Adichunchanagiri Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Chikkamagaluru-577102

LAB MANUAL

(2018-19)

MICROCONTROLLER AND EMBEDDED SYSTEMS LABORATORY

(18CSL48)

IV Semester

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi, Karnataka –590018

IV SEMESTER

MICROCONTROLLER AND EMBEDDED SYSTEMS

LABORATORY

(18CSL48)

Mrs. Deepashri K.S B.E.,M.Tech.,
Asst. Professor
Dept. of IS&E
A.I.T Chikkamagaluru

Dr. Sampath S M.Tech., PhD
Professor & Head
Dept. of IS&E
A.I.T Chikkamagaluru

MICROCONTROLLER AND EMBEDDED SYSTEMS LABORATORY

(Effective from the academic year 2018 -2019)

SEMESTER – IV

Course Code 18CSL48 **CIE Marks** 40

Number of Contact Hours/Week 0:2:2 **SEE Marks** 60

Total Number of Lab Contact Hours 36 **Exam Hours** 03

Credits – 2

Course Learning Objectives: This course (18CSL48) will enable students to:

- Develop and test Program using ARM7TDMI/LPC2148
- Conduct the experiments on an ARM7TDMI/LPC2148 evaluation board usingevaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

Program List		
PART A Conduct the following experiments by writing program using ARM7TDMI/LPC2148 using an evaluation board/simulator and the required software tool.		
1	Write a program to multiply two 16 bit binary numbers	1
2	Write a program to find the sum of first 10 integer numbers.	2
3	Write a program to find factorial of a number	3
4	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM	4
5	Write a program to find the square of a number (1 to 10) using look-up table.	6
6	Write a program to find the largest/smallest number in an array of 32 numbers .	8
7	Write a program to arrange a series of 32 bit numbers in ascending/descending order	10
8	Write a program to count the number of ones and zeros in two consecutive memory locations.	13
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.	20
10	Interface and Control a DC Motor.	22
11	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.	24
12	Determine Digital output for a given Analog input using Internal ADC of ARM controller.	25
13	Interface a DAC and generate Triangular and Square waveforms	31
14	Interface a 4x4 keyboard and display the key code on an LCD.	33
15	Demonstrate the use of an external interrupt to toggle an LED On/Off.	41
16	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between	42

Laboratory Outcomes: The student should be able to:

- Develop and test program using ARM7TDMI/LPC2148
- Conduct the following experiments on an ARM7TDMI/LPC2148 evaluation boardusing evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

1) **Write a program to multiply two 16 bit binary numbers**

```
AREA MULTIPLY,CODE,READONLY
ENTRY
START
        MOV R1,#20          ;MOVE 20 VALUE TO R1 REGISTER
        MOV R2,#220         ; MOVE 20 VALUE TO R2 REGISTER
        MUL R3,R1,R2        ;MULTIPLY R1 & R2 AND STORE RESULT IN R3 EGISTER

        NOP
        NOP
        NOP
        END
```

Output

	Register	**Value**
Input	R1	0X0020
Input	R2	0X0220
Output	R3	0X4400

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

```
            AREA SUM, CODE, READONLY

  ENTRY

              MOV R1,#10                  ; LOAD 10 TO REGISTER

               MOV R2,#0                  ; EMPTY THE REGISTER TO STORE RESULT

LOOP

            ADD R2, R2, R1                ; ADD THE CONTENT OF R1 WITH RESULT AT R2

            SUBS R1,#0x01                 ; DECREMENT R1 BY 1

            BNE LOOP                      ; REPEAT TILL R1 GOES 0


            BACK B BACK                   ; JUMPS BACK TO C CODE

  END
```

Output

	Register	Value
Input	R1	10
Output	R2	0X00000037

3) Program to find factorial of a given 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

        MUL R3, R0,R1               ; MULTIPLICATION
        MOV   R0, R3                ; RESULT R3 MOVED TO R0
        CMP R1, #1                  ; COMPARISON R1 WITH 1
        BNE FACT                    ; BRANCH TO THE FACT IF NOT EQUAL
        NOP
        NOP
        NOP

        END                         ; MARK END OF FILE
```

Output

	Register	Value
Input	R0	7
Output	R0	13B0H (5040)

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

```
        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
        LDRH R3,[R1],#02            ; READ 16 BIT DATA
        ADD R0,R0,R3                ; ADD R2=R2+R3
        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
JMP B JMP

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
```

OUTPUT

Variable	Address	Value
(N)R5		6
VALUE1	0X0000002C	0X1111
	0X0000002E	0x2222
	0X00000030	0x33333
	0X00000032	0XAAAA
	0X00000034	0XBBBB
	0X00000036	0XCCCC
RESULT	0X40000000	29997H(R0)

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

```
AREA SQUARE, CODE, READONLY

ENTRY                           ; MARK FIRST INSTRUCTION TO EXECUTE

START
        LDR    R0, = TABLE1     ; LOAD START ADDRESS OF LOOKUP TABLE
        LDR R1,=8               ; LOAD NO WHOSE SQUARE IS TO BE FIND
        MOV R1, R1, LSL#0X2     ; GENERATE ADDRESS CORRESPONDING TO
                                SQUARE OF GIVEN NO
        ADD R0, R0, R1          ; LOAD ADDRESS OF ELEMENT IN LOOKUP TABLE
        LDR R3, [R0]            ; GET SQUARE OF GIVEN NO IN R3
        NOP
        NOP
        NOP

;LOOKUP TABLE CONTAINS SQUARES OF NOS FROM 0 TO 10 (IN HEX)
TABLE1      DCD   0X00000000        ; SQUARE OF 0=0
            DCD 0X00000001          ; SQUARE OF 1=1
            DCD 0X00000004          ; SQUARE OF 2=4
            DCD 0X00000009          ; SQUARE OF 3=9
            DCD 0X00000010          ; SQUARE OF 4=16
            DCD 0X00000019          ; SQUARE OF 5=25
            DCD 0X00000024          ; SQUARE OF 6=36
            DCD 0X00000031          ; SQUARE OF 7=49
            DCD 0X00000040          ; SQUARE OF 8=64
            DCD 0X00000051          ; SQUARE OF 9=81
            DCD 0X00000064          ; SQUARE OF 10=100

            END                     ; MARK END OF FILE
```

OUTPUT

Variable	Address	Value
TABLE1	0X00000020(R0)	0X00000001
	0X00000024	0X00000002
	0X00000028	0X00000004
	0X0000002C	0X00000009
	0X00000030	0X00000010
	0X00000034	0X00000019
	0X00000038	0X00000024
	0X0000003C	0X00000031
	0X00000040	0X00000040
	0X00000044	0X00000051
	0X00000048	0X00000064
R1(INPUT)		8
R3(OUTPUT)		0X00000040

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

```
        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
        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
```

OUTPUT

Variable	Address	Value
VALUE1	0X00000020(R0)	0X44444444
	0X00000024	0X22222222
	0X00000028	0X11111111
	0X0000002C	0X33333333
	0X00000030	0XAAAAAAAA
	0X00000034	0X88888888
	0X00000038	0X99999999
RESULT(SMALL)	0X40000000	0X11111111
RESULT(LARGE)	0X40000000	0XAAAAAAA

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

```
        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
```

OUTPUT

Before Sorting

Variable	Address	Value
INPUT	0X40000000 (R0)	0X44444444
	0X40000004	0X11111111
	0X40000008	0X33333333
	0X4000000C	0X22222222

After Sorting

Variable	Address	Value
INPUT	0X40000000 (R0)	0X11111111
	0X40000004	0X22222222
	0X40000008	0X33333333
	0X4000000C	0X44444444

NOTE: for Descending order change instruction BLT to BGT (Branch if Greater than)

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

```
  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
             LDR R4, =RES1
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 N EXT

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
             STR R2,[R4],#4
             STR R3,[R4],#4                     ; 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
```

```
                STR R2,[R4],#4
                STR R3,[R4],#4

   JMP b JMP



VALUE   DCD   0X3,   0X2                    ;   TWO VALUES IN AN ARRAY
                AREA DATA2,DATA,READWRITE
RES1   DCD    0X0
RES2   DCD 0X0

        END                                      ; Mark end of file
```

OUTPUT

	ADDRESS	REGISTERS	VALUE
INPUT	0X40000000		0X00000011
	0X40000004		0X00000010
OUTPUT		R2(FOR ONE’S)	0X00000003
		R3(FOR ZERO’S)	0X0000001D

PART –B

Conduct the following experiments on an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

- ARM-Advanced RISC Machine is a 32-bit RISC (Reduced Instruction Set computer) processor architecture developed by ARM Holdings
- ARM7 is most successful and widely used processor family in embedded system applications.
- ARM7 TDMI based NXP controller LPC2148
- LPC2148 is manufactured by NXP Semiconductor (Phillips) and it is preloaded with many in-built features and peripherals.

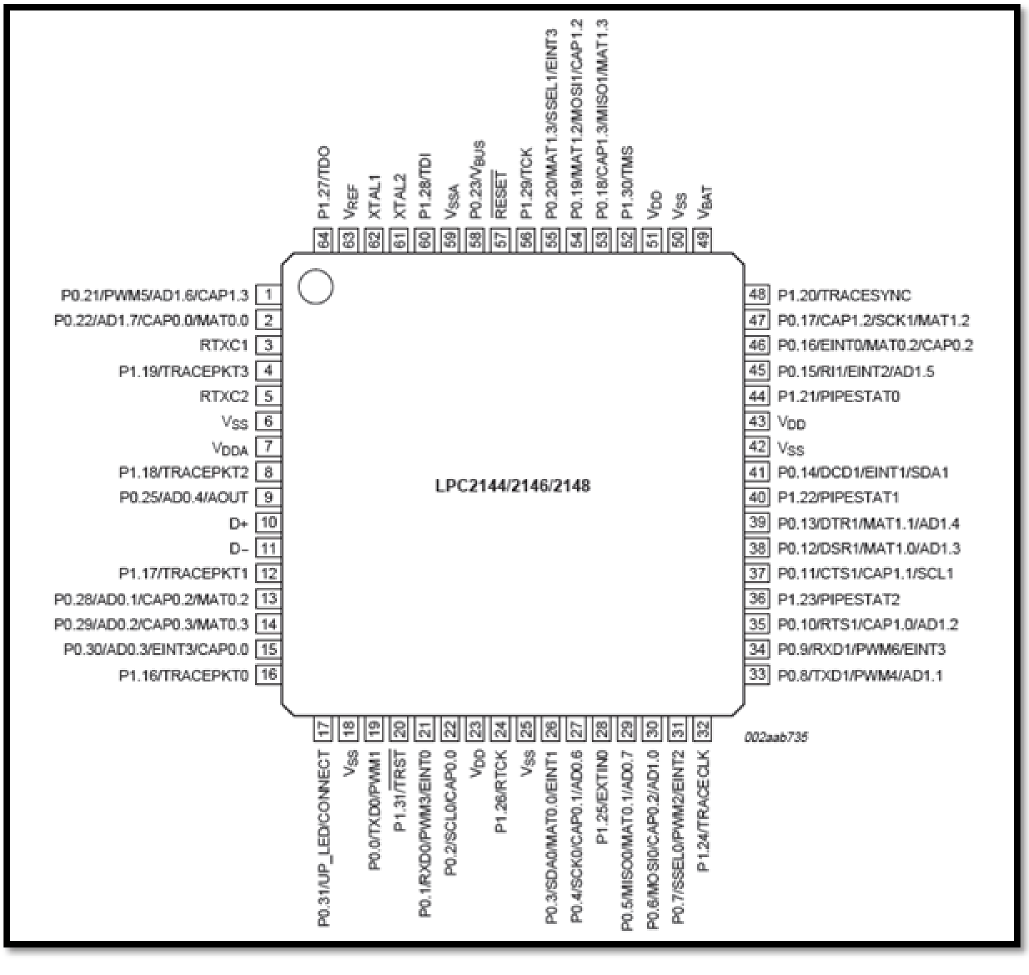

Figure 1:LPC2148

- General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit).
- It can be either input pin or output pin, whose behaviour can be controlled at the run time. A group of these pins is called a port.
- (Example, Port 0 of LPC2148 has 32

pins). LPC2148 has two 32-bit General Purpose

I/O ports.

- 1. **PORT0**
- 2. **PORT1**

PORT0 is a 32-bit

- Out of these 32 pins, 28 pins can be configured as either general purpose input or output.
- 1 of these 32 pins (P0.31) can be configured as general-purpose output only.
- 3 of these 32 pins (P0.24, P0.26 and P0.27) are reserved. Hence, they are not available for use. Also, these pins are not mentioned in pin diagram.

PORT1 is also a 32-bit port. Only 16 of these 32 pins (P1.16 – P1.31) are available for use as general-purpose input or output.

Slow GPIO Registers: There are 4 slow GPIO Registers

- **IOxPIN (GPIO Port Pin value register):** Read/write the value on Port (PORT0/PORT1)
- **IOxSET (GPIO Port Output Set register) :** Port (PORT0/PORT1) HIGH , writing specific bits
- **IOxDIR (GPIO Port Direction control register) :**Configures the corresponding pin as an output pin/Input pin (0 input,1output)
- **IOxCLR (GPIO Port Output Clear register) :** Port (PORT0/PORT1) HIGH , writing specific bits
- **Where x is 0 or 1 indicates port0 are port1**
- Examples :

a) Configure pin P0.0 to P0.3 as input pins and P0.4 to P0.7 as output

pins. IO0DIR = 0x000000F0;

b) Configure pin P0.4 as an output. Then set that pin HIGH.
IO0DIR = 0x00000010; OR IO0DIR =

(1<<4); IO0SET = (1<<4);

c) Configure pin P0.4 as an output. Then set that pin

LOW. IO0DIR = 0x00000010; OR IO0DIR = (1<<4);

IO0CLR = (1<<4);

Fast GPIO Registers

- **FIOxDIR (Fast GPIO Port Direction control register) :**
- **FIOxMASK (Fast Mask register for port) :**
- **FIOxPIN (Fast Port Pin value register using FIOMASK) :**
- **FIOxSET (Fast Port Output Set register using FIOMASK):**
- **FIOxCLR (Fast Port Output Clear register using FIOMASK**

PROJECT CREATION IN KEILUV4 IDE:

Steps

1. Create a project folder before creating NEW project.
2. Open Keil uVision4 IDE software by double clicking on “Keil Uvision4” icon.
3. Go to “Project” then to “New uVision Project” and save it with a name in the respective project folder, already you created.
4. Select the device as “NXP” In that “LPC2148” then press OK and then press “YES” button to add “startup.s” file.
5. In startup file go to Configuration Wizard. In Configuration Wizard window uncheck PLL Setup and check VPBDIV Setup.
6. Go to “File” In that “New” to open an editor window. Create your source file and use the header file “lpc21xx.h” in the source file and save the file. Colour syntax highlighting will be enabled once the file is saved with a extension such as “.C “.
7. Right click on “Source Group 1” and select the option “Add Existing Files to Group Source Group 1“add the *.C source file(s) to the group.
8. After adding the source file you can see the file in Project Window.
9. Then go to “Project” in that “Translate” to compile the File (s). Check out the

Build output window.

10. Right click on Target1 and select options for Target Target1.Then go to option "Target" in that
    a. Xtal 12.0MHz
    b. Select "Use MicroLIB".
    c. Select IROM1 (starting 0x0 size 0x80000).
    d. Select IRAM1 (starting 0x40000000 size 0x8000).
    e. Then go to option "Output"
    f. Select "Create Hex file".
    g. Then go to option "Linker"
    h. Select "Use Memory Layout for Target Dialog". To come out of this window press OK.
11. Go to "Project" in that "Build Target" for building all source files such as ".C",".ASM", ".h", files, etc…This will create the *.HEX file if no warnings & no Errors. Check out the Build output window.
12. FLASH MAGIC software can be used to download the HEX files to the Flash memory of controller.

How to Download?

Connect the serial cross cable from 9-pin DSUB Female connector (DB2) to the PC COM port. Push both SW11/1, 2 to ON position, JP7 should be shorted while using ISP programming. Connect DC +5V Power, through the 9-pin DSUB Male connector (DB1) applied from an external source. Switch ON the power. Open JP13 while downloading the software.

Settings in FLASH MAGIC:

Options -> Advanced options -> Hardware
config enable these options only
Use DTR and RTS to control RST and ISP
pin Keep RTS asserted while COM port

open Press OK then do the below settings

Step1. Communications:

1. Device : LPC2148
2. Com Port : COM1
3. Baud Rate : 9600
4. Interface : None(ISP)
5. Oscillator :

12MHz Step2. ERASE:

1. Select “Erase Blocks Used By Hex File”.

Step3. Hex file:

1. Browse and select the Hex file which you want to

download. Step4. Options

1. Select “Verify after programming”.

Step5. Start:

1. Click Start to download the hex file to the controller.

After downloading the code the program starts executing in the hardware, then remove the ISP jumper JP7.

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

```
#include
<LPC213X.H>
#include <stdint.h>
void UART0_init(void)
{
        PINSEL0 = PINSEL0 | 0x00000005;
        /* Enable UART0 Rx0 and Tx0 pins of UART0 */
        U0LCR = 0x83;           /* DLAB = 1, 1 stop bit, 8-bit character
        length */U0DLM = 0x00;   /* For baud rate of 9600 with Pclk =
        15MHz */
        U0DLL = 0x61;           /* We get these values of U0DLL and U0DLM from
        formula */U0LCR = 0x03; /* DLAB = 0 */
}
unsigned char UART0_RxChar(void) /*A function to receive a byte on UART0 */
{
        while( (U0LSR & 0x01) == 0);
                /*Wait till RDR bit becomes 1 which tells that receiver
                contains valid data */return U0RBR;
}
void UART0_TxChar(char ch) /*A function to send a byte on UART0 */
{
        U0THR = ch;
        while( (U0LSR & 0x60) == 0 );
/* Wait till THRE bit becomes 1 which tells that transmission is completed */
}
void UART0_SendString( char *p)
{         char c;
while(*p!='\0')

        {
              c=*p;
             p++;
              UART0_TxChar(c);

        }

}

int main(void)

{

        char

        receive;

        UART0_in

        it();

        while(1)

        {

                receive = UART0_RxChar();

                UART0_SendString("Received

                :");UART0_TxChar(receive);

                UART0_SendString("\r\n");

        }

}
```

10. Interface and Control a DC Motor.

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

int main()
{
    IO0DIR= 0X00000900;
    IO0SET= 0X00000100;                    //P0.8 should always high.

    while(1)
    {
    clock_wise();
    for(j=0;j<400000;j++);            //delay
    anti_clock_wise();
    for(j=0;j<400000;j++);            //delay
    }                                 //End of while(1)
}                                     //End of Main

void clock_wise(void)
{
    IO0CLR = 0x00000900;              //stop motor and also turn off relay
    for(j=0;j<10000;j++);             //small delay to allow motor to turn off
    IO0SET = 0X00000900;        //Selecting the P0.11 line for clockwise and turn on motor
}

void anti_clock_wise(void)
{
    IO0CLR = 0X00000900;              //stop motor and also turn off relay
    for(j=0;j<10000;j++);             //small delay to allow motor to turn off
    IO0SET = 0X00000100;              //not selecting the P0.11 line for Anti clockwise
}
```

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

```
**********************************************************************
* A stepper motor direction is controlled by shifting the voltage across
 * the coils. Port lines : P0.12 to P0.15
 ***************************************************************************/

 #include <LPC21xx.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)
 {
     PINSEL0 = 0x00FFFFFF;                //P0.12 to P0.15 GPIo
     IO0DIR |= 0x0000F000;                //P0.12 to P0.15 output

     while(1)
     {
       for(j=0;j<50;j++)       // 20 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++)       // 20 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 = 0x00000800;                //For Clockwise
   for(i=0;i<=3;i++)          // for A B C D Stepping
     {
       var1 = var1<<1;          //For Clockwise
       var2 = ~var1;
       var2 = var2 & 0x0000F000;

       IO0PIN = ~var2;
              for(k=0;k<3000;k++);    //for step speed variation
     }
 }
void anti_clock_wise(void)
 {
     var1 = 0x00010000;                //For Anticlockwise
    for(i=0;i<=3;i++)         // for A B C D Stepping
     {
       var1 = var1>>1;          //For Anticlockwise
       var2 = ~var1;
       var2 = var2 & 0x0000F000;

       IO0PIN = ~var2;
       for(k=0;k<3000;k++);           //for step speed variation

     }
 }
```

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

```
//10-bit internal ADC //AIN0 pin is selected //you can change the channel by changing
PINSEL1 and ADCR value              //

#include <lpc214x.h>
#include <Stdio.h>

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);

unsigned int data_lcd=0;
unsigned int adc_value=0,temp_adc=0,temp1,temp2;
float temp;
char var[15],var1[15];
char *ptr,arr[]= "ADC O/P= ";
char *ptr1,dis[]="A I/P  = ";

#define vol 3.3                  //Reference voltage
#define fullscale 0x3ff          //10 bit adc

int main()
{
    PINSEL1 = 0X00040000;        //AD0.4 pin is selected(P0.25)
   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 first line 1 th pos
  lcd_com();
  delay(800);

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

  ptr1 = arr;
  temp1 = 0xC0;                            //Display starting address of second line 4 th pos
  lcd_com();
  delay(800);

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

//infinite loop
while(1)
{
    //CONTROL register for ADC
    AD0CR = 0x01200010;                                //command 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;
        sprintf(var1,"%4.2fV",temp);
        sprintf(var,"%3x",adc_value);

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

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

        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;          // E=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;                    // E=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);
}
```

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

i)Triangular Waveform program

```
#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()
```

i)Square Waveform program

```
// 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<=95000;i++);
}
```

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

```
/*Program to demonstrate keyboard operation  Takes a key from key board and displays it
on LCD screen*/

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


/******* FUNCTION PROTOTYPE*******/
void lcd_init(void);
void clr_disp(void);
void lcd_com(void);
void lcd_data(void);
void wr_cn(void);
void wr_dn(void);
void scan(void);
void get_key(void);
void display(void);
void delay(unsigned int);
void init_port(void);

unsigned long int scan_code[16]= {0x00EE0000,0x00ED0000,0x00EB0000,0x00E70000,
                        0x00DE0000,0x00DD0000,0x00DB0000,0x00D70000,
                        0x00BE0000,0x00BD0000,0x00BB0000,0x00B70000,
                        0x007E0000,0x007D0000,0x007B0000,0x00770000};



unsigned char ASCII_CODE[16]= {'0','1','2','3',
                      '4','5','6','7',
                      '8','9','A','B',
                      'C','D','E','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,disp[] = "4X4 KEYPAD";
unsigned char disp0[] = "KEYPAD TESTING";
unsigned char disp1[] = "KEY = ";
int main()
{
   //   __ARMLIB_enableIRQ();
  init_port();                    //port intialisation
  delay(3200);                  //delay
  lcd_init();                     //lcd intialisation
  delay(3200);                  //delay
    clr_disp();                   //clear display
  delay(500);                    //delay

  //........LCD DISPLAY TEST.........//
  ptr = disp;
  temp1 = 0x81;              // Display starting address
  lcd_com();
  delay(800);

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

  //........KEYPAD Working.........//
  while(1)
            {
            get_key();
```

```
                display();
            }

    }  //end of main()

    void get_key(void)              //get the key from the keyboard
    {
            unsigned int  i;
            flag = 0x00;
            IO1PIN=0x000f0000;
            while(1)
            {
                    for(row=0X00;row<0X04;row++)     //Writing one for col's
                    {
                            if( row == 0X00)
                            {
                                    temp3=0x00700000;
                            }
                            else if(row == 0X01)
                                    {
                                    temp3=0x00B00000;
                                    }
                            else if(row == 0X02)
                            {
                                    temp3=0x00D00000;
                            }
                            else if(row == 0X03)
                            {
                                    temp3=0x00E00000;
                            }
                    var1 = temp3;
                    IO1PIN = var1;                  // each time var1 value is put to port1
                    IO1CLR =~var1;        // Once again Conforming (clearing all other bits)
```

```
                        scan();
                        delay(100);                          //delay
                        if(flag == 0xff)
                                break;
                } // end of for
                if(flag == 0xff)
                break;
        }  // end of while

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

       void scan(void)
       {
         unsigned long int t;
         temp2 = IO1PIN;                         // status of port1
         temp2 = temp2 & 0x000F0000;                  // Verifying column key
         if(temp2 != 0x000F0000)                    // Check for Key Press or Not
         {
                delay(1000);                          //delay(100)//give debounce delay check again
                temp2 = IO1PIN;
                temp2 = temp2 & 0x000F0000;                //changed condition is same

            if(temp2 != 0x000F0000)               // store the value in res1
            {
                flag = 0xff;
            res1 = temp2;
            t = (temp3 & 0x00F00000);         //Verfying Row Write
            res1 = res1 | t;                  //final scan value is stored in res1
        }
        else
        {
            flag = 0x00;
        }
    }
}  // end of scan()

void display(void)
{
    ptr = disp0;
  temp1 = 0x80;                       // Display starting address of first line
  lcd_com();

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

    ptr = disp1;
  temp1 = 0xC0;                               // Display starting address of second line
  lcd_com();

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

// load command for lcd function setting with lcd in 4 bit mode,
// 2 line and 5x7 matrix display

  temp = 0x28;
  lcd_com();
  delay(3200);

// load a command for display on, cursor on and blinking off
```

```
  temp1 = 0x0C;
  lcd_com();
  delay(800);

// command for cursor increment after data dump
  temp1 = 0x06;
  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;        // E=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;        // E=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 = 0x000000FC;       //configure o/p lines for lcd
  IO1DIR = 0XFFF0FFFF;
}
```

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

```
#include <LPC21xx.h>
unsigned int delay;
int main ()
{
        PINSEL1 = 0x00000000 ;              // Configure P0.16 to P0.31 as GPIO
        IO0DIR  = 0x00FF0000 ;              // Configure P0.16 to P0.23 as Output
          while(1)
          {
               IO0CLR = 0x00FF0000;         // CLEAR (0) P0.10 to P0.13 and P0.18 to P0.21,
          LEDs ON
               for(delay=0; delay<500000; delay++);        // delay
               IO0SET = 0x00FF0000;         // SET (1) P0.10 to P0.13 and P0.18 to P0.21, LEDs
          OFF
               for(delay=0; delay<500000; delay++);        // delay
          }

}
```

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

```
\\\\\\\\\\\\\\DISPLAY ARE CONNECTED IN COMMON CATHODE MODE\\\\\\\\\\\\\\\\\\\\
        Port0 Connected to data lines of all 7 segement displays
          a
         ----
        f|  g |b
         |----|
        e|    |c
         ----  . dot
          d
        a = P0.16
        b = P0.17
        c = P0.18
        d = P0.19
        e = P0.20
        f = P0.21
        g = P0.22
        dot = P0.23

                Select lines for four 7 Segments
        DIS1    P0.28
        DIS2    P0.29
        DIS3    P0.30
        DIS4    P0.31

        Values Correspoinding to Alphabets 1, 2, 3 and 4


#include <LPC21XX.h>
```

```
unsigned int delay;
unsigned int Switchcount=0;
unsigned int Disp[16]={0x003F0000, 0x00060000, 0x005B0000, 0x004F0000,
0x00660000,0x006D0000,  0x007D0000, 0x00070000, 0x007F0000, 0x006F0000,
0x00770000,0x007C0000,   0x00390000, 0x005E0000, 0x00790000, 0x00710000 };

#define SELDISP1 0x10000000                 //P0.28
#define SELDISP2 0x20000000                //P0.29
#define SELDISP3 0x40000000                //P0.30
#define SELDISP4 0x80000000                //P0.31
#define ALLDISP 0xF0000000                 //Select all display
#define DATAPORT 0x00FF0000                //P0.16 to P0.23 Data lines connected to drive Seven Segments

int main (void)
{
        PINSEL0 = 0x00000000;
        PINSEL1 = 0x00000000;
        IO0DIR = 0xF0FF0000;
        IO1DIR = 0x00000000;

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

                if(!(IO1PIN & 0x00800000))         // if the key is pressed
                {
                        for(delay=0;delay<100000;delay++); // delay


                   if((IO1PIN & 0x00800000))   // check to see if key has been released
                        {
```

```
                                        Switchcount++;
                                        if(Switchcount == 0x10)        // 0 to F has been displayed ? go back to 0
                                    {
                                          Switchcount = 0;
                                          IO0CLR =0xF0FF0000;
                                    }
                                }
                        }
                }
}
```