm_init(void);

void PWM1_IRQHandler(void);

unsigned long int i;

unsigned char flag,flag1;

int main(void)

{         pwm_init();

          while(1);

}

void pwm_init(void)

{

          LPC_SC->PCONP = (1<<6);                                   //PWM1 is powered

          LPC_PINCON->PINSEL7 = 0x000C0000;  //pwm1.2 is selected for the pin P3.25

          LPC_PWM1->PR  = 0x00000000;         //Count frequency : Fpclk

          LPC_PWM1->PCR = 0x00000400;         //select PWM2 single edge

          LPC_PWM1->MCR = 0x00000003;         //Reset and interrupt on PWMMR0

          LPC_PWM1->MR0 = 30000;         //setup match register 0 count

          LPC_PWM1->MR2 = 0x00000100;         //setup match register MR1

          LPC_PWM1->LER = 0x000000FF;         //enable shadow copy register

          LPC_PWM1->TCR = 0x00000002;         //RESET COUNTER AND PRESCALER

          LPC_PWM1->TCR = 0x00000009;         //enable PWM and counter

          NVIC_EnableIRQ(PWM1_IRQn);

}
```

```
void PWM1_IRQHandler(void)

{

        LPC_PWM1->IR = 0xff;                            //clear the interrupts

        if(flag == 0x00)

   {

        LPC_PWM1->MR2 += 100;

                LPC_PWM1->LER = 0x000000FF;

                if(LPC_PWM1->MR2 >= 27000)                  //Is Duty Cycle 90% ??

                {

        flag1 = 0xff;

        flag = 0xff;

        LPC_PWM1->LER = 0x000000fF;

                }

}    else if(flag1 == 0xff)

   {

                LPC_PWM1->MR2 -= 100;

                LPC_PWM1->LER = 0x000000fF;

                if(LPC_PWM1->MR2 <= 0x300)                  //Is Duty Cycle 1% ??

                {

                        flag  = 0x00;

                        flag1 = 0x00;

                LPC_PWM1->LER = 0X000000fF;

}

        }

}
```

8. Demonstrate the use of an external interrupt to toggle an LED On/Off.

```
#include<LPC17xx.h>

void EINT3_IRQHandler(void);

unsigned char int3_flag=0;

int main(void)

{

        LPC_PINCON->PINSEL4 |= 0x04000000;          //P2.13 as EINT3

        LPC_PINCON->PINSEL4 &= 0xFCFFFFFF;          //P2.12 GPIO for LED

        LPC_GPIO2->FIODIR = 0x00001000;             //P2.12 is assigned output

        LPC_GPIO2->FIOSET = 0x00001000;             //Initiall LED is kept on


        LPC_SC->EXTINT = 0x00000008;//writing 1 clears the interrupt, get set if there is interrupt

        LPC_SC->EXTMODE = 0x00000008;//EINT3 is initiated as edge senitive, 0 for level sensitive

        LPC_SC->EXTPOLAR = 0x00000000;//EINT3 is falling edge sensitive, 1 for rising edge

//above registers, bit0-EINT0, bit1-EINT1, bit2-EINT2,bit3-EINT3

        NVIC_EnableIRQ(EINT3_IRQn);             //core_cm3.h

        while(1) ;

}

void EINT3_IRQHandler(void)

{

                LPC_SC->EXTINT = 0x00000008;                    //cleares the interrupt
                if(int3_flag == 0x00)                                                          //when
flag is '0' off the LED

                {

                    LPC_GPIO2->FIOCLR = 0x00001000;

                    int3_flag = 0xff;

                }

                else
                                //when flag is FF on the LED

                {

                    LPC_GPIO2->FIOSET = 0x00001000;

                    int3_flag = 0;

                }

}
```

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

```
#include <LPC17xx.h>

unsigned int delay, count=0, Switchcount=0,j;

unsigned int Disp[16]={0x000003f0, 0x00000060, 0x000005b0, 0x000004f0,
0x00000660,0x000006d0,  0x000007d0, 0x00000070, 0x000007f0, 0x000006f0,
0x00000770,0x000007c0,  0x00000390, 0x000005e0, 0x00000790, 0x00000710 };

#define ALLDISP  0x00180000
            //Select all display

#define DATAPORT 0x00000ff0
            //P0.4 to P0.11 : Data lines connected to drive Seven Segments

int main (void)
```

```
{

        LPC_PINCON->PINSEL0 = 0x00000000;

        LPC_PINCON->PINSEL1 = 0x00000000;

        LPC_GPIO0->FIODIR =      0x00180ff0;

        while(1)

        {

                LPC_GPIO0->FIOSET |= ALLDISP;

                LPC_GPIO0->FIOCLR =      0x00000ff0;
                // clear the data lines to 7-segment displays

                LPC_GPIO0->FIOSET = Disp[Switchcount];  // get the 7-segment display value
from the array

                         for(j=0;j<3;j++)

                        for(delay=0;delay<30000;delay++);
        // delay

                                Switchcount++;

                                if(Switchcount == 0x10)
                                // 0 to F has been displayed ? go back to 0

                           {

                                  Switchcount = 0;

                                  LPC_GPIO0->FIOCLR  =  0x00180ff0;

                           }

        }

}
```

10. Interface a simple Switch and display its status through Relay, Buzzer and LED.

```
#include <LPC17xx.H>

int main(void)

{

        LPC_PINCON->PINSEL1 = 0x00000000;
        //P0.24,P0.25 GPIO

        LPC_GPIO0->FIODIR = 0x03000000;
                //P0.24 configured output for buzzer,P0.25 configured output for Relay/Led

  while(1)

  {

                                if(!(LPC_GPIO2->FIOPIN & 0x00000800))                   //Is
GP_SW(SW4) is pressed??

                                {

                                        LPC_GPIO0->FIOSET = 0x03000000;         //relay on

                                }

                                else

                                {

                                        LPC_GPIO0->FIOCLR = 0x03000000;         //relay off

                                }

         }

}
                                                                                                  //end
```