MES LAB MANUAL

PART A

1 ALP to multiply two 16 bit binary numbers.

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

3 ALP to find factorial of a number.

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

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

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

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

8 ALP to count the number of ones and zeros in two consecutive memory locations.

PART B

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 LEDOn/Off.

16 Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay

1) **ALP to multiply two 16 bit binary numbers**.

```
1         AREA MULTIPLY, CODE, READONLY
2    ENTRY ; Mark first instruction to execute
3       MOV R1,#0X6400 ; STORE FIRST NUMBER IN R0
4       MOV R2,#0X3200 ; STORE SECOND NUMBER IN R1
5       MUL R3,R1,R2 ; MULTIPLICATION
6    HERE B HERE
7       END
```

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

```
1       AREA INTSUM, CODE, READONLY
2   ENTRY ; Mark first instruction to execute
3      MOV R1,#10 ; LOAD 10 TO REGISTER
4      MOV R2,#0 ; EMPTY R2 REGISTER TO STORE RESULT
5   LOOP ADD R2,R2,R1 ; ADD THE CONTERNT OF R1 WITH RESULT AT R2
6      SUBS R1,#0X01 ; DECREMENT R1 BY 1
7      BNE LOOP ; REPEAT TILL R1 GOES TO ZERO
8   HERE B HERE
9     END
```

3) ALP to find factorial of a number

```
1       AREA FACTORIAL,CODE, READONLY
2    ENTRY ; MARK FIRST INSTRUCTION TO EXECUTE
3       MOV R1,#05 ; COUNTER BIT FOR 5 16BIT ADDITION
4       SUB R1,#01 ; DECREMENTED BY 1 BECAUSE WE ADD ONLY 4 TIME
5       MOV R0,#0X40000000; R0 POINTING TO 0X40000000 MEMORY LOCATION
6       LDRH R2,[R0] ; LODING HALF WORD POINTED BT RO TO R2
7    UP ADD R0,R0,#2 ; MEMORY POINTER INCREMENTED BY 2
8       LDRH R3,[R0] ; SECOND 16BIT NUMBER IS LOADED TO R3
9       ADD R2,R2,R3 ; ADDITION IS DONE
10      SUBS R1,#01 ; DECREMENTS COUNTER BIT FOR NUMBER OF ADDITION
11      BNE UP ; IF COUNTER?0 THE EXECUTION JUMPS TO THE LABEL 'UP'
12      MOV R0,#0X40000000 ; MEMORY LOCATION WHERE RESULT SHOULD BE SAVED
13      STR R2,[R0] ; STORING OF RESULT
14   HERE B HERE
15      END ; MARK END OF FILE
```

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

```
1       AREA FACTORIAL,CODE, READONLY
2   ENTRY ; MARK FIRST INSTRUCTION TO EXECUTE
3       MOV R1,#05 ; COUNTER BIT FOR 5 16BIT ADDITION
4       SUB R1,#01 ; DECREMENTED BY 1 BECAUSE WE ADD ONLY 4 TIME
5       MOV R0,#0X40000000; R0 POINTING TO 0X40000000 MEMORY LOCATION
6       LDRH R2,[R0] ; LODING HALF WORD POINTED BT RO TO R2
7   UP ADD R0,R0,#2 ; MEMORY POINTER INCREMENTED BY 2
8       LDRH R3,[R0] ; SECOND 16BIT NUMBER IS LOADED TO R3
9       ADD R2,R2,R3 ; ADDITION IS DONE
10      SUBS R1,#01 ; DECREMENTS COUNTER BIT FOR NUMBER OF ADDITION
11      BNE UP ; IF COUNTER?0 THE EXECUTION JUMPS TO THE LABEL 'UP'
12      MOV R0,#0X40000000 ; MEMORY LOCATION WHERE RESULT SHOULD BE SAVED
13      STR R2,[R0] ; STORING OF RESULT
14  HERE B HERE
15      END ; MARK END OF FILE
```

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

```
        AREA SQUARE,CODE,READONLY
START
        LDR R0,TABLE1
        MOV R1,#7
        MOV R1,R1,LSL#0x02
        MOV R4,#0x40000000
        ADD R0,R0,R1
        LDR R3,[R0]
        STR R4,[R3]
TABLE1  DCD 0X40000000
        DCD 0X40000001
        DCD 0X40000004
        DCD 0X40000009
        DCD 0X40000000
        DCD 0X40000009
        DCD 0X40000010
        DCD 0X40000019
        DCD 0X40000024
        DCD 0X40000031
        DCD 0X40000040
        DCD 0X40000051
         DCD 0X40000064
        END
```

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

```
        AREA LAR_SMAL, CODE,READONLY
ENTRY
   MOV R5,#06 ; COUNTER VALUE E.G 7 NUMBERS
   MOV R1,#0X40000000 ; START OF THE DATA MEMORY
    MOV R2,#0x4000001C ; RESULT LOCATION
    LDR R3,[R1] ; GET THE FIRST DATA
LOOP ADD R1,R1,#04 ; MEMORY POINTER UPDATED TO FETCH 2ND DATA
    LDR R4,[R1] ; GET SECOND NUMBER
    CMP R3,R4 ; COMPARE BOTH NUMBERS
    BLS LOOP1 ;BHI ? for large; IF 1ST> 2ND THAN LOOP1
    MOV R3,R4
LOOP1 SUBS R5,R5,#01 ; DECREMENT THE COUNTER
    CMP R5,#00
    BNE LOOP
    STR R3,[R2]
STOP B STOP
    END
```

7. ALP TO ARRANGE A SERIES OF 32 BIT NUMBERS IN ASCENDING/DESCENDING ORDER.

```
1        AREA ASCENDING, CODE, READONLY
2    ENTRY
3         MOV R0,#05
4    OUTTERLOOP MOV R5,#0X40000000
5                ADD R6,R5,#4
6               MOV R3,#4
7    INNERLOOP LDR R1,[R5]
8           LDR R2,[R6]
9           CMP R1,R2
10          BLO LOOP
11          MOV R4,R2
12          MOV R2,R1
13          MOV R1,R4
14   LOOP STR R1,[R5]
15        STR R2,[R6]
16        ADD R5,R5,#04
17        ADD R6,R6,#04
18        SUBS R3,R3,#01
19        BNE INNERLOOP
20        SUBS R0,R0,#1
21        BNE OUTTERLOOP
22   STOP B STOP
23       END
```

8) ALP TO COUNT THE NUMBER OF ONES AND ZEROS IN TWO CONSECUTIVE MEMORY LOCATIONS

```
1        AREA ONEZERO , CODE, READONLY
2    ENTRY ;MARK FIRST INSTRUCTION TO EXECUTE
3        MOV R2,#0 ; COUNTER FOR ONES
4        MOV R3,#0 ; COUNTER FOR ZEROS
5        MOV R6,#0X00000002; LOADS THE VALUE
6        MOV R1,#32 ; 32 BITS COUNTER
7        MOV R0,R6 ; GET THE 32 BIT VALUE
8        MOV R0,R6 ; GET THE 32 BIT VALUE
9    LOOP0 MOVS R0,R0,ROR #1 ; RIGHT SHIFT TO CHECK CARRY BIT (1'S/0'S)
10       BHI ONES ; IF C=1 GOTO ONES BRANCH OTHERWISE NEXT
11   ZEROS ADD R3,R3,#1 ; IF C= 0 THEN INCREMENT THE COUNTER BY 1(R3)
12       B LOOP1 ; BRANCH TO LOOP1
13   ONES ADD R2,R2,#1 ; IF C=1 THEN INCREMENT THE COUNTER BY 1(R2)
14   LOOP1 SUBS R1,R1,#1 ; COUNTER VALUE DECREMENTED BY 1
15       BNE LOOP0 ; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT
16   STOP B STOP
17       END
```

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

```
#include<lpc17xx.h>
unsigned char A[11]={"Hello World"};
unsigned int lsr_status;
unsigned char rec_data;
int i;
int main()
{
        LPC_PINCON->PINSEL0=0X50;
        LPC_UART0->LCR = 0x83;
        LPC_UART0->DLL = 0XA2;
        LPC_UART0->DLM = 0X00;
        LPC_UART0->LCR = 0x03;
        LPC_UART0->FCR = 0x07;
while(1)
{
        lsr_status =LPC_UART0->LSR;
        if((lsr_status & 0x20) ==0x20)
          for(i=0;i<12;i++)
                {
                  LPC_UART0->THR=A[i];
                  }
      }
}
```

10. Interface and Control a DC Motor.

```
#include <LPC17xx.H>
void Clock_Wise(void);
void AClock_Wise(void);
unsigned long i,j;
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<800000;i++);
AClock_Wise();
```

```
for(i=0;i<800000;i++);
} //end while(1)
} //end main


void Clock_Wise(void)
{
LPC_GPIO1->FIOCLR = 0x01000000; //P0.23 Kept
low to off DCM
for(j=0;j<8;j++)
{
for(i=0;i<90000;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; //P0.23 Kept low to off DCM
        for(j=0;j<8;j++)
        {
         for(i=0;i<90000;i++); //delay to componsate inertia
            LPC_GPIO0->FIOCLR = 0x04000000; //coil is off
            LPC_GPIO1->FIOSET = 0x01000000; //Motor is on
   } //end void AClock_Wise(void)
}
```

11. 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;
unsigned int i=0,j=0,k=0;
int main(void)
{
LPC_PINCON->PINSEL4 = 0x00000000;
LPC_GPIO2->FIODIR = 0x0000000F;
while(1){
          for(j=0;j<50;j++)
            clock_wise();
          for(k=0;k<65000;k++);
             for(j=0;j<50;j++)
```

```
                anti_clock_wise();
            for(k=0;k<65000;k++);
                }
        }
void clock_wise(void)
{
var1 = 0x00000001;
for(i=0;i<=3;i++) {
            LPC_GPIO2->FIOCLR = 0X0000000F;
            LPC_GPIO2->FIOSET = var1;
            var1 = var1<<1;
         for(k=0;k<15000;k++);
}
}
void anti_clock_wise(void)
{
var1 = 0x0000008;
for(i=0;i<=3;i++)
{
        LPC_GPIO2->FIOCLR = 0X0000000F;
         LPC_GPIO2->FIOSET = var1;
         var1 = var1>>1; //For Anticlockwise
         for(k=0;k<15000;k++); //for step speed variation
}
}
```

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

// DAC to generate square waveform

```
#include <LPC17xx.H>
void delay(void);
int main ()
{
LPC_PINCON->PINSEL0=0x00000000 ;
// Configure P0.4 to P0.11 as GPIO
LPC_GPIO0->FIODIR = 0x00000FF0 ;
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++);
}
```

// DAC to generate triangular waveform

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

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

```
#include <LPC214x.H> /* LPC214x definitions */
#include "lcd.h"
// Matrix Keypad Scanning Routine
// COL1 COL2 COL3 COL4
// 0 1 2 3 ROW 1
// 4 5 6 7 ROW 2
// 8 9 A B ROW 3
// C D E F ROW 4
#define SEG7_CTRL_DIR IO0DIR
#define SEG7_CTRL_SET IO0SET
#define SEG7_CTRL_CLR IO0CLR
#define COL1 (1 << 16)
#define COL2 (1 << 17)
#define COL3 (1 << 18)
#define COL4 (1 << 19)
```

```
#define ROW1 (1 << 20)
#define ROW2 (1 << 21)
#define ROW3 (1 << 22)
#define ROW4 (1 << 23)
#define COLMASK (COL1 | COL2 | COL3 | COL4)
#define ROWMASK (ROW1 | ROW2 | ROW3 | ROW4)
#define KEY_CTRL_DIR IO1DIR
#define KEY_CTRL_SET IO1SET
#define KEY_CTRL_CLR IO1CLR
#define KEY_CTRL_PIN IO1PIN
/////////////// COLUMN WRITE ////////////////////
Void col_write( unsigned char data )
{
Unsigned int temp=0;
Temp=(data << 16) & COLMASK;
KEY_CTRL_CLR |= COLMASK;
KEY_CTRL_SET |= temp;
}
////////////////////////////// MAIN ////////////////////////////////////
Int main (void)
{
Unsigned char key, i;
Unsigned char rval[] = {0x7,0xb,0xd,0xe,0x0};
Unsigned char keypadmatrix[] =
{
'4','8','B','F',
'3','7','A','E',
'2','6','0','D',
'1','5','9','C'
};
Init_lcd();
KEY_CTRL_DIR |= COLMASK; //Set cols as Outputs
KEY_CTRL_DIR &= ~(ROWMASK); // Set ROW lines as Inputs
Lcd_putstring16(0,"Press HEX Keys..");
Lcd_putstring16(1,"Key Pressed = ");
While (1)
{
Key = 0;
For( i = 0; i < 4; i++ )
{
// turn on COL output one by one
Col_write(rval[i]);
// read rows - break when key press detected
If (!(KEY_CTRL_PIN & ROW1))
Break;
```

```
Key++;
If (!(KEY_CTRL_PIN & ROW2))
Break;
Key++;
If (!(KEY_CTRL_PIN & ROW3))
Break;
Key++;
If (!(KEY_CTRL_PIN & ROW4))
Break;
Key++;
}
If (key == 0x10)
Lcd_putstring16(1,"Key Pressed = ");
Else
{ lcd_gotoxy(1,14);
Lcd_putchar(keypadmatrix[key]);
}
}
}
```

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

a)Demonstrate the Blinking LED program.

```
#include<lpc17xx.h>
int delay_cnt;
int main()
{
LPC_PINCON->PINSEL4=0x00000000;
LPC_GPIO2->FIODIR=0x1000;
while(1)
{
LPC_GPIO2->FIOCLR=0x1000;
for(delay_cnt=0;delay_cnt<30000;delay_cnt++);
LPC_GPIO2->FIOSET=0x1000;
for(delay_cnt=0;delay_cnt<30000;delay_cnt++);
}
}
```

b) //Write a C program to control LED (P2.12) ON/OFF using key press P2.11

```
#include<lpc17xx.h>
unsigned int i;
int main()
{
LPC_PINCON->PINSEL4=0x00000000;
LPC_GPIO2->FIODIR =0x1000;
```

```
while(1)
{
i=LPC_GPIO2->FIOPIN;//read p2 for key
i = i & 0x800;
if(i==0) //if key pressed
{
LPC_GPIO2->FIOCLR =0x1000;
}
else//if key is released
{
LPC_GPIO2->FIOSET =0x1000;
}
36
}
}
```

c)

```
#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 word length
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
}
void UART0_IRQHandler(void)
{
unsigned long Int_Stat;
Int_Stat = LPC_UART0->IIR; //reading the data from interrupt
identification register
Int_Stat = Int_Stat & 0x06;//masking other than txmit int & rcve data indicator
if((Int_Stat & 0x02)== 0x02) //transmit interrupt
tx0_flag = 0xff;
}
void delay(unsigned int r1)
{
for(r=0;r<r1;r++);
}
```

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

```
#include<lpc17xx.h>

int delay_cnt,Switchcount=0,j;

unsigned int Disp[17]={0x000003f0, 0x00000060, 0x000005b0,

0x000004f0, 0x00000660,0x000006d0,

0x000007d0, 0x00000070, 0x000007f0, 0x000006f0,

0x00000770,0x000007c0, 0x00000390, 0x000005e0, 0x00000790,

0x00000710 };

int main()

{

LPC_PINCON->PINSEL0=0x00000000;
```

```
LPC_PINCON->PINSEL1=0x00000000;
LPC_GPIO0->FIODIR=0x180ff0;
LPC_GPIO0->FIOSET=0x00080000;
while(1)
{
LPC_GPIO0->FIOCLR=0x00000ff0;
LPC_GPIO0->FIOSET = Disp[Switchcount];
for(j=0;j<7;j++)
for(delay_cnt=0;delay_cnt<30000;delay_cnt++); //
1s delay
Switchcount++;
}
}
```

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

```
#INCLUDE<LPC214X.H>
#DEFINE RS 0X00400000
#DEFINE RW 0X20000000
#DEFINE EN 0X10000000
UNSIGNED INT RESULT;
FLOAT VOLTAGE;
CHAR VOLT[18];
VOID DELAY(UNSIGNED INT X)
{
UNSIGNED INT I,J;
FOR(I=0;I<X;I++)
FOR(J=0;J<1275;J++);
}
VOID CMD( CHAR C)
{
   IOCLR0=0X00003FC0;
IOSET0=C<<6;
IOCLR0=RW;
IOCLR0=RS;
IOSET0=EN;
DELAY(100);
IOCLR0=EN;
}
VOID DATA( CHAR C)
{
```

```
IOCLR0=0X00003FC0;
IOSET0=C<<6;
IOCLR0=RW;
IOSET0=RS;
IOSET0=EN;
DELAY(100);
IOCLR0=EN;
}
VOID LCD_STR(CHAR *S)
{
WHILE(*S)
{
DATA(*S);
S++;
DELAY(20);
}
}
VOID ADC_INIT()
{
AD0CR=0X00210308;
PINSEL1=0X10000000;
}
```